DrunkWalk: Collaborative and Adaptive Planning for Navigation of Micro-Aerial Sensor Swarms

Xinlei Chen
Carnegie Mellon University
Pittsburgh, PA, USA
xinlei.chen@sv.cmu.edu

Aveek Purohit
Carnegie Mellon University
Pittsburgh, PA, USA
imaveek@gmail.com

Carlos Ruiz Dominguez
Carnegie Mellon University
Pittsburgh, PA, USA
carlos.ruiz.dominguez@sv.cmu.edu

Stefano Carpin
UC Merced
Merced, CA, USA
scarpin@ucmerced.edu

Pei Zhang
Carnegie Mellon University
Pittsburgh, PA, USA
peizhang@cmu.edu

ABSTRACT

Micro-aerial vehicle (MAV) swarms are a new class of mobile sensor networks with many applications, including search and rescue, urban surveillance, radiation monitoring, etc. These sensing applications require autonomously navigating a high number of low-cost, low-complexity MAV sensor nodes in hazardous environments. The lack of preexisting localization infrastructure and the limited sensing, computing, and communication abilities of individual nodes makes it challenging for nodes to autonomously navigate to suitable preassigned locations.

In this paper, we present a collaborative and adaptive algorithm for resource-constrained MAV nodes to quickly and efficiently navigate to preassigned locations. Using radio fingerprints between flying and landed MAVs acting as radio beacons, the algorithm detects intersections in trajectories of mobile nodes. The algorithm combines noisy dead-reckoning measurements from multiple MAVs at detected intersections to improve the accuracy of the MAVs' location estimations. In addition, the algorithm plans intersecting trajectories of MAV nodes to aid the location estimation and provide desired performance in terms of timeliness and accuracy of navigation. We evaluate the performance of our algorithm through a real testbed implementation and large-scale physical feature based simulations. Our results show that, compared to existing autonomous navigation strategies, our algorithm achieves up to 6× reduction in location estimation errors, and as much as 3× improvement in navigation success rate under the given time and accuracy constraints.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; I.2 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

Keywords

Mobile Sensor Networks, Micro-Aerial Vehicle, Swarm

SenSys'15, November 1–4, 2015, Seoul, South Korea..

DOI: http://dx.doi.org/10.1145/2809695.2809724

1. INTRODUCTION

Many hostile, dangerous, or otherwise inaccessible environments (such as urban search and rescue, environmental monitoring, surveillance, etc.), situational awareness is needed. However, in these dangerous scenarios, manual deployment of sensors is often not feasible.

In such scenarios, autonomously navigating MAV swarms to a set of goal locations, in accordance with the needs of domain experts, can provide significant benefit. Further, utilizing a large number of low-cost, low-complexity mobile sensor nodes, as opposed to using a limited number of sophisticated robots, can be more cost effective and provide increased robustness through redundancy. 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.

MAV swarms are an emerging class of networked mobile systems with widespread applications in such domains. These swarms consist of miniature aerial sensor nodes with limited individual sensing, computing and communication capabilities [1, 2]. Initial work in the operation of MAVs has focused on outdoor or highly instrumented environments that rely on external sensors to control individual devices [3, 4]. However, such centralized sensing approaches are hampered in indoor environments by obstructions (walls, furniture, etc.). At the same time, reliance on sensing infrastructure implies requirement for a large deployment of support sensors covering all the locations that a MAV may visit [5]. Thus these approaches are only applicable in pre-surveyed locations.

This paper presents DrunkWalk, a technique for cooperative and adaptive navigation of swarms of micro-aerial sensors in environments not formerly preconditioned for operation. The key focus behind this networked MAV swarm research is to rely on collaboration to overcome limitations of individual nodes and efficiently achieve system-wide sensing objectives.

In DrunkWalk, the MAV swarm self-establishes a temporary infrastructure of a few landed MAV's acting as radio beacons. Using radio signature or fingerprints 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 according to the certainty of their location estimations – directing movement to improve location estimates when certainty is low, and directing MAV to the goal location when certainty of location estimates is high. The adaptive strategy enables DrunkWalk to improve the location estimation accuracy and success rate of navigation under given time and accuracy constraints.

The main contributions of this paper are:

- An adaptive planning algorithm for navigation that enables the swarm to collaboratively achieve up to 6× reduction in location estimation errors, and as much as 3× improvement in navigation success rate under the given time and accuracy constraints.
- A planning algorithm that determines the quality of location estimations and uses it to adaptively plan node motion.
- 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 validating our assumptions.

The rest of this paper 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 paper. Section 4 evaluates and analyzes the system through extensive simulations and validates assumptions through MAV testbed experiments. Further discussion of some extra system details are stated in Section 5. 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. OVERVIEW

Potential MAV swarm sensing applications will require mobile sensors to autonomously navigate to desired locations in operating environments with no localization infrastructure. In this paper, we address the problem of how a network of mobile sensors can be navigated to pre-determined positions under time and accuracy constraints.

2.1 Operation & Architecture

The system begins operation with a swarm of MAV's being introduced into the operating environments. We make the assumption that a coarse map of the building is available and can be utilized by domain experts to pre-determine suitable placement of sensors. 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).

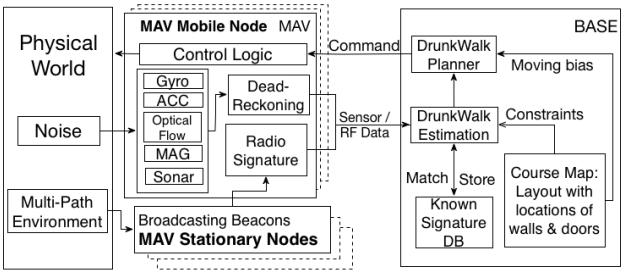

Figure 1: The figure shows the architecture of our navigation system. The mobile MAV nodes send dead-reckoning sensor data and radio signatures to a base station. The base runs the DrunkWalk estimation and planning algorithm and issues movement commands to individual MAV nodes.

The proposed system has 3 major operational phases: setup, estimation and planning (the latter two proceed in conjunction):

- **Setup:** 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 objective of the stationary nodes is to enable mobile MAV nodes to obtain radio signatures or fingerprints of locations traversed on their paths. These nodes use a simple dispersion algorithm [6, 7] that lets them spread out in the environment *without any estimation of their location.*
- **Estimation:** The system then desires to estimate the locations of nodes in order to guide them to their goal locations. To realize this, the system first uses dead reckoning sensors such as an optical flow velocity sensor and magnetometer (in our test MAV platform) to get a rough estimate 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 improve location estimations of the entire swarm.
- **Planning:** Having estimated locations, the system plans paths for each node that 1) leads to subsequent goal positions and 2) improves location estimation accuracy. 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 don't know where they are, they cannot plan a correct path to their destination.*

Our system thus considers the quality of location estimation in planning node paths. The path planner commands nodes movement such that they increase the number of snapshot points and potentially improve location estimates when the quality of their estimates is likely to be low. On the other hand, when the location estimates are likely to be more accurate, the planner uses the map to direct them to their designated goal locations.

Figure 1 shows the architecture of the system. Through dispersion algorithm, the system deploys **Stationary MAV Nodes** that act as wireless beacons. **Mobile MAV Nodes** explore and obtain dead-reckoning measurements from their on-board sensors and radio RF-signatures from the stationary beacons. The mobile nodes relay this to a **Base**. The Base stores a database of known radio signatures (**Signature DB**) that is used to determine snapshot point in node paths and apply corrections to their dead-reckoning estimates. The corrected location estimates are used by the Base in conjunction with a coarse map (indoor layout with location of walls and doors) of the environment to command the subsequent movements of MAV nodes.

2.2 Improving Location Estimation Through Swarms

The core idea behind our estimation approach is to use relatively large number of mobile sensors 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 due to noise in inertial sensors that are independent across nodes and time [8]. 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 point from radio measurements.

2.2.1 *Determining Snapshot Points*

The location estimation requires a node to be able to determine when it visits a location previously visited by itself or by another 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), 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, it is added to the database as a new entry.

2.2.2 *Combining Estimates at Snapshot Points*

The process of combining location estimations 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.

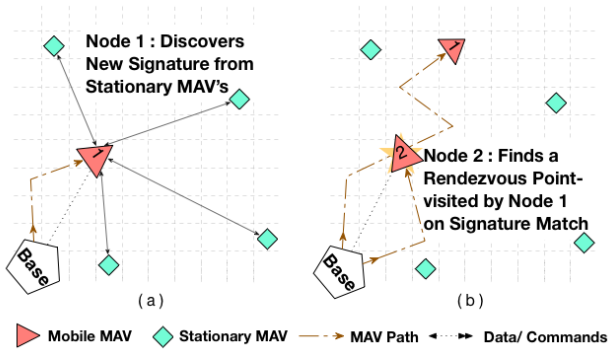

Figure 2: The figure shows the process of determining snapshot points. (a) Node 1 moves and obtains a radio signature from stationary MAV. This is entered into the the Base Signature DB as new signature. (b) When Node 2 visits the same location, its collected radio signature matches existing signature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

Combining estimations 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 [9] 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 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

We described how a snapshot point between the paths of nodes can be utilized to improve location estimates. Planning paths is thus 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.

2.3.1 *DrunkWalk*

In order to reach the goal regions, we use an indoor layout with location of walls and doors of the environment. It

should be noted that the algorithm does not require high quality maps with information of the position of obstacles. Such rough maps are generally available or easy to obtain in most application scenarios.

The rough map enables us to bias the direction of node movement towards 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 MAV node attempts to seek snapshot points that can potentially improve the location estimates of the MAV node. This is executed when the quality of location estimations (determined by the entropy of the tracking particle filter distribution) is low.
- **Navigation:** In this mode, the MAV node attempts to follow the direction of the bias from the graph using the estimated location from the DrunkWalk algorithm. This is executed when the quality of location estimates 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 increases the time of navigation. Therefore, the DrunkWalk algorithm seeks to optimize this trade-off by adaptively switching between these two modes.

3. DESCRIPTION

This section provides a detailed description of the major components of our proposed system. First, this section describes how the location and orientation of the MAVs and the positions of the signatures are estimated over time using a set of particle filters. 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, a base station 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.

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 fifties [10] and became popular in robotics in the last two decades [9].

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

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

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

$$p(z_t|x_t) \tag{2}$$

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 Eq. 1 and Eq. 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 [9]. 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 ith particle in $\mathcal{X}_t$, $x_t^{[i]}$, represents the ith 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.

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 $\quad x_t^{[i]} \leftarrow$ sample $\sim p(x_t|x_{t-1}^{[i]}, u_t)$;
5 $\quad w_t^{[i]} \leftarrow p(z_t|x_t^{[i]})$;
6 $\quad \mathcal{X} \leftarrow \mathcal{X} \cup \{< x_t^{[i]}, w_t^{[i]} >\}$;
7 **end**
8 **for** $i \leftarrow 1$ **to** M **do**
9 $\quad$ draw j with probability $\alpha\ w_t^{[j]}$;
10 $\quad \mathcal{X}_t \leftarrow \mathcal{X}_t \cup \{x_t^{[j]}\}$;
11 **end**

Algorithm 1: Generic particle filter algorithm

3.2 MAV Location Tracking

In this subsection, we show how the generic particle filter estimator can be specialized to estimate the location of the MAVs. 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.

3.2.1 Prediction from Motion Models

For the prediction step, it is necessary to use a generative law to implement the particle creation in line 4 of Algorithm 1. To this end, we use equations similar to the ones given in [7]. 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 form 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:

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

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

where v_t and ω_t are the nominal commands. In our simulations, according to [7], $p(n_v)$ 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 hinges 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 [11], 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.

The process is substantially different for RF-signature snapshot points. 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, 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 created; see section 3.3 for details.) 2) On the contrary, if the database determines that a signature snapshot points is taking place, the second step starts. First, on the database side, the particle filter estimating the position of the signature being revisited is updated (see section 3.3 for details.) After the RF-signature particle filter has been updated, each particle in the MAV particle filter is assigned a weight as follows. A GMM is created starting from the particles in the signature being matched. Such GMM is a bidimensional probability density function with the following equation:

$$f_{GMM}(x,y) = \frac{1}{M}\sum_{i=1}^{M} f_{\mathcal{N},\Sigma}^i(x,y) \quad (7)$$

where $f_{\mathcal{N},\Sigma}^i$ 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

$$w_t^{[i]} \leftarrow f_{GMM}(c_{x_t}^{[i]}, c_{y_t}^{[i]}). \quad (8)$$

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

3.2.3 Adding Particles Using Coarse Map

Due to the unavoidable errors in the estimation process, we implemented an additional step to counter the formerly mentioned particle depletion problem. After the new set $\mathcal{X}_t$ 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 Particle Filter for Snapshot Points

We now describe how the spatial location of the signatures can be estimated using a set of particle filters. For the MAVs case, we do not compute a centralized estimation, but we rather associate a filter with each signature to be tracked. This estimation process has two main differences with the position and orientation estimation for the MAVs. First, the number of signature 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, signatures 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 signature is generated when the known signatures database receives a query from one of the MAVs with an 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 particles filter is instantiated. The initial set of particles for this new filter is copied from the particles of the vehicle 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

Correction happens when a MAV queries signature database with a signature that can be matched with one of the entries already discovered. In this case, Algorithm 1 is executed for the signature filter, with the exception of line 4, because no prediction takes place. The weight for $w_t^{[i]}$ for the ith 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 $\overline{x}$ be the computed average position of the MAV generating the match, and let $s_{t-1}^{[i]}$ be the position of the ith particle in the signature particle filter at time $t-1$, and let $d_i = \left\| \bar{x} - s_{t-1}^{[i]} \right\|_2$ 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

$$w_t^{[i]} = F_{d,\delta}(d+K) - F_{d,\delta}(d-K) \tag{9}$$

where $F_{d,\delta}$ is the cumulative density function of a Gaussian distribution with mean d and variance δ. 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. Once weights have been computed, correction for the estimate of the signature particle filter can then take place through resampling, as described in Algorithm 1.

3.3.3 Database of Fingerprints

In order to help location estimation correction with snapshot points after each movement, the system maintains a database of fingerprints. A fingerprint 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 fingerprint and the fingerprints stored in the database. A pre-defined threshold T_{sig} is used to decide whether it is a new or known fingerprint.

3.4 DrunkWalk Planning

In this section, we describe how the system plans the paths of MAV nodes with location estimates of varying quality in order to deploy quickly. Figure 3 shows a flowchart of the planning algorithm.

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.

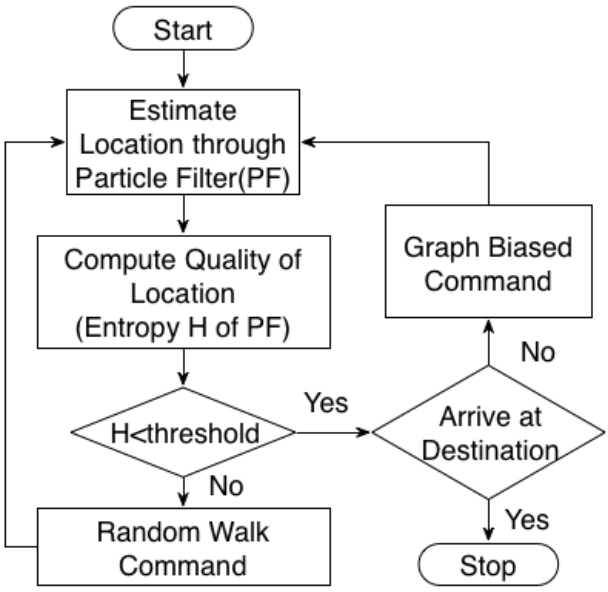

Figure 3: The figure shows the flowchart of the DrunkWalk planning algorithm. The planner adaptively changes between random walk and graph biased movement based on the entropy of particle filters tracking respective MAV nodes.

The coarse map makes very few assumptions about the quality of the map but provides a way to bias the motion of MAV nodes towards designated locations.

3.4.2 Entropy as Quality of Location Estimates

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

$$H(x) = E[-\log_2 p(x)] \tag{10}$$

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 [9] of the weights at time t as

$$H_t = -\sum_{i=1}^{M} w_t^{[i]} \log_2 w_t^{[i]}. \tag{11}$$

3.4.3 Exploration

When the entropy of the particle filter is high (> threshold T_H), the system seeks to primarily improve the location estimates. The intuition here is that with an incorrect estimation of current location, using the bias from the graph is likely to be incorrect. This also results in cases where the

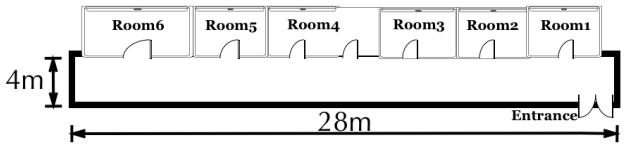

Figure 4: The figure shows the floor plan of 6 rooms with a hallway used for physical feature based simulation and real testbed experiments. The MAVs start from the entrance of the building and are navigated to different goal areas.

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

Correspondingly, when the uncertainty of location estimates is low ($H_t < T_H$), 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 1 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 [1] rely on their low weight and often use collisions themselves to discover obstacles. However, a strategy is needed in dealing with collision so as 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 planning MAVs paths through physical feature based simulation and real experiments on a MAV testbed. Both simulation and real experiments are conducted in a building with

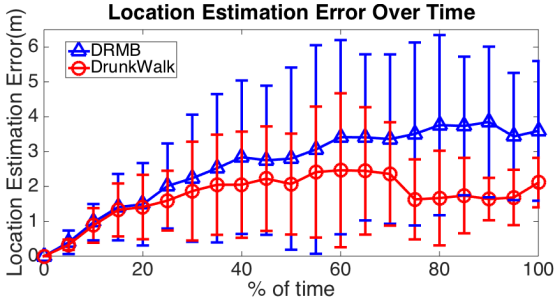

Figure 5: The figure shows the average and standard deviation of location estimation errors at different flying duration heading for the near destination using DrunkWalk and DRMB from 5 experiments. Drunkwalk achieves around 2m location estimation errors on average and 1-1.5m standard deviation.

multiple rooms connected with a hallway as shown in Figure 4. The MAVs start from the entrance and are navigated to different goal areas (rooms).

The evaluation focuses on the following aspects:

- Characterizing the performance of the system in terms of 1) navigation duration and 2) average accuracy of location estimations in comparison to existing navigation approach.
- Testing the robustness of the system with changing parameters, such as number of stationary MAV nodes, noise of sensors, and radio fingerprint accuracy.
- Validating the assumptions of the simulation experiments through real MAV testbed experiments.

For both testbed experiments and simulation, we compare our DrunkWalk algorithm to another online navigation strategy that does not require any location infrastructure. 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 [12]. 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 DrunkWalk.

4.1 Testbed Experiment Setup

To validate our system in a realistic setting, we implement our algorithm on a server and the SensorFly [1] [13] MAV testbed. The SensorFly platform used in our test has an 8-bit 16Mhz AVR AtMega128rfa1 micro-controller, a 3-axis accelerometer, a 3-axis gyroscope, a 2-axis optical flow velocity sensor, a 3-axis magnetometers, a ultrasonic altitude sensor and a XBee radio [14]. 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.

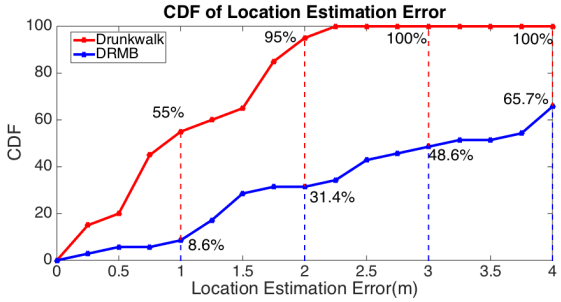

Figure 6: The figure shows the cumulative distribution function (CDF) of location estimation errors to arrive at near destination using both DrunkWalk and DRMB from a typical run.

In our setup, we manually fly 8 MAVs on a $4m \times 28m$ arena shown in Figure 4. 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

We utilize the MAV testbed to illustrate the location estimation errors from short to long distances to compare performances and robustness of DrunkWalk with that of DRMB.

Figure 5 compares the performance and robustness of DrunkWalk and DRMB at different phases of navigation duration. We plot the average and standard deviation of location estimation errors from 5 experiments at different % 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, DrunkWalk performs similarly as DRMB due to lack of snapshot points to correct location estimation errors. After this initial period, DrunkWalk's snapshot point correction maintained location estimation errors within the range of 1.5m to 2m. In contrast, the error of DRMB kept accumulating to larger than 3m. This is because multiple measurements at snapshot points can correct the location estimation errors in DrunkWalk. In addition, after 30% of the navigation time, the standard deviation of DrunkWalk location estimation is also $30\% - 60\%$ smaller than DRMB approach. This shows DrunkWalk is more reliable than DRMB.

Figure 6 shows cumulative distribution function (CDF) of location estimation errors using DrunkWalk and DRMB from a typical experiment. In DrunkWalk, corrections from snapshot points help keep errors within 2.5 meters. In comparison, location estimation error of DRMB remains unbounded. More than 50% of the time, DRMB has more than 3 meters error, compared to less than 1 meter error for DrunkWalk.

To illustrate the performance of DrunkWalk and DRMB for medium and far destinations, we use Figures 7 and 8 to show single run experiment results. We further evaluate them through simulated results in section 4.3. Note that

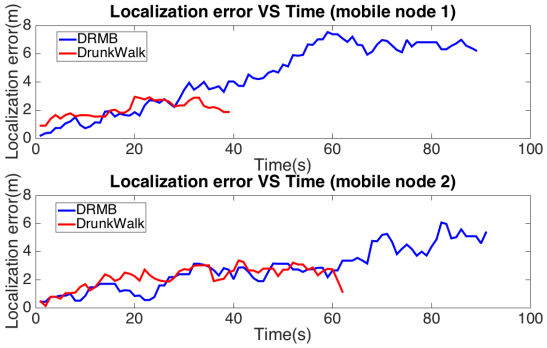

Figure 7: The figure shows the location estimation error over time using DrunkWalk and DRMB to the medium destinations. Mobile nodes with DrunkWalk arrive at the destination earlier than those with DRMB due to their capability to limit the location estimation errors.

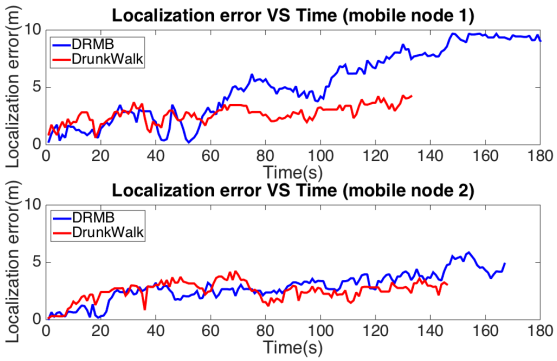

Figure 8: The figure shows the location estimation error over time using DrunkWalk and DRMB to the far destinations. It is noted that we do not plot all the data for mobile node 1 since it fails to arrive at the destination before the battery was exhausted (600 seconds).

besides the node using DRMB in the upper plot in Figure 8, 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, DrunkWalk has similar location estimation errors with DRMB. After this initial period, adequate snapshot points help DrunkWalk to limit the location estimation errors while the error of DRMB keeps accumulating. This shows that multiple measurements at snapshot points in DrunkWalk do help limit location estimation errors. The higher location estimation accuracy from 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. On the other hand, both mobile node 1 and 2 with DrunkWalk arrive at the medium and far destination within similar durations. This shows the stability of DrunkWalk with the help of snapshot points.

4.3 Physical Feature Based Simulation Environment

We implemented a MAV simulation environment [15] 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.

For our evaluations, we configure the simulator as follows:

- **Arena** – We use a multi-room indoor scenario shown in Figure 4, 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 4.
- **Node Sensors** – The sensor nodes in the simulation are modeled after the SensorFly [1] 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 $0.8m$ to $1.2m$ in one simulation tick.
- **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.

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 run the simulation for a time period of 600 seconds ($\tilde{1}0$ minutes) corresponding to the typical battery life of current generation MAV nodes.

4.4 System Performance

This section evaluates our system performance under different destination constraints and time limitations. A successful navigation is achieved when the node can be navigated to the destination within the given accuracy and time limitation. For example, if the destination coordinate is (4 meters, 5 meters), the required accuracy is 1 meter and time limit is 90 seconds, a successful navigation means that the mobile node can arrive within the range of 1 meter from (4 meters, 5 meters) within 90 seconds.

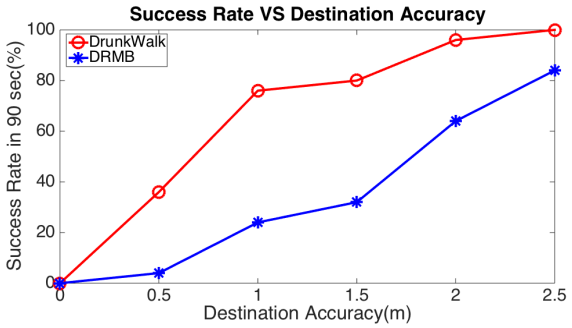

Figure 9: The figure shows the navigation success rate under different destination accuracy constraints within 90 seconds for DrunkWalk and DRMB.

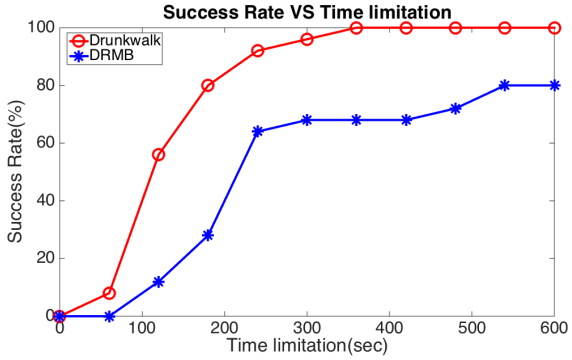

Figure 10: The figure shows the success rate as a function of time limitation under 0.5m destination accuracy constraint, using DrunkWalk and DRMB alone.

Figure 9 shows the navigation success rate as a function of destination accuracy constraints within 90 seconds using DrunkWalk and DRMB. When the destination accuracy is strict (0.5 meter), DrunkWalk achieves 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 DrunkWalk can still work with the help of snapshot points. For less constrained destination accuracy (1 meter to 2 meters), DrunkWalk shows consistently 30% to 40% higher success rate than DRMB. Futhermore, DrunkWalk achieves 100% success rate for even looser destination accuracy constraint (2.5m) while DRMB can also get 84% success rate. This is expected since above 2.5 meter range is more than one half of the hallway width (4.0 meters), where even with high location estimation errors, the mobile node can still arrive at the destinations simply by following the walls.

Figure 10 plots the success rate as a function of time limitation for both DrunkWalk and DRMB under a 0.5m destination accuracy constraint. Under a strict time limitation constraint (60 seconds), even DrunkWalk only achieved 8% success rate since it is not able to get enough snapshot points to get accurate location estimation on the way to the destination. When the time limitations are released (120 seconds to 300 seconds), DrunkWalk shows 30% to 50% higher success rate than DRMB. This is because DrunkWalk has enough time to get snapshot points to correct the location

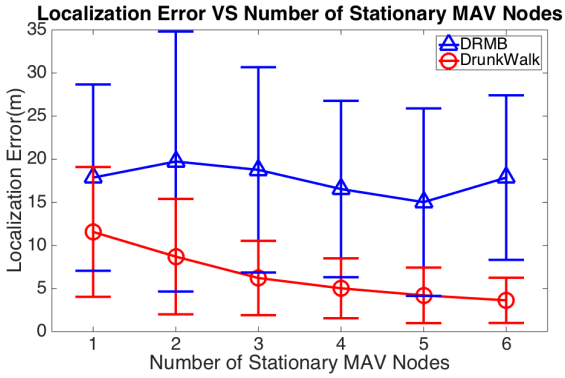

Figure 11: The figure shows the location estimation error with varying number of stationary MAV nodes using DrunkWalk and DRMB. DrunkWalk has an obvious decreasing trend when the number of stationary MAV nodes increases. It is noted that DRMB does not use stationary MAV and its performance variance in the figure is due to noise from sensors.

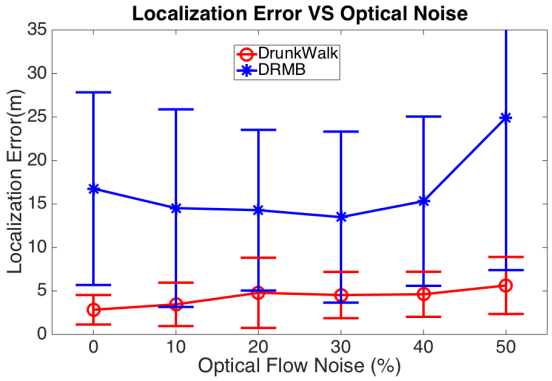

Figure 12: The figure shows location estimation error as a function of optical flow noise using DrunkWalk estimation and DRMB alone. The noise per sensor is modeled as a normal distribution with varying standard deviation. The plot shows that DrunkWalk is able to correct the DRMB error and maintain low standard deviation.

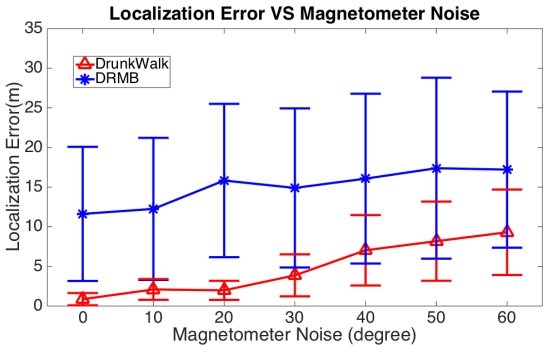

Figure 13: The figure shows location estimation error as a function of magnetometer noise using DrunkWalk estimation and dead reckoning alone. The noise per sensor is modeled as a normal distribution with varying standard deviation. The plot shows that DrunkWalk is able to correct the DRMB error and maintain low standard deviation.

estimation errors, while the error of DRMB keeps accumulating. After 300 seconds, 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, DrunkWalk still gets more than 20% higher success rate than DRMB.

4.5 System Robustness

In this section, we evaluate the robustness of our system by examining the location estimation errors under different system setups of both DrunkWalk and DRMB.

4.5.1 Number of Stationary MAV Nodes

Figure 11 shows the location estimation errors for different numbers of stationary MAV nodes, where Drunkwalk achieves 1.5× to 6× reduction for average and 1.5× to 4× reduction for standard deviation compared to DRMB. This shows its better capability and reliability to limit location estimation error. This can be attributed to improvement on matching snapshot with larger number of stationary nodes broadcasting beacons. We can also see the decreasing trend for both average and standard deviation in Drunkwalk when the number of stationary nodes increase, which means that increasing the number of stationary nodes do help enhance performance.

4.5.2 Navigational Sensor Noise

The noise in motion measurements due to Dead-Reckoning 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 [7], [11]. For the optical flow velocity sensor, a noise corresponding to a normal distribution with 0 mean and standard deviation of 20% of the measured velocity value was added. For the magnetometer, a 30° noise corresponds to a normal distribution with 0° mean and standard deviation of 30°. The resultant noise in DRMB location can be computed as per the motion update equation [7], [11]. We apply sensor noise to both DRMB and DrunkWalk estimates.

Figure 12 and 13 show the location estimation error in DRMB and DrunkWalk for 10 nodes at various sensor noise levels. In Figure 12, with optical flow noise level increasing from 0 to 50%, the average location estimation error of DrunkWalk increases from 2.5m - 5.5m while that of DRMB goes up very quickly to around 25 meters. The error increase is limited due to the corrections from the multiple measurements at the same snapshot points. In Figure 13, a faster increasing trend (from 0.5 to 7.5 meters) is observed with the magnetometer noise increase. In addition, similar increasing trends are observed for both DrunkWalk and DRMB. This indicates that high noise from magnetometer weakens the correction capability on location estimation errors. However, during

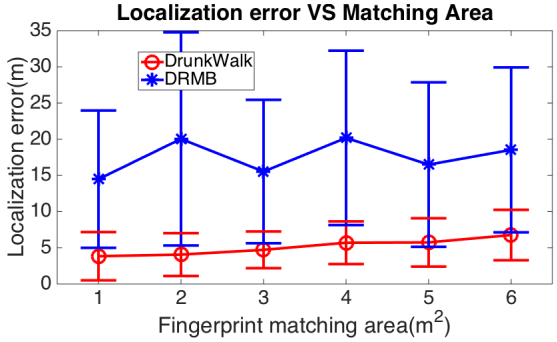

Figure 14: The figure shows location estimation error as a function of signature matching area for DrunkWalk and DRMB. To be noticed, DRMB does not work with stationary MAV and noise from sensors cause the performance variance.

operation, raw magnetometer noise can be mitigated by gyroscope data.

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 DrunkWalk depends on the resolution of the fingerprints matching, i.e., the area or distance within which two radio fingerprints can be reliably classified as being at the same location. If the criteria is too loose, incorrect points may be matched together, which leads to few snapshot points.

Figure 14 shows location estimation error as a function of signature matching area for DrunkWalk and DRMB for 10 nodes. We observe that DrunkWalk offers an improvement of over 3× compared to DRMB even for a poor matching resolution of $6m^2$. In 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 fingerprint accuracies of around $1m^2$, DrunkWalk provides a much larger reduction in error.

5. DISCUSSION

Several aspects of DrunkWalk planning algorithm warrant further discussion and could probably lead to extensions to DrunkWalk.

5.1 Parameter Tuning

We now describe how some key parameters affect the system performance and give some guidance on tuning them.

We model all the sensor and noise distributions according to real tests either by ourselves or from related work [7], [11]. After we got these values, we use them in our simulation platform and the real system.

The signature matching resolution is affected by four parameters: number of stationary nodes, K and δ in Equation 9, and T_{sig} in section 3.3.3. Just as trilateration [16], we need at least 3 stationary nodes to estimate x and y location. However, due to the noise from radios, we need more stationary beacons. In our experiments, we found 6 seems to give good results. In order to tune K and δ in Equation 9 and the threshold T_{sig} for matching signatures, we refer to our previous work [7]. After we decide the value of K, we should set up the sample rate for snapshot points higher than K/2 per sample according to the Nyquist-Shannon sampling theorem [17] to ensure recover the radio map.

Another parameter, T_H in section 3.4.3 affects the adaptive planning strategy. The range of entropy is from 0 to log(N), where N is the number of particles. How we set T_H depends on how we hope to bias our planning strategy. If we set T_H close to 0, that means we bias more to random walk. If we set the T_H close to log(N), that means we prefer to trusting the location estimation more and bias more to follow the coarse map.

Moreover, the reader can refer to [18] for guidance to set up the number of particles . In our system, we adopt 100 particles for both motion tracking and snapshot points.

5.2 Different Layout Geometry

DrunkWalk can deal with more complicated layout geometry and should achieve similar performance, although it is only tested and simulated in a layout shown in Figure 4. The multiple rooms connected with a hallway in Figure 4 can be decomposed into basic unit for complicated layout geometry. For a different layout geometry, we can regard it as a concatenation of these basic units and set up a series of intermediate destinations to help the drones arrive at the final destinations. E.g. for those multiple interconnected rooms with no hall layout, it is similar to the situation with only room 1, room 2 and the hallway in Figure 4. In this case, we regard the hallway as a loose room interconnected with room 1 and room 2. In addition, by sensing the sudden change in altitude and using a floor altitude look-up table, DrunkWalk could also deal with multiple floor conditions.

5.3 Message Overhead

In our system, message overhead is composed of three parts: 1) Stationary drones broadcasting messages to flying drones for RSSI measurements takes 1 bytes for each message. Each stationary drones broadcast messages 10 times per second. 2) Flying drones report the RSSI values, heading information and some other status information to the server every second. The corresponding message size is 20 bytes for each flying drone. 3) The link from server sending different command messages to different flying drones every second. The overhead of the command message takes 14 bytes. In real systems, due to the collision, the real overhead is often higher than the above value.

6. RELATED WORK

Works related to DrunkWalk maily fall into three domains: sensor networks, robotics and mobile computing.

The sensor network domain has a number of works on deploying and navigating mobile sensors [19, 20, 21, 22]. Howard et al. [23, 24] 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. [25] 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 paper 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 [9, 26, 27, 28]. 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 have been copious research in SLAM using either methods based on Kalman filters [29, 30, 31, 32] or particle filters [33, 34, 35, 36]. Both approaches, however, have been mostly applied to solve 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, etc. Methods based on Kalman filters are not applicable for the scenario we consider because we are dealing with multimodal, non-parametric probability distributions. Therefore, we opt for a solution based on particle filters.

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* [37, 38, 39, 40]. A recent paper by Twigg et al. [41] 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 [42, 43, 44, 45], or organic landmark maps [46, 47]. 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. [7] 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 the best of our knowledge, this paper presents 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.

7. CONCLUSION

This paper presents a system for collaborative and adaptive planning of resource-constrained MAV sensing swarms to quickly and efficiently navigate to preassigned locations. The system uses collaboration between nodes of the swarm to overcome the sensing and computational limitations of MAV nodes, and the challenging operating environments. We comprehensively evaluate the system through large-scale simulations and real MAV testbed experiments showing that DrunkWalk achieves up to 6× reduction in location estimation errors, and as much as 3× improvement in navigation success rate under the given time and accuracy constraints.

8. ACKNOWLEDGEMENTS

We would like to thank our shepherd Prof. Akos Ledeczi and the anonymous reviewers for their insightful and constructive comments. This research was supported by Intel, Nokia, NSF under the grant CNS1135874 and CNS1149611, and DARPA under the grant D11AP00265. The 4th author Stefano Carpin is partially supported by the Army Research Lab under contract MAST-CNC-15-4-4. Any opinions, findings, and conclusions or recommendations expressed in these materials are those of the author and should not be interpreted as representing the official policies, either expressly or implied, of the funding agencies of the U.S. Government.

9. REFERENCES

[1] A Purohit, Zheng Sun, F Mokaya, and Pei Zhang. SensorFly: Controlled-mobile sensing platform for indoor emergency response applications. In *Proceeding of the 10th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 223–234, 2011.

[2] Robert J. Wood. The First Takeoff of a Biologically Inspired At-Scale Robotic Insect. *IEEE transactions on robotics*, 24(2):341–347, 2008.

[3] Karthik Dantu, Bryan Kate, Jason Waterman, Peter Bailis, and Matt Welsh. Programming micro-aerial vehicle swarms with karma. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 121–134, New York, NY, USA, 2011. ACM.

[4] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, 33(1-2):143–156, February 2012.

[5] S. Shen, N. Michael, and V. Kumar. Vision-based autonomous navigation in complex environments with a quadrotor. In *iros*, Tokyo, Japan, nov 2013.

[6] Ryan Morlok and Maria Gini. Dispersing robots in an unknown environment. In *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2004.

[7] Aveek Purohit, Zheng Sun, and Pei Zhang. Sugarmap: Location-less coverage for micro-aerial sensing swarms. In *Proceedings of the 12th International Conference on*

Information Processing in Sensor Networks, IPSN '13, pages 253–264, New York, NY, USA, 2013. ACM.

[8] I. Constandache, R.R. Choudhury, and I. Rhee. Towards mobile phone localization without war-driving. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[9] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2006.

[10] A. Doucet, J.F.G. de Freitas, and N.J. Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer-Verlag, 2001.

[11] Zheng Sun, Aveek Purohit, Shijia Pan, Frank Mokaya, Raja Bose, and Pei Zhang. Polaris: Getting accurate indoor orientations for mobile devices using ubiquitous visual patterns on ceilings. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 14:1–14:6, New York, NY, USA, 2012. ACM.

[12] Johann Borenstein, Liqiang Feng, and H. R. Everett. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Natick, MA, USA, 1996.

[13] Aveek Purohit, Pei Zhang, Brian M Sadler, and Stefano Carpin. Deployment of swarms of micro-aerial vehicles: from theory to practice. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5408–5413. IEEE, 2014.

[14] Digi International. XBee Multipoint RF Modules Datasheet, 2009.

[15] A Purohit and Pei Zhang. Controlled-mobile sensing simulator for indoor fire monitoring. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1124–1129, 2011.

[16] Dimitris E Manolakis. Efficient solution and performance analysis of 3-d position estimation by trilateration. *Aerospace and Electronic Systems, IEEE Transactions on*, 32(4):1239–1248, 1996.

[17] Abdul J Jerri. The shannon sampling theorem: Its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, 1977.

[18] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

[19] Maxim A Batalin and Gaurav S Sukhatme. Coverage, exploration, and deployment by a mobile robot and communication network. In *Information Processing in Sensor Networks*, pages 376–391. Springer, 2003.

[20] Maxim A Batalin and Gaurav S Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems*, 26(2-4):181–196, 2004.

[21] Ting Liu, Christopher M Sadler, Pei Zhang, and Margaret Martonosi. 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*, pages 256–269. ACM, 2004.

[22] Yong Wang, Pei Zhang, Ting Liu, Chris Sadler, and Margaret Martonosi. Movement data traces from princeton zebranet deployments. *CRAWDAD Database*, 129, 2007.

[23] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. An Incremental Self-Deployment Algorithm for Mobile Sensor Networks. *Autonomous Robots*, 13(2):113–126, September 2002.

[24] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

[25] Gaurav S. Sukhatme Maxim A. Batalin. Spreading Out: A Local Approach to Multi-robot Coverage. *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics System*, 2002.

[26] Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti, Bastian Steder, Rainer Kummerle, Christian Dornhege, Michael Ruhnke, Alexander Kleiner, and Juan D Tardós. A comparison of slam algorithms based on a graph of relations. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2089–2095. IEEE, 2009.

[27] Denis Chekhlov, Mark Pupilli, Walterio Mayol-Cuevas, and Andrew Calway. Real-time and robust monocular slam using predictive multi-resolution descriptors. In *Advances in Visual Computing*, pages 276–285. Springer, 2006.

[28] Austin I Eliazar and Ronald Parr. Dp-slam 2.0. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1314–1320. IEEE, 2004.

[29] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568. IEEE, 2006.

[30] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions of Robotics and Automation*, 17(3):229–241, 2001.

[31] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 473–479. IEEE, 2008.

[32] Shoudong Huang and Gamini Dissanayake. Convergence and consistency analysis for extended kalman filter based slam. *Robotics, IEEE Transactions on*, 23(5):1036–1049, 2007.

[33] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437. IEEE, 2005.

[34] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):36–46, 2007.

[35] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi. Fast and accurate slam with rao–blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38, 2007.

[36] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[37] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE Int. Conference on Computer Communications*, pages 775–784, 2000.

[38] A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki. On the feasibility of using wireless ethernet for indoor localization. *IEEE Transactions on Robotics and Automation*, 20(3):555–559, 2004.

[39] Moustafa Youssef and Ashok Agrawala. The horus wlan location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218. ACM, 2005.

[40] S. Zickler and M. Veloso. RSS-based relative localization and tethering for moving robots in unknown environments. In *IEEE Int. Conference on Robotics and Automation*, pages 5466–5471, 2010.

[41] J. Twigg, J. Fink, P.L. Yu, and B.M. Sadler. Efficient base station connectivity area discovery. *International Journal of Robotics Research*, 2013.

[42] Brian Ferris, Dieter Fox, and Neil Lawrence. Wifi-slam using gaussian process latent variable models. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2480–2485, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[43] Mikkel Baun Kjærgaard. A taxonomy for radio location fingerprinting. In *Location-and Context-Awareness*, pages 139–156. Springer, 2007.

[44] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. Walkie-markie: Indoor pathway mapping made easy. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, pages 85–98, Berkeley, CA, USA, 2013. USENIX Association.

[45] Jie Yin, Qiang Yang, and Lionel M Ni. Learning adaptive temporal radio maps for signal-strength-based location estimation. *Mobile Computing, IEEE Transactions on*, 7(7):869–883, 2008.

[46] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 197–210, New York, NY, USA, 2012. ACM.

[47] Aveek Purohit, Zheng Sun, Shijia Pan, and Pei Zhang. Sugartrail: Indoor navigation in retail environments without surveys and maps. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*, pages 300–308. IEEE, 2013.