SugarMap: Location-less Coverage for Micro-Aerial Sensing Swarms

Aveek Purohit
Carnegie Mellon University
Pittsburgh, PA, USA
apurohit@ece.cmu.edu

Zheng Sun
Carnegie Mellon University
Pittsburgh, PA, USA
zhengs@ece.cmu.edu

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

ABSTRACT

Micro-aerial vehicle (MAV) swarms are emerging as a new class of mobile sensor networks with many potential applications such as urban surveillance, disaster response, radiation monitoring, etc., where the swarm is tasked with collaboratively covering a hazardous unknown environment. However, efficient collaborative coverage is challenging due to limited individual sensing, computing and communication resources of MAV sensor nodes, and lack of location infrastructure in the unknown application environment.

We present SugarMap, a novel system that enables such resource-constrained MAV nodes to achieve efficient sensing coverage. The self-establishing system uses approximate motion models of mobile nodes in conjunction with radio signatures from self-deployed stationary anchor nodes to create a common coverage map. Consequently, the system coordinates node movements to reduce sensing overlap and increase the speed and efficiency of coverage. The system uses particle filters to account for uncertainty in sensors and actuation of MAV nodes, and incorporates redundancy to guarantee coverage. Through large-scale simulations and a real implementation on the SensorFly MAV sensing platform, we show that SugarMap provides better coverage than the existing coverage approaches for MAV swarms.

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

1. INTRODUCTION

Micro-aerial vehicle (MAV) swarms are an emerging class of mobile sensor networking systems with many potential applications such as urban surveillance, disaster response and crop pollination. These swarms comprise of miniature aerial sensor nodes capable of autonomous movement but with limited sensing, computing and communication resources on each node [25, 19]. In a majority of the proposed applications, such as surveillance or survivor search, the sensor network is tasked with moving and covering a target space. These networks must rely on collaboration to quickly and efficiently achieve their system-wide sensing objectives despite the limitations of individual nodes.

The collaborative swarm approach provides the advantage of greater resilience, adaptability and speed of sensing as compared to a monolithic robot. However, the limited individual capabilities of MAV's present many unique challenges in enabling collaboration between a swarm of nodes. A fundamental primitive for collaborative algorithms, especially for spatial sensing coverage, is a common frame of reference for location. Indeed, the knowledge of the relative locations of sensor nodes is assumed or computed in most prior work on multi-robot spatial coverage [7, 9, 11, 14, 22, 24]. However, these existing techniques are unsuitable for MAV swarms due to the following challenges:

- **Lack of Infrastructure** – Many multi-robot systems rely on existing infrastructure for inferring the relative location of nodes such as GPS, Wi-Fi access points, motion-tracking camera systems etc. This infrastructure is unavailable in many of the intended operating environments for MAV swarms such as indoor buildings (GPS denied) or disaster scenarios (infrastructure denied).
- **Limited Sensors** – Robots routinely use range sensors such as LIDARs, laser-range finders or multiple ultrasonic rangers to compute relative positions [2, 5, 6, 16]. MAV platforms have severe weight constraints and cannot accommodate most of these sensors that weigh in the hundreds of grams.
- **Limited Computing Power** – Many robotic systems employ computer vision with computation intensive feature detection algorithms to generate maps and compute their relative location. MAV swarms do not have the local processing ability to employ these algorithms.
- **Low-bandwidth Communication** – Robots have the ability to relay bandwidth intensive images and data to a base station for processing. The MAV swarms are limited in the available bandwidth and

connectivity due to the their low-power and low-range radio's.

In this paper, we present SugarMap – a system that enables resource-constrained MAV sensor swarms to collaboratively cover an area. SugarMap coordinates node movements to reduce the amount of sensing overlap between swarm nodes and increase the efficiency and speed of coverage. Most importantly, the system does not rely on external location infrastructure, bulky or sophisticated sensors, computation intensive algorithms or high-bandwidth communication.

The main contribution of this paper is threefold:

- a multi-node location-less coverage algorithm that uses self-dispersed anchor nodes to obtain radio RF-signatures and provides a common spatial frame of reference for MAV swarm nodes.
- a particle filter framework to estimate coverage taking into account the uncertainties and inaccuracies introduced by semi-controlled motion in MAV systems.
- a coverage algorithm that inherently provides redundancy as per the uncertainty of node motion.

In the SugarMap system, radio RF-signatures are obtained by *explorer* sensor nodes through querying a set of *anchor* nodes. The anchor nodes are MAV nodes that SugarMap deploys (lands) in the area, through a dispersion algorithm, at initialization. The system does not require a location information for deploying anchor nodes and is thus self-establishing. Consequently, SugarMap uses the radio measurements from the anchors as a common spatial frame of reference to coordinate node motion and collaboratively cover an unknown area. The algorithm uses particle filters to account for the actuation uncertainty of low-cost MAV nodes and uses redundancy in node paths to guarantee coverage with the desired degree of confidence.

We evaluate the performance of SugarMap through large-scale simulation and validate through a real-system implementation on the SensorFly [19] MAV sensor platform. We compare the performance of SugarMap to the state-of-the-art in existing online coverage algorithms for swarms. We show that SugarMap provides faster coverage than existing approaches in absence of location information.

The rest of the paper is organized as follows. Section 2 gives an overview of the SugarMap system. Section 3 describes the different components and algorithms of SugarMap in detail. Section 4 describes our implementation, experimental setup, results and provides comparisons with other approaches. Section 6 describes the state-of-the-art in multi-agent coverage algorithms in context of MAV swarms. Finally, Section 7 summarizes our conclusions.

2. SYSTEM OVERVIEW

This section gives an overview of the different aspects of the SugarMap system, including the capabilities of nodes and the typical operating scenario.

Figure 1 gives a flow diagram of data and commands between the major components of the SugarMap system. At system initialization, nodes are sent into the environment and landed in a constrained dispersion manner. The SugarMap system has a base station *brain* running the SugarMap algorithm. The base station gives commands to the mobile nodes (explorers) to move in a coordinated fashion. The explorers collect RF-signatures from stationary nodes (anchors) and relay it to the base station. The base station uses particle filters and RF-signatures obtained from the explorer nodes to update a global coverage map for the swarm. The global coverage map in turn helps the base station recursively plan the next movement of explorer nodes.

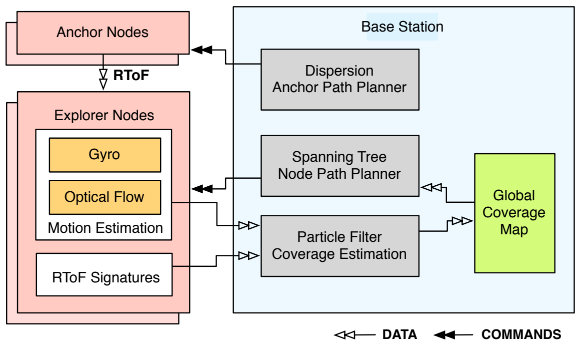


Figure 1: An overview of the SugarMap system.

2.1 MAV Sensor Nodes

We assume the MAV nodes to be weight-limited resource-constrained platforms. They have limited on-board computation capability and light-weight components such as MEMS-based inertial motion sensors, an altitude measurement sensor, a sensor to estimate velocity, and a radio for communication and RF-signature estimation.

For our prototype evaluation, we implement SugarMap on the SensorFly [19] MAV swarm platform. The SensorFly is a MAV platform with a 8-bit AVR AtMega128 microcontroller, inertial motion sensors – 3-axis accelerometer and 3-axis gyro, an ultrasonic ranger for altitude estimation, an optical flow sensor for velocity estimation, and a 802.15.4a compatible radio with Round-trip time-of-flight (RToF) measurement capability. The entire platform is under 30g in weight striking a careful balance between weight and sensing capability as is typical of MAV sensing platforms.

2.2 Operating Environment

The predominant application of proposed MAV swarms is in attaining sensing coverage of unknown environments that are inaccessible or hazardous for humans to enter. For example, in applications such as urban surveillance, survivor search after disasters, or nuclear radiation monitoring.

Figure 2 gives an illustration of the operating environment. The anchors are deployed by dispersion at the boundaries, explorers move and sense the area, and the base station received data from explorers and transmits commands. We make the following assumptions about the operating environment from the application scenario:

- The initial position and pose of swarm nodes is known. The swarm nodes are introduced into the operating environment manually. For example, a firefighter introduces nodes into an indoor structure damaged by an earthquake through an accessible opening.

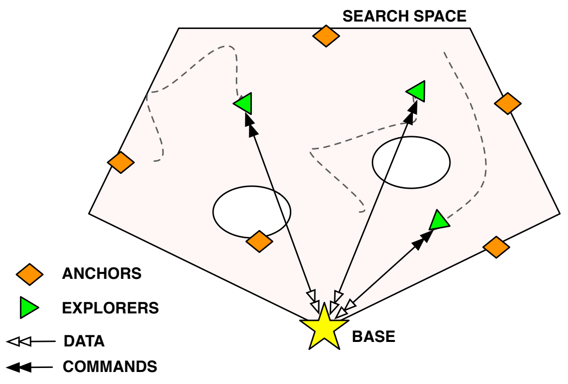


Figure 2: An illustration of the SugarMap system operating environment.

- The search space for a single group of swarm nodes is continuous. The swarm nodes do not have the capability to break through obstructing structures.

2.3 System Components

The SugarMap system has 3 major components:

- **Anchors** – Anchors are MAV nodes that the system lands at initialization in the environment. The nodes land using a simple dispersion algorithm where they seek to approximately spread out from each other but at the same time maintain radio connectivity with all other anchors. The dispersion does not use any location information but relies on the approximate proximity information provided by the radio time-of-flight measurements.

- **Explorers** – Explorers are MAV nodes that move and sense the environment. The nodes collect radio signatures from the anchor nodes, execute commanded motion with feedback from their motion sensors, sense the environment using application specific sensors and relay this data to a coordinating base station. The nodes receive high-level commands from the base station to follow a movement path.

- **Base Station** – The base station is a node at a safer location with access to higher computing power than the swarm nodes. The nodes relay information to the base station. The base station computes the probabilistic coverage from obtained data and directs the motion of all the explorer nodes as per the coverage algorithm. The base station provides a real-time coverage status to the users of the swarm.

2.4 SugarMap Coverage Algorithm

The base station runs the SugarMap online coverage algorithm. Every node in SugarMap starts with an empty grid map of the world with a known initial position. The size of a grid cell is equal to the sensing radius of each sensor. As explorer nodes move, they approximately measure their motion using sensors (optical flow and inertial), and collect RToF signatures from landed anchor nodes, and relay this data to the base station.

The SugarMap algorithm directs explorer nodes to perform a depth first search (spanning tree coverage [7]) of the environment but to avoid areas already covered by other nodes. This minimizes sensing overlap and speeds up coverage. To achieve this, the algorithm constructs a common grid map for areas covered by all nodes.

However, the motion estimation of MAV nodes is approximate and exact coverage area is hard to determine. The algorithm uses a particle filter to model the coverage uncertainty (due to motion and sensing uncertainty) of each SensorFly node. Every node is represented by n particles that track the coverage path of the SensorFly on the grid, where the weight of each particle gives the probability of each cell on its path being covered. The algorithm detects revisits in node paths by matching RToF signatures and uses this to update particle weights. The algorithm combines the probabilistic coverage map of all particles by summing and normalizing the cell coverage probabilities to arrive at a coverage map for each SensorFly.

Thus, the SugarMap algorithm constructs a probabilistic spanning tree for each node's coverage. Each node of the spanning tree corresponds to a unique RToF signature of a covered area, while the edges are defined through the relative motion estimates. Using the common initial position of all nodes, individual maps are overlaid on a common grid map by again combining the probabilities of individual cells to arrive at a global coverage map of the swarm. The algorithm uses this global coverage map to direct nodes to cells with a lower probability of having been covered.

3. SYSTEM DESCRIPTION

In this section, we describe the details of the different components and algorithms used in SugarMap. We discuss the deployment of anchors, the path planner that commands explorer nodes, and the particle filter based probabilistic coverage map estimation.

3.1 Deployment of Anchors

The SugarMap system initializes by deploying a subset of MAV nodes in the environment to act as radio anchors. The absolute position of the anchor nodes is not critical. However, it is desirable for the anchors to be dispersed over the search space to provide robust RToF signatures. Dispersion algorithms for constrained robots have been studied in prior work [17, 13, 15] and many potential approaches can be employed. In SugarMap, the anchors deploy using a *fiducial dispersion* algorithm [17] based on the nodes' ability to obtain RToF measurements from other nodes. The objective of the dispersion algorithm is for nodes to spread away from each other till they encounter obstacles or lose radio range with other anchors.

Figure 3 shows a flowchart of the algorithm used to deploy anchor nodes. The base station commands a sub-set of MAV nodes (designated as anchors) to move in randomly chosen directions. Every node periodically attempts to obtain radio RToF measurements from other anchors and relay it to the base station. If any anchor node is out of radio range and does not respond, the node attempts to retrace its path by turning 180°. Conversely, if all nodes are in range, each RToF measurement (d_i) is compared to a proximity threshold (T_p). If no other anchor is within the proximity threshold, the node lands and deploys. Otherwise, the node

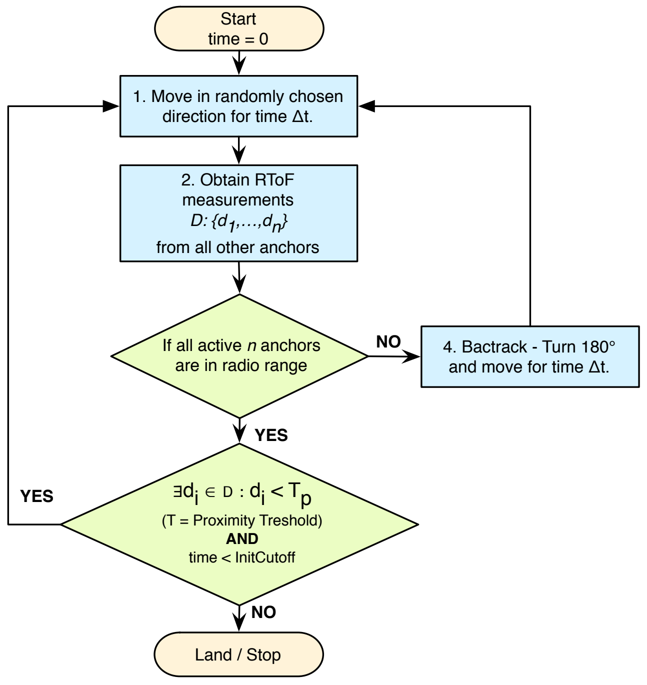

Start
time = 0
1. Move in randomly chosen direction for time Δt.
2. Obtain RToF measurements D: {d1,...,dn} from all other anchors
If all active n anchors are in radio range
NO
4. Bactrack - Turn 180° and move for time Δt.
YES
∃di ∈ D : di < Tp
(T = Proximity Treshold)
AND
time < InitCutoff
YES
NO
Land / Stop

Figure 3: Figure shows a flowchart of the constrained dispersion algorithm used to deploy anchor nodes at initialization of SugarMap.

moves in a random direction and recursively repeats the above sequence of operations.

In addition, a maximum cutoff time for anchor deployment ($InitCutoff$) is defined. If an equilibrium is not reached within the cutoff time, the nodes still land and deploy.

It must be noted that the explorer nodes use RToF measurements from the anchor nodes as a signature and not as a measure of distance. The uniqueness of the signature is a feature of the position of the anchors and multi-path radio propagation characteristics of the physical environment. The system relies on measurements from a relatively large number of anchor nodes (at least greater than 4) to improve signature uniqueness. The system does not assume any particular anchor node topology, such as non-linear, as would be the case for computing location coordinates from deployed anchors.

3.2 Coverage Path Planner

The central idea of the SugarMap coverage algorithm is to coordinate explorer node movements so as to cover the environment with a certain measure of confidence.

In the application scenario, the explorer MAV nodes have no a priori knowledge of the search space. However, due to manual placement, the relative pose and position of the explorer nodes is known. SugarMap approximates the world with a grid with a cell size of D, where D is determined by the effective radius of the application specific sensor deployed on the MAV node.

A MAV node is commanded to move along 4 basic direction relative to itself – N, S, W and E, and must be located to within the D-size cell. For simplicity, we first describe the algorithm assuming nodes can be localized to the D-size cell

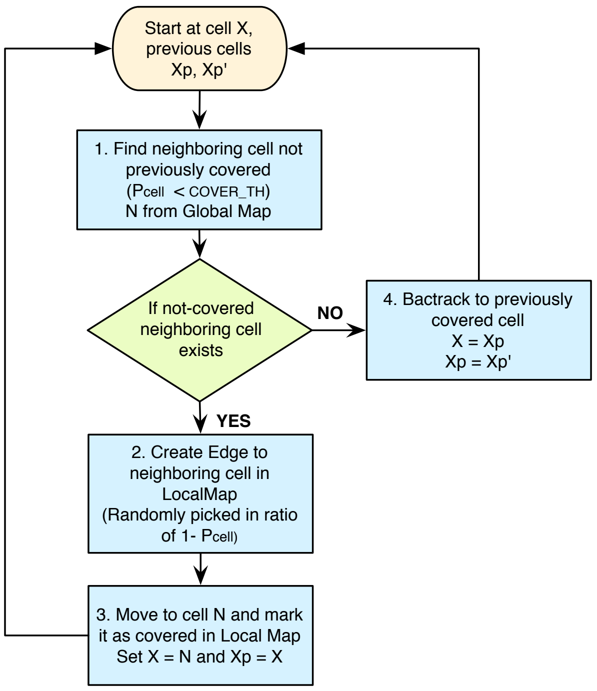

Start at cell X, previous cells Xp, Xp'
1. Find neighboring cell not previously covered (Pcell < COVER_TH) N from Global Map
If not-covered neighboring cell exists
NO
4. Bactrack to previously covered cell X = Xp Xp = Xp'
YES
2. Create Edge to neighboring cell in LocalMap (Randomly picked in ratio of 1- Pcell)
3. Move to cell N and mark it as covered in Local Map Set X = N and Xp = X

Figure 4: The figure shows a flow chart of the SugarMap coverage path planner algorithm. The distributed algorithm runs on every node and builds a spanning tree of covered nodes using a depth first exploration approach.

with a certain probability. In section 3.3, we explain how this is achieved by the SugarMap system in practice.

Each MAV node builds a local spanning tree of cells that it discovers, while tracking the portion of the map covered by other nodes through a global map. The global map is obtained by aggregating the local maps from all nodes in the swarm. The spanning tree is built using a depth-first approach – 1) find a neighboring cell that hasn't been covered(by itself or by any node in the swarm); 2) create a tree edge to the neighboring cell; 3) move to the cell and mark it as covered in its local map; recursively repeat steps 1,2,3 with this cell; 4) if all the neighbors are covered or blocked (obstacles), the node backtracks along its local spanning tree to the previously covered cell.

To account for the uncertainty in sensing and position of MAV's, the coverage of a cell is not designated by a binary value but a real number between 0 and 1, representing the probability that a cell is covered. The algorithm considers a cell as not covered if the coverage probability of a cell (P_{cell}) is less than a desired coverage confidence threshold ($COVER\_TH$). The algorithm chooses the next cell from neighboring non-visited cells randomly, where the probability of picking a cell is inverse of its confidence of coverage. When all neighboring cells are above the coverage confidence threshold, the algorithm increments the probability of picking the last covered cell biasing the node towards backtracking on its traversed path.

Figure 4 shows a flowchart of the coverage path planner algorithm. The algorithm is distributed and runs on every node. The nodes are coordinated using the global map that is aggregated from each node's local coverage estimates.

3.3 Probabilistic Coverage Maps

The path planner requires each node to mark coverage in a grid map of the search space. In addition, the planner requires a global coverage map that is obtained by aggregating the coverage maps of individual swarm nodes.

As the initial location of all nodes is known, the system can potentially track the cells covered by nodes keeping track of the commanded motion of the MAV nodes. However, MAV nodes have low-quality inertial sensors and imperfect actuators. This makes it impossible for MAV nodes to accurately execute the commanded path. SugarMap uses a particle filter based probabilistic approach that combines approximate motion noise models of the explorer nodes with radio round-trip time-of-flight (RToF) position signatures, to create a common probabilistic coverage map for the swarm.

3.3.1 Particle Filter (PF)

A particle filter (PF) [12] is a Bayesian estimation method used to estimate a system's state based on noisy sensor information. In a particle filter, a probability distribution $p(x)$ is represented by a number of N weighted samples or *particles* $x^{[i]}, i = 1..N$, with weights $w^{[i]}$ as:

$$p(x) = \sum_i w^{[i]} \delta(x^{[i]} - x) \tag{1}$$

With a initial probability $p(x_0)$, which is represented as equally distributed samples with equal weights, a recursive update at time t_k to estimate system state x_k is performed in 3 steps:

1. **Prediction** – Every particle $(x^{[i]}_{k-1}, w^{[i]}_{k-1})$ of the *a posteriori* distribution $p(x_{k-1}|z_0, \ldots, z_{k-1})$, where $z_0, \ldots, z_k$ are measurements about the system up to time t_k, is replaced according to a process model $p(x_k|x_{k-1})$. The process model incorporates knowledge of the evolution of the system over time. In coverage estimation, we use an empirically obtained actuation noise profile from the MAV nodes' motion sensors to construct this model. Thus a new set of particles $(\tilde{x}^{[i]}_k, \tilde{w}^{[i]})$ is obtained representing the *a priori* distribution.

2. **Correction** – The weight $w^{[i]}$ of every sample of the *a priori* distribution, is updated according to a measurement model as:
$$w^{[i]} = \tilde{w}^{[i]} \cdot p(z_k|\tilde{x}^{[i]}_k), \sum_i w^{[i]} = 1 \tag{2}$$
With the weight update, the prior particles now approximate the *a posteriori* probability. In coverage estimation, we use the radio RToF signatures obtained by the explorer nodes from the anchor nodes to compute the term $p(z_k|\tilde{x}^{[i]}_k)$, which relates the coverage state of the system to its observation.

3. **Resampling** – A new set of particles is drawn with replacement from the prior set with probability of a particle being drawn given by its weight. The samples are weighted equally. The resampling step prunes the less likely state estimates.

3.3.2 Applying PF To Coverage Estimation

In this section, we describe the application of a particle filter to estimate coverage in SugarMap. We first consider a single MAV node. Each MAV node is represented by N particles. Each particle holds an array ($LocalMap$) representing the particle's coverage on a local grid map of the search space. Cells not covered are set to 0 in the array, while covered cells are set to 1. The state of a particle $x^{[i]}_k$ at time t_k is given by this $LocalMap^{[i]}_k$.

Prediction: In the prediction step, the $LocalMap^{[i]}_k$ is updated according to the commanded motion of the MAV node and a actuation noise model. The MAV node executes a motion command – turn and velocity, using feedback from its inertial sensor (gyro) and optical flow velocity sensor. Therefore, if c_x and c_y are coordinates of the last cell covered by a particle, v_k is the velocity, ϕ_k is the change in pose, the $LocalMap^{[i]}_k$ is calculated as:

$$\begin{pmatrix} c_x \\ c_y \end{pmatrix}^{[i]}_k = \begin{pmatrix} c_x \\ c_y \end{pmatrix}^{[i]}_{k-1} + (v^{[i]}_k \cdot \delta t) \begin{pmatrix} sin(\phi^{[i]}_k) \\ cos(\phi^{[i]}_k) \end{pmatrix} \tag{3}$$

$$LocalMap^{[i]}_k \left[c^{[i]}_{x_k}\right] \left[c^{[i]}_{y_k}\right] = 1 \tag{4}$$

Noise is added to the velocity and turn commands as per the empirically obtained actuation noise models $p(n_v)$ and $p(n_\phi)$. This is based on the actuation mechanism of the MAV platform used. Thus, $v^{[i]}_k$ and $\phi^{[i]}_k$ are obtained as:

$$v^{[i]}_k = \quad v_k + n^{[i]}_v \ , \ n^{[i]}_v \text{ is drawn from } p(n_v) \tag{5}$$

$$\phi^{[i]}_k = \quad \phi_k + n^{[i]}_\phi \ , \ n^{[i]}_\phi \text{ is drawn from } p(n_\phi) \tag{6}$$

Correction: In the correction step, the weight of the particles is updated using the radio RToF area signatures obtained by the MAV node. A detailed explanation of the radio RToF area signatures is given in Section 4.4.

The central idea of the correction step is to compare the radio RToF signature s^k obtained at time t_k to a set of known area signatures $S : \{s^1, \ldots, s^j\}$. The set S is empty at initialization. A list of location estimates $(c^{[i]}_x, c^{[i]}_y)$ from every particle is stored for every signature in set S.

A distance function $f(s^k, s^j)$ gives the distance between the currently obtained signature and previously known signatures in set S. If distance given by $f(s^k, s^i)$ is more than a threshold (SIG_TH) for all signatures in set S i.e. the obtained signature is of an unexplored area, signature s^k is added to S. Conversely, if the distance is less than the threshold, the signatures are considered similar and the area is designated as a previously covered area.

On identifying a matching previously visited signature s^j, the corresponding distance in location estimates for the current signature s^k and known signature s^j for each particle is computed as:

$$d^{[i]}_{s^j,s^k} = \quad EuclideanDist\left\{(c_x, c_y)^{[i]}_{s^j}, (c_x, c_y)^{[i]}_{s^k}\right\} \tag{7}$$

Consequently, the weights of the particles are updated as a function of the distance between the two estimates as:

$$w^{[i]} = \frac{\left\{\sum_{i}^{N} d^{[i]}_{s^j,s^k}\right\} - d^{[i]}_{s^j,s^k}}{\sum_{i}^{N} d^{[i]}_{s^j,s^k}} \tag{8}$$

Resampling: In the resampling step, a new set of particles is drawn with replacement using the weights as the probability of drawing. The weights are reset to their initialization values.

Merging: Finally, the local maps of each particle are combined to compute a node coverage map at time t_k as:

$$NodeMap^j_k = \frac{\sum_{i=1}^{N} LocalMap^{[i]}_k}{N} \tag{9}$$

Similarly, the central base station combines the NodeMaps from all MAV nodes to arrive at the GlobalMap for the swarm:

$$GlobalMap_k = 1 - \prod_{j=1}^{n} \left(1 - NodeMap^j_k\right) \tag{10}$$

where n is the number of nodes in the swarm.

4. EVALUATION

In this section we evaluate the capability of SugarMap to command a swarm of MAV's collaboratively cover a search space in realistic large-scale simulations and in a real MAV testbed. We characterize the performance of our system with respect to the percentage of coverage achieved by a fixed number of nodes as a function of time. Percentage coverage as a function of time is an essential metric in applications such as disaster response, where the speed of covering an area and identifying survivors is critical to the success of the operation. Even random walk algorithms eventually attain complete coverage of an area, however we seek to minimize the time for coverage through coordinated node movement approaching that of an algorithm with accurate location measurements.

Due to the limited suitable MAV coverage algorithms available in literature, we compare SugarMap to currently available online coverage algorithms for MAV nodes namely random walk coverage [8], and Online Multi-robot Spanning Tree Coverage (OMSTC) [9] that assumes accurate node location is available.

4.1 Simulation Setup

We have developed a simulation environment for the SensorFly MAV indoor sensor swarm to evaluate our coverage algorithm at scale in a realistic scenario. The simulator incorporates a realistic physical arena, MAV node virtual sensors and sensor noise models, MAV node mobility models, wireless communication and radio path loss model, and application specific sensing models. These models are generated through data collected from MAV nodes in indoor environments. In addition, the simulator allows users to program the logic for actuation of MAV's and implement coverage algorithms such as SugarMap or Random Walk. The simulator, now ported to Python, extends previous work [20] to include additional application scenarios and adds the ability to interface with actual hardware and run hardware-in-loop simulations.

To simulate SugarMap we configure the different aspects of the simulator as follows:

- **Arena** – We assume an indoor search and rescue scenario, where nodes are required to move and cover a continuous indoor space such as a room. The simulation arena constitutes the search space and we evaluate SugarMap in a $20m \times 20m$ square arena, but with multiple configurations of boundary walls and obstacles.

- **Node Sensors** – We model the MAV node based on the SensorFly [19] MAV platform used for our real implementation. The node is equipped with 3 virtual sensors: an inertial gyroscope sensor, an optical flow velocity sensor, an ultrasonic altitude measurement sensor, and a radio with round-trip time-of-flight measurement capability. The application sensor has a sensing footprint equal to a square cell of $0.5m \times 0.5m$. This corresponds to the cell size of the grid map used by SugarMap to compute coverage.

- **Node Mobility** – Like the SensorFly nodes, the simulated nodes can be commanded to turn by a desired angle and move forward for a designated time. The nodes execute the commands with feedback from their virtual sensors, incorporating the errors from their associated noise models. The nodes operate at speeds of $0.25m/s$ to $0.45m/s$.

- **Anchor Nodes** – We model the deployment of anchor nodes as described in Section 3.1. The simulation clock starts after initialization of the system and anchor node deployment.

- **Radio Model** – Shadowing with a path loss exponent of 3 is used as the radio link model, which is an estimate for an indoor single-floor scenario [21]. The movement algorithm ensures that nodes maintain connectivity.

- **Simulation Time-steps** – The simulation time-step is configurable and determines the resolution of node movement that can be recorded by the simulator. The nodes execute their movement based on their velocity, specified in meters per simulation-seconds. Therefore, the minimum distance that can be achieved by the node in one command is determined by the duration of the simulation time-step. For the purpose of the evaluation, we selected a time step of $1sec$ that enables nodes to cover a distance of $0.25m$ to $0.45m$ in a single simulation tick. In terms, of sensing footprint ($0.5m \times 0.5m$), a node can travel from one sensing cell to another in a single tick.

The coverage algorithm or the base-station *brain* of the swarm does not receive the real physical locations of the nodes but only their presumed locations based on movement commands. The nodes are also not aware of the map and are assumed to be at the center of an infinite space at initialization. However, nodes are aware of their relative initial positions with respect to each other in accordance with the manual deployment at the entrance to the search space.

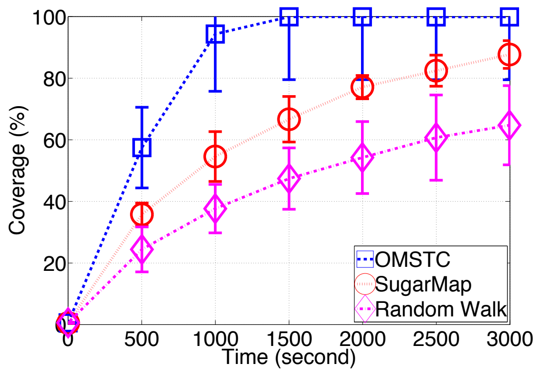


Figure 5: The figure shows the percentage coverage of two nodes as a function of time for the SugarMap algorithm, Random-Walk, and a spanning tree coverage algorithm – OMSTC that assumes locations of nodes are known. The error bars show the standard deviation over 10 runs.

4.2 Comparing Coverage

We evaluate the performance of SugarMap by comparing the percentage coverage achieved as a function of time to state-of-the-art existing online coverage approaches applicable to MAV swarms.

We implement two approaches – random walk and online multi-robot spanning tree coverage (OMSTC) (assumes locations of nodes is known with high accuracy and precision). Random walk [8] is the most popular approach used by resource-constrained nodes when no location information or prior knowledge of the environment is available. This provides us a baseline for comparison. On the other hand, OMSTC [9] is a guaranteed multi-node coverage algorithm that assumes that nodes can locate themselves in the space. Although, perfect location is unattainable, this presents us with an ideal system for comparison.

Figure 5 shows the percentage coverage as a function of time for SugarMap, Random-Walk (baseline) and OMSTC (ideal). The simulation uses 4 explorer nodes for all algorithms and runs for 3000 simulation seconds. SugarMap performs better than random walk achieving a faster rate of coverage in the simulation scenarios.

4.3 Analyzing SugarMap

In addition to comparing coverage with other approaches, we analyze the trade-offs involved in selecting parameters for a SugarMap deployment.

4.3.1 Impact of Number of Anchors

The SugarMap system uses anchor nodes to obtain a radio signature for various covered areas. Individual radio measurements (round-trip time-of-flight) are subject to variability as shown in Figure 14. However, a signature combining measurements from multiple anchors is more stable.

Figure 6 shows the coverage achieved as a function of time while the increasing number of anchors deployed by the system. The number of explorers is set at 4 and the number of particles used for each SugarMap node is 10. A stable area signature enables SugarMap to determine visited locations better and compute better trajectories for the swarm nodes.

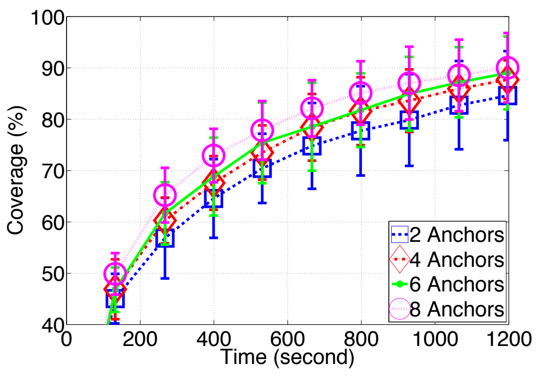


Figure 6: The figure shows the evolution of coverage for a varying number of deployed anchor nodes with 4 explorer nodes. The error bars show the standard deviation over 10 runs.

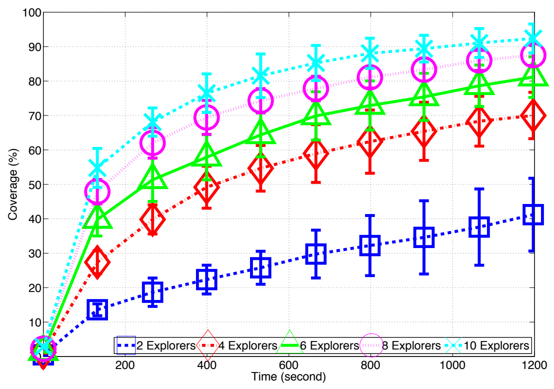


Figure 7: The figure shows the evolution of coverage for varying number of explorer nodes with 6 anchor nodes.The error bars show the standard deviation over 10 runs.

Therefore, coverage improves with an increase in the number of anchors. However, depending on the variance in individual measurements, the benefit of increasing the number of anchors diminishes after a point. The number of anchors also depends on the size of the area.

With fewer anchor nodes (2 nodes in Figure 6), the radio location signatures are not sufficiently unique and the accuracy of coverage estimation suffers. Without estimation of coverage, the path planning algorithm cannot efficiently compute motion and coverage achieved shows high variance similar to a random walk algorithm.

4.3.2 Impact of Number of Explorers

Figure 7 shows the coverage achieved as a function of time for a varying number of explorer nodes. 6 anchor nodes are used for the simulation. As nodes explore the environment in parallel, the larger the number of explorer nodes the greater is the coverage. Moreover, unlike heuristic algorithms, SugarMap coordinates node movements to reduce the overlap in area coverage. Thus, increasing the number of nodes shows

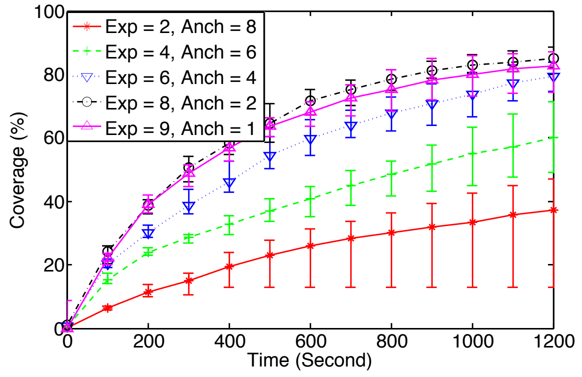


Figure 8: The figure shows the percentage coverage for fixed size 10-node deployment with varying number of anchors and explorers.

an almost linear increase in speed of coverage. However, due to noise in sensor and actuation, some overlap in individual node coverage exists in SugarMap. Therefore, the benefit of increasing explorer nodes diminishes after a point.

4.3.3 *Anchor-Explorer Trade-off*

Figure 8 shows the rate of coverage for varying ratio of anchor nodes to explorer nodes for a fixed-size (10-node) deployment in a $20m \times 20m$ arena. The plot shows the trade-off between anchor nodes and explorer nodes. It is evident that for a given size deployment, the coverage rate is determined largely by the number of explorer nodes. The larger the number of explorer nodes the faster the coverage attained. However, the incremental performance gain of introducing additional explorer nodes diminishes when the number of anchors is reduced below 4. With a small number of anchor nodes, the radio location signatures are not sufficiently unique and the accuracy of SugarMap coverage estimation decreases with time. Consequently, the nodes cannot coordinate efficiently and performance drops.

4.3.4 *Impact of Number of Particles*

SugarMap uses a particle filter to model the uncertainty in actuation of MAV nodes. Each node is represented by a number of particles, each of which holds an estimate of the node's coverage map. Figure 9 shows the percentage coverage achieved by a deployment of 2 explorer nodes and 6 anchor nodes in 400 ticks of the simulation, while the number of particles used is varied. The greater the number of particles, the better is the estimation of the area covered by SensorFly's. This translates to a lower overlap in sensed area and a higher speed of coverage. As a trade-off, the larger number of particles require higher memory and computation at the base station. In addition, the choice of particles depends on the sensor and actuation noise of the MAV node. Larger sensor noise requires higher number of particles.

4.3.5 *Impact of Search Space Geometry*

We evaluate the performance of SugarMap in multiple space configurations. The simulation environment enables the programmer to specify walls (red cells), open navigable area (blue cells) and obstacles (green cells) by providing a bitmap image. Figure 10 shows 3 different geometries used to evaluate SugarMap. Geometry (a) is an almost circular open space with a few obstacles. Geometry (b) is a U-shaped region with a narrow leg similar to a room with a connecting hallway. While, geometry (c) models a narrow corridor.

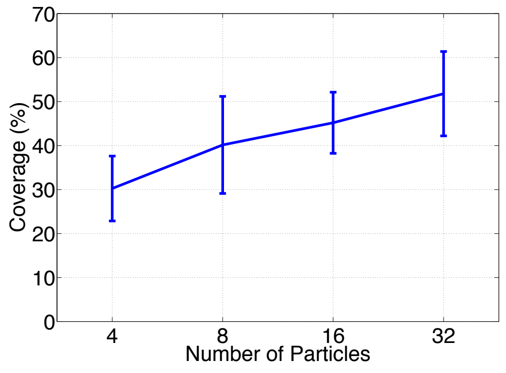


Figure 9: The figure shows the coverage achieved in 400 simulation ticks as a function of the number of particles used in the SugarMap algorithm. The number of explorer nodes is fixed at 2 and number of anchors nodes is 6. The error bars show the standard deviation over 10 runs.

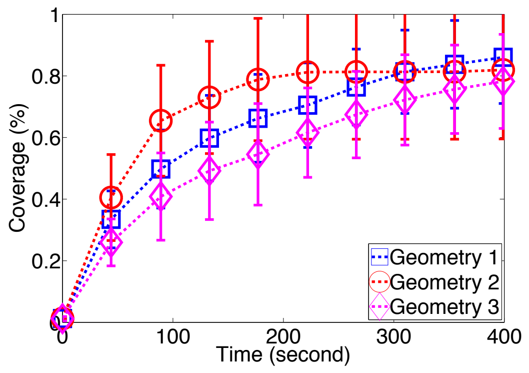
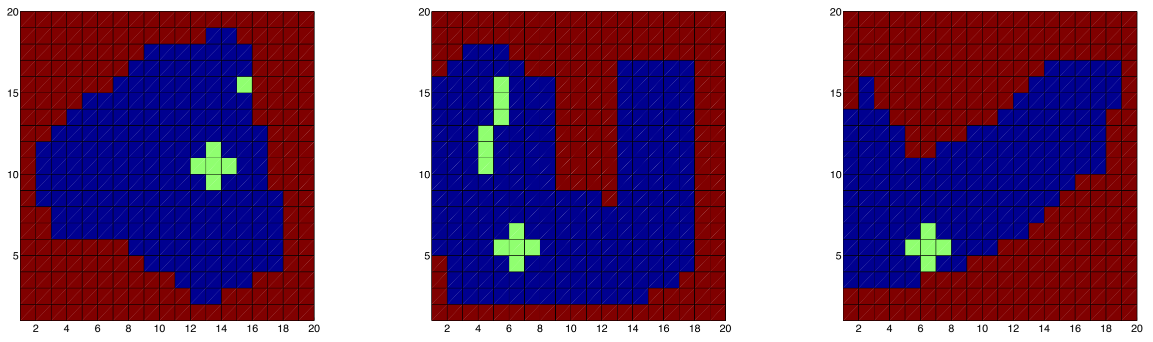


Figure 11: Figure shows the coverage as a function of time for the SugarMap algorithm in 3 different geometries (Figure 10) of the simulation arena. The error bars show the standard deviation over 10 runs.

Figure 11 shows the coverage as a function of time for the 3 different arena geometries. The curves for the 3 spaces follow each other closely showing that SugarMap is robust to variations in space configuration.

4.3.6 *Algorithm Parameters*

The SugarMap algorithm has two principal parameters – (1) a threshold to match radio signatures (SIG_TH), for determining if an observed signature is similar to a previously observed one; and (2) a threshold to select the next location for movement ($COVER\_TH$), for determining the confidence of coverage for a neighboring location as per the global coverage map.



(a) (b) (c)

Figure 10: The figure shows 3 geometries of the simulation arena with different obstacle configurations used to evaluate the robustness of the SugarMap algorithm. The red cells are walls, the blue cells represent navigable space, and the green cells designate obstacles.

The SIG_TH is determined empirically based on the radio feature and metric used to compute the similarity of signatures. The SugarMap system implementation uses a vector of RToF (Round-trip Time-of-Flight) measurements from a set of anchor nodes as the radio signature. Purohit et. al. [19] provide a more detailed evaluation of RToF measurements and their correlation with distance. We use Euclidean distance as a metric to measure similarity between N-dimensional RToF signatures, given by,

$$d_{i,j} = \frac{\|\overrightarrow{s^i} - \overrightarrow{s^j}\|}{\#\text{ of visible common anchors}} \tag{11}$$

For the purpose of our evaluation, we selected a SIG_TH of 1.2, obtained by measuring the average RToF signature distance for 20 signatures in a 2-meter radius circle, over 5 distinct locations in our lab.

The $COVER\_TH$ is defined as an exit condition for the path planning algorithm. Since, the coverage map represents confidence probabilistically it does not achieve 100% coverage confidence for a location in finite time. The parameter determines the coverage confidence that is desired by the application for the path planner to avoid revisiting a location. We chose 95% as the coverage threshold for our evaluation signifying a high degree of certainty that the area is covered.

4.3.7 Actuation Noise Model

The SugarMap algorithm uses particle filters to account for the uncertainty in the movement actually executed by MAV nodes on a given command. A model of the actuation noise is required in the prediction step of the particle filter as described in Section 3.3.2. The actuation uncertainty depends on the sensors and control algorithm for the specific MAV platform. For the purposes of the evaluation, we use the SensorFly platform to empirically determine an approximate noise model. The SensorFly platform uses PID control with feedback from an optical flow sensor (velocity) and a gyro (turn) to execute the commanded motion. We measured the standard deviation in turn executed by the platform using a vision-based ground truth measurement relative to the commanded turn value to derive a model. For the evaluation, we used a normal distribution with a standard deviation of 20% of the commanded turn value to predict turn noise. Similarly, a normal distribution with a standard deviation of 15% of the commanded velocity value was used as the velocity noise model.

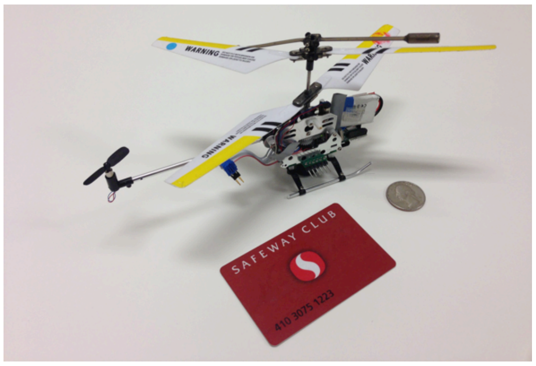


Figure 12: The figure shows a SensorFly node used to implement SugarMap on the prototype testbed.

4.4 MAV Testbed

In addition to the simulation experiments, we evaluate SugarMap in our MAV swarm testbed. We implement SugarMap on the SensorFly [19] MAV platform.

The SensorFly platform is equipped with a 8-bit 16Mhz AVR AtMega128rfa1 micro-controller, inertial motion sensors – 3-axis accelerometer and 3-axis gyro, an ultrasonic ranger for altitude estimation, an optical flow sensor for velocity estimation, and a 802.15.4a compatible radio with Round-trip time-of-flight (RToF) measurement capability. The entire platform is under 30g in weight and has a flight time of 6-8 minutes. The SensorFly nodes are capable of receiving high-level movement commands such as *"Turn X degrees"* and *"Move forward X seconds"*. The nodes execute the movement in accordance with on-board PID control algorithms utilizing angular and translational velocity feedback from the nodes' gyro and optical-flow sensors. Figure 12 shows a SensorFly node with the battery and basic set of sensors.

The testbed consists of a $5m \times 3m$ arena where the SensorFly nodes move. A grid is painted on the arena dividing it

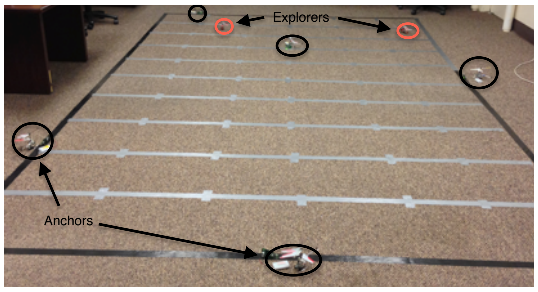


Figure 13: The figure shows the arena for the MAV swarm testbed with deployed SensorFly nodes.

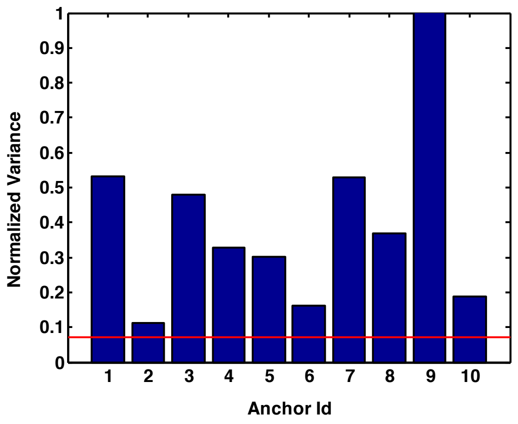


Figure 14: Figure shows the normalized variance in RToF measurements from a explorer node and 10 individual anchors (bars) over a 3 day period. The variance of the signature (combined set of 10 individual measurements) is shown by the horizontal line. The low variance of the combined vector makes it suitable as an area signature.

into cells of $0.5m \times 0.5m$. A camera is deployed on the ceiling to capture the entire arena in its field of view. A workstation running a color blob detection algorithm uses the feed from the camera to compute the *ground-truth* location of all MAV nodes on the grid. The cell size of the grid is chosen to reflect the sensing radius of the MAV node. The blades of the SensorFly nodes are affixed with red-tape so as to be easily detectable by the vision-based ground-truth tracking system. Figure 13 shows the arena of the testbed with deployed SensorFly anchors and explorers.

RToF Area Signature Stability

The explorer SensorFly nodes use a set of round-trip time-of-flight (RToF) measurements from stationary anchor nodes as a signature of area covered. The signature enables explorer nodes to determine if a covered area is being revisited. This information is used for by the coverage path planner for coordination and by the probabilistic coverage estimation algorithm for updating particle weights.

The round-trip time-of-flight method measures the elapsed time between the host node sending a data signal to the remote node, and receiving an acknowledgment from it. Our implementation, based on the SensorFly platform, uses physical layer timestamps and hardware-generated acknowledgments to compute a RToF measurements [19, 18].

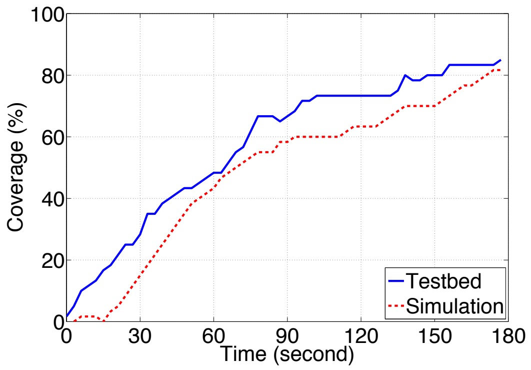


Figure 15: The figure shows the evolution of coverage for the 7-node MAV swarm testbed experiment (solid-line). The result is in agreement with our simulation (dotted-line) using similar arena-size, number of anchors, number of explorers, and the actuation noise model.

We perform an experiment to measure the repeatability of a set of RToF measurements and hence validate its suitability as a area signature. RToF measurements from 10 nearby anchors were collected at a mobile node location over a 3 day period, in a lab environment. The lab is subject to frequent environment changes due to movement and activity of people. Figure 14 shows the variance in RToF measurements from each anchor. We observe a large variation in individual RToF measurements over the 3-day period. However, despite these environmental variations, we observe that the overall ten-dimension signature vector of the RToF measurements remains consistent (red-line), showing very low variance over the entire test period.

Testbed Results

In our testbed experiment, we use a swarm of 7 MAV nodes running the SugarMap algorithm. 5 nodes are allowed to disperse and deploy as anchors at initialization. 2 nodes are used as explorers. The two explorer nodes start at grid locations [1,1] and [2,1], respectively, and move as per commands given by the base-station. The explorer nodes obtain RToF readings from anchor nodes and relay it to the base-station at every step. The actual coverage achieved is computed for evaluation purposes using the ground-truth vision-based tracking system of the testbed. The testbed experiment runs for 3-minutes of node flying time which enables us to execute multiple runs on a single charge.

Figure 15 shows the evolution of coverage attained by the 7-node swarm running SugarMap (solid-line). The swarm is able to cover 85% of the arena in the 3-minute experiment time. For comparison, we also plot a simulation result (dotted-line) using a similar size arena, 5 anchor nodes, 2 explorer nodes, and the actuation noise model for SensorFly nodes. The results from the simulation and real experiment closely follow each other as time progresses, validating the simulation setup and methodology.

5. DISCUSSION

Having presented the SugarMap coverage algorithm, we note that several aspects warrant further discussion and could result in possible extensions to this work.

5.1 Multi-hop Communication

The explorer nodes obtain RToF measurements from the deployed anchor nodes as a location signature. This requires explorer nodes to be in single-hop communication range with a large enough sub-set of anchor nodes to obtain a unique radio location signatures. In this paper, we limit our application and evaluation to a single-space coverage scenario where all nodes are within radio range. The explorer nodes stop exploring if they lose radio range with deployed anchor nodes. Thus explorer nodes employ a single-hop communication scheme to relay data back to the base station. We continue to explore large-space or multi-room scenarios in our ongoing work, where the network can extend its reach by progressively deploying exploring nodes as new anchors and requires support for multi-hop communication schemes.

5.2 Distributed Operation

The current SugarMap implementation has a central base station that aggregates the coverage estimates of individual nodes to arrive at a global coverage map. The global coverage map is utilized by the explorer nodes to decide their next movement. For a single-space scenario with single-hop communication range this simple mechanism reduces communication and computation on the nodes. For large-space and multi-room scenarios, it is desirable for nodes to broadcast their local coverage maps and for each node to itself aggregate neighboring nodes' maps to determine global coverage. Such distributed operation would remove the single point of failure at the base station and also provide latency advantages as nodes would not have to communicate with the base station over multiple hops. However, a distributed scheme would require consideration for lack of consensus on coverage maps due to lossy wireless links. We seek to explore this in our future work.

6. RELATED WORK

The coverage problem has been addressed in the past by research in multi-robot coverage algorithms that focus on the space swept by the robot's sensor. Choset [3] provides a survey of early coverage algorithms and classifies them into *off-line*, in which a map of the area is known beforehand, and *on-line* algorithms, in which the area is unknown. The MAV swarm applications demand an on-line multi-node coverage approach. In this section, we discuss some of the existing work in online coverage that is applicable to MAV swarms.

Gage [8] analyzes randomized robot coverage for robots without costly localization sensors or valuable computational resources for calculating their position. A random search does not guarantee coverage but may be the only approach with very low capability sensor nodes in absence of motion sensing.

Wagner et al. [24] propose a pheromone based stigmergic algorithm, where nodes coordinate by leaving markers in the environment. The algorithm is heuristic in nature and requires nodes to be capable of leaving and detecting physical markers in the environment. This is not practical for lightweight MAV nodes.

A series of simultaneous localization and mapping (SLAM) approaches [22] exist for multi-robot systems that use laser range-finders or computer vision algorithms (to extract visual features) in conjunction with noisy odometry to construct maps of the area and obtain coverage. This approach is very promising for ground-based robots or larger aerial nodes. Currently available laser rangers tend to be too bulky for deployment on MAV's, while vision-based approaches require better computation and communication capabilities.

Howard et al. [1] considers the problem of deploying a mobile sensor network in an unknown environment. The approach assumes that each node is equipped with a sensor that allows it to determine the range and bearing of both nearby nodes and obstacles, such as scanning laser range-finders or omni-cameras. Using these, the system constructs fields such that each node is repelled by both obstacles and by other nodes, thereby forcing the network to spread itself throughout the environment. The algorithm has the advantage of requiring no communication between nodes but does not guarantee completeness. In Spreading-Out [14], Batalin et al. present an algorithm for robot teams without access to maps or a Global Positioning System (location). The robots are assumed to be equipped with planar laser range-finders, color camera and vision beacons, and robots select a direction away from all their immediate sensed neighbors and move in that direction. The approach is heuristic with the premise that robots must 'spread out' over the environment in order to achieve good coverage.

Hazon et al. [9] present a guaranteed robust multi-robot coverage algorithm based on spanning tree coverage paths. Each robot works within an assigned portion of the work area, constructing a local spanning-tree covering this portion, as it moves. It coordinates movement with other robots to minimize overlap in coverage. However, the algorithm assumes the robots have access to relative location of all nodes in the area. SugarMap employs the concept of spanning trees to guarantee completeness of coverage but further extends it to remove the requirement for node location. Moreover, SugarMap introduces probabilities into the coverage map to account for and overcome the sensor and actuation uncertainty of MAV nodes.

Related work in robotics and sensor networks [4, 10, 23] uses particle filters for monitoring robot or human position in indoor environments. Their primary focus is to localize nodes using measurement from motion sensors and observed environmental landmarks. Our approach seeks to guarantee coverage of an unknown space by a swarm of nodes based on the coordinated motion commands, using particle filters to account for the uncertainty in MAV actuation.

7. CONCLUSION

This paper presents SugarMap, a location-less coverage system that allows a swarm to collaboratively and efficiently attain sensing coverage of a target area with an MAV swarm. SugarMap does not require sophisticated or bulky sensors, advanced on-device processing abilities, or a pre-existing location infrastructure as do many state-of-the-art prior approaches. We provide a comprehensive evaluation of the SugarMap system through large-scale simulations and a real implementation on the SensorFly [19] platform. Our experimental evaluations show that SugarMap performs significantly better than existing coverage approaches for resource-limited (in terms of sensors, energy, location, processing power) mobile sensing nodes such as random-walk especially in larger locations with fewer nodes. This algorithm will enable swarms of the new class of realistic resource constrained micro-aerial vehicles in new applications.

8. ACKNOWLEDGEMENTS

Special thanks to our shepherd Dr. Luca Mottola and the anonymous reviewers for their insightful and constructive comments. This research was supported by Intel, NSF under grant 1121690-CNS1135874-NSF-ZHANG and DARPA under grant 1080247-D11AP00265-DOI-ZHANG. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, DARPA, Intel, CMU, or the U.S. Government or any of its agencies.

9. REFERENCES

[1] M. J. M. Andrew Howard. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, 2002.

[2] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragón, C. Martínez, and M. Olivares. Visual 3-D SLAM from UAVs. *Journal of Intelligent and Robotic Systems*, 55(4-5):299–321, Jan. 2009.

[3] H. Choset. Coverage for robotics - A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, May 2001.

[4] W. B. Dieter Fox . Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, 1999.

[5] L. Doitsidis, S. Weiss, A. Renzaglia, M. W. Achtelik, E. Kosmatopoulos, R. Siegwart, and D. Scaramuzza. Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision. *Autonomous Robots*, 33(1-2):173–188, Mar. 2012.

[6] S. Fu, H.-y. Liu, L.-f. Gao, and Y.-x. Gai. SLAM for mobile robots using laser range finder and monocular vision. In *2007 14th International Conference on Mechatronics and Machine Vision in Practice*, pages 91–96. IEEE, Dec. 2007.

[7] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pages 1927–1933. IEEE, 2001.

[8] D. W. Gage. Randomized Search Strategies With Imperfect Sensors. In *SPIE Mobile Robors VIII*, pages 270–279, Boston, 1993.

[9] N. Hazon, F. Mieli, and G. Kaminka. Towards robust on-line multi-robot coverage. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1710–1715. IEEE.

[10] L. Klingbeil and T. Wark. A wireless sensor network for real-time indoor localisation and motion monitoring. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 39–50. IEEE Computer Society Washington, DC, USA, 2008.

[11] S. Koenig and D. Kempe. Multi-robot forest coverage. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3852–3857. IEEE, 2005.

[12] J. Liu. *Monte Carlo strategies in scientific computing.* Springer Publishing Company, Jan. 2008.

[13] L. Ludwig and G. Maria. Robotic swarm dispersion using wireless intensity signals. In *International Symposium on Distributed Autonomous Robotic Systems*, 2006.

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

[15] J. McLurkin and J. Smith. Distributed Algorithms for Dispersion in Indoor Environments using a Swarm of Autonomous Mobile Robots. *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2004.

[16] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In *Eighteenth national conference on Artificial intelligence*, pages 593–598, July 2002.

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

[18] Nanotron Tecnologies Gmbh. Real Time Location Systems (RTLS). 2007.

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

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

[21] D. Rutledge. *Investigation of indoor radio channels from 2.4 GHz to 24 GHz.* IEEE.

[22] S. Thrun. An Online Mapping Algorithm for Teams of Mobile Robots. *International Journal of Robotics Research*, 2001.

[23] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, May 2001.

[24] I. Wagner, M. Lindenbaum, and A. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.

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