ASTRO: A System for Off-grid Networked Drone Sensing Missions

RICCARDO PETROLO, Konica Minolta Lab Europe, Italy
ZHAMBYL SHAIKHANOV, YINGYAN LIN, and EDWARD KNIGHTLY, Rice University, USA

We present the design, implementation, and experimental evaluation of ASTRO, a modular end-to-end system for distributed sensing missions with autonomous networked drones. We introduce the fundamental system architecture features that enable agnostic sensing missions on top of the ASTRO drones. We demonstrate the key principles of ASTRO by using on-board software-defined radios to find and track a mobile radio target. We show how simple distributed on-board machine learning methods can be used to find and track a mobile target, even if all drones lose contact with a ground control. Also, we show that ASTRO is able to find the target even if it is hiding under a three-ton concrete slab, representing a highly irregular propagation environment. Our findings reveal that, despite no prior training and noisy sensory measurements, ASTRO drones are able to learn the propagation environment in the scale of seconds and localize a target with a mean accuracy of 8 m. Moreover, ASTRO drones are able to track the target with relatively constant error over time, even as it moves at a speed close to the maximum drone speed.

CCS Concepts: • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies** → *Machine learning*; *Distributed computing methodologies*;

Additional Key Words and Phrases: Cyber-physical systems, drones, sensing drones

ACM Reference format:
Riccardo Petrolo, Zhambyl Shaikhanov, Yingyan Lin, and Edward Knightly. 2021. ASTRO: A System for Off-grid Networked Drone Sensing Missions. *ACM Trans. Internet Things* 2, 4, Article 24 (July 2021), 22 pages.
https://doi.org/10.1145/3464942

1 INTRODUCTION

In this article, we present ASTRO, a system that realizes distributed data-driven sensing missions via autonomous drone networks. In contrast to leader-follower applications, we empower each drone with independent on-board machine learning capabilities, along with communication and coordination mechanisms that enable them to cooperatively realize mission objectives. We experimentally demonstrate, for the first time, heterogeneous and dynamic on-drone learning from live sensor data without the necessity of a pre-trained process. We study exemplary missions in which

This research was supported by Cisco, Intel, and by NSF Grant No. CNS-1801865.
Authors' addresses: R. Petrolo (corresponding author), Konica Minolta Lab Europe, Rome, Italy, email: riccardo.petrolo@konicaminolta.it; Z. Shaikhanov (corresponding author), Y. Lin, and E. Knightly, Rice University, 6100 Main Street MS 366, Houston, TX 77005, USA; emails: {zs16, yingyan.lin, knightly}@rice.edu.

2577-6207/2021/07-ART24 $15.00
https://doi.org/10.1145/3464942

a group of drones find and track a mobile radio using only signal-strength measurements and without communicating to ground stations during the mission. Applications of this scenario include off-the-grid search-and-rescue for a beaconing phone and urban public safety. In this context, we make the following three contributions.

First, we present the system architecture and implementation of ASTRO. The key features include (i) a software architecture that enables realization of high level mission logic at a layer above low-level flight controller functions; (ii) a hardware architecture that enables "plug and play" of diverse mission-oriented sensors beyond flight controller sensors such as gyroscope and compass; (iii) **software-defined radio (SDR)**-based communication capabilities.

Second, we implement and demonstrate online light-weight on-drone machine learning algorithms that exploit domain knowledge. Namely, we begin with an Friis' equation and learn parameters such as path loss exponents and antenna gains via batch gradient descent for the non-linear regression; likewise, we select k-means clustering for the drones to reach a coordinated target location estimate with low-error despite the high error of non-linear Friis-based multi-lateration. The approach therefore realizes heterogeneous learning in which different drones can be subjected to different propagation environments (different shadowing, pathloss, etc.). Moreover, the method is "online" in that no prior training is required to realize the mission objectives, such that we do not require drones to have flown prior missions in similar environments.

Together these design features enable *off-grid* missions. Namely, we do not require ground control stations for sending and receiving control signals and/or data and thus do not require an air-to-ground network. This contrasts with existing systems that use either human or machine control from the ground [1–3]. Consequently, missions can proceed during outages with ground control or in areas without communication infrastructure, e.g., during disaster response. Nonetheless, when network infrastructure is available, ASTRO can utilize it to (i) offload machine learning computations to ground resources if beneficial and (ii) report back mission results while the mission is in progress or after completion and landing.

Third, to demonstrate ASTRO, we implement all of the aforementioned aspects and report on over 1,000 h of test flights.[1] We deploy a suite of missions to test the key components of ASTRO. In an exemplary mission, all drones are launched from a common location outside of the range of the target, so that no ASTRO drone can initially detect the target. Without any prior training data and pre-train process, the drones coordinate during an initial search phase in which they cover the largest possible area given their constraints of sensing capabilities, inter-drone connectivity, battery life limit, and so on. Once a drone has identified the target, it requests that all other drones aid it for the greatest possible tracking accuracy, guided by ASTRO's on-drone embedded machine learning algorithms. In other experiments, we test heterogeneous propagation environments by hiding the target inside an angular concrete slab weighing several tons and surrounded by buildings. Last, we test dynamic environments by continuously moving the target, including with abrupt changes of the targets' direction.

This article extends our earlier work in Reference [4] by (*i*) providing more system design details, e.g., summarizing our lessons during the 24 months of drone system design and exploration, (ii) performing various system design space exploration, e.g., studying the relationship between the accuracy of parameter estimation and the number of samples and that of the localization accuracy and algorithmic parameters, and (iii) evaluating the ASTRO system in a more thorough manner (over 1,000 h of test flights vs. 500 h in Reference [4]), e.g., considering targets hidden under a big slab in addition to those in open environments and studying the impact of drone formation on the achieved localization accuracy.

[1]One of the authors has a drone pilot license as required at the test locations.

The remainder of this article is organized as follows. In Section 2 we describe related works. In Section 3, we present the design and implementation of ASTRO, describing exemplary mission scenario in Section 3.2 and ASTRO's on-drone online light-weight machine learning algorithms in Section 3.3. In Section 4, we present extensive experimental results to demonstrate the capabilities of ASTRO in real-world outdoor environments. Finally, Section 5 concludes the article.

2 RELATED WORK

Autonomous drones. Recent advances have demonstrated autonomous drone flights [5]. Thanks to open-source communities such as ardupilot [6], it is now possible to achieve high control accuracy [7]. Commercial platforms are also advancing: for example, in 2016, Amazon tested the first autonomous drone delivery [8]. Likewise, the DJI Phantom, the Yuneec Typhoon, GoPro Karma, Skydio, which are widely used for aerial cinematography, present features such as *Follow Me* that make drones automatically follow a target person [9]. However, in these applications, a drone is still tethered to the smartphone, which behaves as a **ground control station (GCS)**. In contrast, we leverage open software and hardware to build an open modular drone platform that can support tetherless operation, in which an air-to-ground interface is not mandatory. Consequently, ASTRO can leverage outcomes of other projects such as DARPA FLA [10], that target to develop navigation, perception, planning, and control algorithms to enable autonomous and high-speed flights through unknown environments. Our architecture also suits customizable commercial platforms such as the Matrice 100 [11], a quadcopter for developers that can be programmed via a Software Development Kit.

Networked drones. Multiple drones working cooperatively to complete a task can be more efficient than single-drone platforms, especially given flight-time constraints [12–14]. Moreover, drone sensor networks can enrich the accuracy of the sensed data. Recently, several studies have been conducted on aerial and in particular on ad hoc networks [15–17]. Although their features have been widely investigated, there are only a small number of implementations, and most target inter-drone routing protocols [18, 19], joint trajectory and transmit power optimization [20], and so on. Such advances can be used to improve the performance of ASTRO in different mission contexts.

Spectrum sensing and localization. Recently, UAVs have been used for conducting aerial spectrum sensing and localization. An implementation of a system based on a mini unmanned helicopter—equipped with an off-the-shelf smartphone as a wireless sniffer—is designed to fly over a pre-designed flight route and send collected logs to the ground where localization algorithms are applied [1]. Likewise, a drone equipped with 8 wireless sniffers can perform aerial wardriving [21]. In contrast, ASTRO drones analyze measurement data on-board and in real-time, and adjust their flight paths accordingly. Moreover, in contrast to single-drone solutions [1, 21–24], ASTRO realizes a network of multiple drones that communicate and coordinate among themselves.

Machine learning-based localization and tracking. There has been a growing interest in leveraging the adaptability and information extraction capability of machine learning algorithms for localization and tracking. One line of research employs machine learning for RSSI-based localization of wireless sensor nodes. Specifically, learning regression tree [25], support vector machines [26], neural networks [27, 28], and k nearest neighbors [29], have been proposed. While these techniques show the potential of machine learning algorithms for robust and accurate localization or tracking, they are all developed for indoor environments and employ static sensor nodes. Moreover, their learning processes are supervised, i.e., pre-labeled training dataset are required, limiting their widespread applicability to many applications in which pre-labeled data cannot be obtained or are too expensive to be collected. Another research thrust develops machine learning-based localization and tracking techniques for drones. Machine learning algorithms have been employed, including Reed-Xiaoli algorithm [30], support vector machines [31, 32], and convolutional

neural networks [33–35]. As these approaches require a camera and rely on image processing, their computational complexity requires resources beyond the capabilities of the companion computer that we implemented on ASTRO. Nonetheless, they can potentially be applied in image-based missions provided that computation resources can meet size, weight, and power constraints. Thus, in contrast to prior works, ASTRO is the first work to leverage unsupervised light-weight machine learning algorithms for online drone-based moving target localization and tracking in outdoor environments.

Swarm optimization and control. Great progress has been made for swarm optimization and control of robots [36]. Solutions to position a team of robotic routers to provide communication coverage to the remaining client robots [37] have been proposed. Likewise, other research thrusts focus on the path planning while taking into the coverage for the area and obstacle avoidance [38]. Other approaches focus on bringing edge/cloud computing on UAVs to achieve high **quality of service (QoS)** guarantees [39]. In contrast, we do not take leader-follower or bio-inspired approaches with simple inter-drone rules to enable large scale swarms. We target that each drone has equal learning and autonomy capabilities to complete the mission, albeit with greater on-drone computing requirements.

3 SYSTEM DESIGN

In this section, we first describe ASTRO system architecture, providing details about each of the modules of the system. Next, we discuss ASTRO in the context of the mission scenario of finding and tracking a mobile radio target. We describe different phases of the mission and discuss our machine learning algorithms that leverage fundamental principles of signal propagation to learn the environment and track the target. Then, we discuss how the design principle, i.e., combining domain knowledge and machine learning algorithms, is general and can be applied to a broad class of sensing missions. For the ease of exposition, we focus on the case of the mobile target in this article and provide an example of our recent work [40] that uses ASTRO to find and track hazardous plumes from extreme environmental events with software defined radio sensors replaced with gas sensors.

3.1 System Architecture

We design ASTRO as a modular system as shown in Figure 1. Based on different mission requirements (e.g., sensing specifications, desired flight time, weight budget), modules can be modified and upgraded, while new modules can also be added to the system to enable new features. Below, we describe the main ones, and in Section 3.4, we discuss Mission Sensing in the context of radio frequency sensing, gas sensing, as well as other potential sensing modalities.

MISSION LOGIC (SKYNET). Mission Logic (SkyNet) is the core of the system while all other modules interface and interact with it by offering their computing, sensing, navigation, and networking resources. This architecture enables simple and intuitive methods for developing mission applications and makes ASTRO maximally use case-agnostic. In SkyNet, we develop a set of *API libraries* [41] to abstract out the complexities of avionics from the main mission objective. The libraries (e.g., skyengine, skysense, and skycontrol) contains Python wrapper functions that offer easy access to different functionalities of the drone infrastructure, enabling system flexibility without sacrificing usability. In addition, we design a framework for users to develop their missions. The framework is based on the principle of **peer-to-peer (P2P)** distributed computation and provide convenient interfaces for interoperability between drones.

To test and validate the mission logic, we also develop a *simulator* that closely mimics the physical deployment of drones in the field. The simulator is built by virtualizing drones and infrastructures such as sensors and flight controllers in docker containers [42] and visualizing

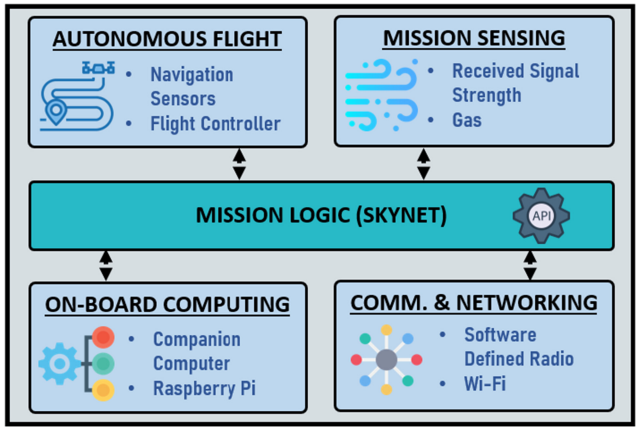
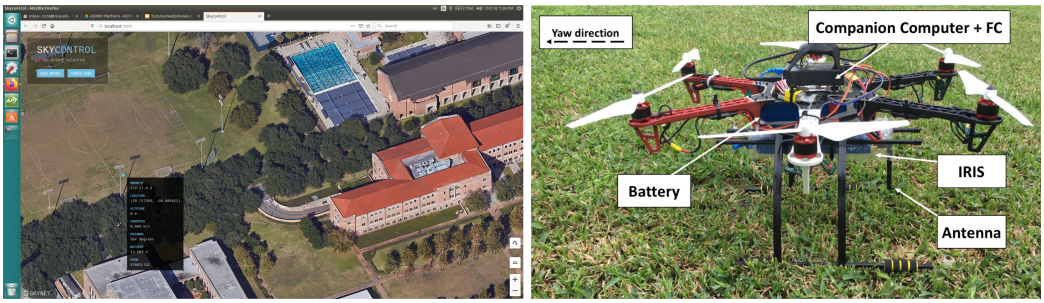


Fig. 1. ASTRO System Architecture.



(a) Mission simulator (b) Hexadrone

Fig. 2. An illustration of the ASTRO system that comprises both software architecture and hardware architecture.

mission status and drone state in a web application as illustrated in Figure 2(a). It is a convenient tool for testing underlying mission logic in a virtual environment, both for single and multi-drone missions. With the simulator, users can debug any mission logic flaws early in the development phase and avoid potential failures in the field.

AUTONOMOUS FLIGHT. ASTRO drones fly autonomously in a mission, without requiring human-operator in the loop. To this end, we equip drones with Flight Controller, embedded hardware that manages flight dynamics of the drone. The Flight Controller is programmed with autopilot firmware [43], which communicates with on-board navigation sensors such as gyroscopes, accelerometer, barometer, and magnetometer. Based on the specified mission logic, it then temporally and spatially adapts the state of the drone. The Flight Controller can also receive commands from a human-operator, allowing him/her to take control over the drone as necessary, e.g., emergency landing.

For our targeted outdoor environments, we use GPS to aid navigation and localization. Differential-GPS can be used in case some missions quire highly accurate positioning, providing centimeter-level accuracy. While standard cost-efficient GPS has an error in the scale of meters. To autonomously detect and avoid obstacles, we also equip drones with light-weight 360° lidar sensors. The obstacle avoidance can be set to have the highest priority so that the flight path could be temporarily altered from what is directed by the mission objective to safely continue the mission.

COMMUNICATION AND NETWORKING. In ASTRO, drones form a network and establish drone-to-drone communication. For that, drones can employ either an internal Wi-Fi card embedded on Companion Computer, attachable USB Wi-Fi dongles, or an SDR. While Wi-Fi card and Wi-Fi dongle (e.g., Panda Wireless with 802.11n [44]) provide simple plug-and-play solutions, SDR allows tuning for a wide range of frequencies. We employ IRIS software defined radio, which has a transceiver that can be tuned from 50 MHz to 3.8 GHz, with up to 56 MHz of contiguous bandwidth and a 12-bit ADC.

For ease of implementation, ASTRO drones maintain connectivity with a routing layer employing **Better Approach to Mobile Ad hoc Networking (BATMAN)** [45], a mesh routing operating at Layer 2. BATMAN eliminates the need to spread information concerning network changes to every node in the network and supports multiple network interfaces per node. While they are capable of multi-hopping, exploration of network effects is beyond the scope of this article, and all drones remain mutually in range during our experiments.

ON-BOARD COMPUTING. Together with the Flight Controller and the resource-constrained embedded hardware responsible for flight coordination, ASTRO drones mount a more powerful dedicated Companion Computer to perform several important computing tasks. First, it runs SkyNet and interacts with the Flight Controller to manage avionics—they communicate via the MAVLink protocol [46] and passes computed flight decisions in real-time. Next, it processes data received from the sensing module and makes the pre-processed data available for further decision making. Then, it runs mission logic and executes *online light-weight machine learning* methods that we discuss in Section 3.3. Last, the Companion Computer manages the *networking* stack, which enables discovering and maintaining air-to-air links between drones, and if available and needed, air-to-ground links to the GCS, e.g., to offload machine learning computations.

In ASTRO, we used two different embedded hardware, Raspberry Pi3 and UpBoard, as the Companion Computer that can be mounted on the drone. Raspberry Pi3 is a standard and more cost-efficient option that runs Raspbian Stretch Lite operating system. In contrast, UpBoard is more suitable for missions that have high computation requirements, since it hosts a powerful Intel Atom x5 Z8350 quad-core processor and 2 GB RAM. Note that other single-board computing devices can also be used as Companion Computer such as Nvidia Jetson, and so on, as long as they fulfill requirements in terms of limited size, weight, and power consumption.

TETHERLESS. Most of the drone systems available on the market support only a single aerial vehicle and *require* a GCS. The GCS is typically a standard computer used as a centralized planning and adaptation point, e.g., to configure mission parameters such as coordinating to cover through waypoints navigation and the action to take at each waypoint [7]. Likewise, popular platforms such as GoPro Karma, DJI Phantom, and Skydio, realize *Follow Me* applications in which drones autonomously follow a mobile target. However, in these applications, a drone is still tethered to the smartphone, which behaves like a mobile GCS [3].

In contrast, ASTRO can realize tetherless off-grid operations in which drones communicate with each other but are not required to maintain communication with any ground systems, and computation can be performed on-drone. This enables successful mission execution even during outages with the ground station. As described above, ASTRO drones are enhanced with a software and hardware architecture that supports tetherless and autonomous network sensing missions. Thus, GCS and air-to-ground interfaces are not mandatory in our architecture.

3.2 Mission Scenario

To further motivate the design and evaluation of ASTRO, here we describe an exemplary mission. We consider a mobile target that has a signature of transmitting between 563 and 568 MHz. The drones are equipped with sensors that can measure only received power and their high level

objective is to find and track the target. We consider that drones' maximum velocity exceeds the target as otherwise, the target can simply "out run" the drones. In the scenario, ASTRO drones are launched from a location that we do not assume to be within the range of the target, i.e., the mission can be initiated from a point in which no drone can sense the target. The drones coordinate to realize the mission and if available, can use ground control to offload the machine learning computations. While scenarios with multiple targets can be implemented in ASTRO, the required algorithmic extensions are beyond the scope of this article. ASTRO does not achieve the objectives by simple "war driving" in which drones exhaustively sense over a region. Instead, ASTRO drones must adapt their flight patterns according to sensed data to collaboratively and adaptively find and track the target for minimizing the mission time, which is more practical in real-world environments given the limited flight time of drones.

3.3 On-Drone Online Light-Weight Machine Learning

The main challenges of the described mission, as well as distributed mobile sensing missions in general, reside in data processing and control aspects: (i) the lack of prior knowledge or training data about the mission environments; (ii) sensed data are often error-prone due to various real-world environmental disturbances, e.g., potentially failed sensors or drones, and so on; (iii) the requirement for real-time responses despite the limited processing capability of the on-drone computer; (iv) constrained drone flight time due to weight constraints.

To address these challenges, we design ASTRO to operate in two phases: "search and learn" and "swarm and track," depending on whether the target has been initially identified by any drones. Specifically, ASTRO drones could be launched from an initial arbitrary location, even outside of the sensing range of the target. During the search and learn phase, drones fly in accordance with pre-computed partitioned zones and planned paths, obtained by solving a multiple traveling salesman problem under the constraints of the number of drones, flight time, on-board sensing range, and the target environment [47]. The zone partition will determine how to divide the target area into zones so that each drone takes charge of one zone, and the planned paths provide guidance for drones to fly within their corresponding zones. Also, during this phase, each drone works independently and learns the signal propagation model parameters while trying to identify a potential target (see Section 3.3.1). Once a drone is in the range of a target, it informs all other drones. Subsequently, all drones will swarm to the target and switch to the tracking phase in which they work collaboratively to locate and track the target (see Section 3.3.2), while continuously updating their estimates of the propagation environment.

3.3.1 Search and Learn. The goal of this phase is to independently learn model parameters as well as to search for a target. Thus, it features both a machine learning algorithm and the Friis propagation model to exploit *domain knowledge*.

We select received power and the Friis model to track the target to realize it with a group of single-antenna drones, without any additional hardware, and potentially encountering high vibrations and mobility. Namely, compared to the time of arrival [48], angle of arrival [49], or time difference of arrival [50], our method is more suitable for drone-based applications, considering drones' mobility and strict constraint on weight. However, locating a transmitter solely based on received power can encounter a high error, as this metric is highly affected by reflections and other interactions of the radio signal with the environment. We, therefore, couple domain knowledge with a simple machine learning algorithm to dynamically adapt model parameters in real-time, aiming to improve modeling and tracking accuracy while maintaining low complexity.

At the beginning of the search phase, both the environment-dependent propagation parameters and the target location information are unknown. To tackle the challenges of modeling the varied

interactions between received power and environment and capturing the target position in a short time, we formulate the problem as a machine-learning framework using nonlinear regression [51] and use a batch **gradient descent (GD)** algorithm [52] to solve it. This approach is well suited for the search phase as it is an effective yet simple method for enabling the learning of the model parameters and locating the target simultaneously in real-time.

Friis' propagation model indicates that receive power decays with the distance D from the transmitter to the receiver, such that in dBm [53]:

$$P(D) = P(D_0) - 10\gamma \log_{10}\left(\frac{D}{D_0}\right) + \sigma, \tag{1}$$

in which $P(D_0)$ is the received signal strength at a reference distance from the transmitter D_0 (incorporating transmit and receive antenna gains, etc.), σ is a zero-mean Gaussian random variable that represents shadowing, and γ is the environment and frequency dependent path loss exponent. For our purposes, we can express Equation (1) as

$$P(D) = \alpha log_{10}(D) + \eta, \tag{2}$$

collapsing the parameters to two unknowns such that α is proportional to the environment and frequency dependent path-loss exponent, and η incorporates the remaining parameters.

Denoting the positions of the drone and target in the Cartesian coordinate system as (x, y, z) and (x^c, y^c, z^c), respectively, the distance is $D = \sqrt{(x^c - x)^2 + (y^c - y)^2 + (z^c - z)^2}$. Note that each ASTRO drone is equipped with an on-board RF receiver and GPS receiver that can obtain P and (x, y, z), respectively. Thus, the goal in the search and learn phase is to utilize these measurements to learn the model parameters (α, η) and the target position (x^c, y^c, z^c).

ASTRO drones continuously sample their own position (x, y, z) and the received signal strength P. In a batch GD algorithm, parameters are updated using a batch of measured data at each step. We choose the batch size to make sure that the model parameters (α, η) and the transmitter position (x^c, y^c, z^c) can be assumed to be constant within each batch of measurements under the given sampling rate. We formulate the estimation of the five parameters (i.e., $\alpha, \eta, x^c, y^c, z^c$) as a nonlinear regression problem, in which a cost function $J(\alpha, \eta, x^c, y^c, z^c)$ can be defined as

$$J(\alpha, \eta, x^c, y^c, z^c) = \frac{1}{2M} \sum_{i=1}^{M} \left(P_i^e(\alpha, \eta, x^c, y^c, z^c) - P_i^m\right)^2, \tag{3}$$

where M denotes the number of measurements in each batch, P_i^m denotes the measured received signal strength in the ith measurement, and $P_i^e(\alpha, \eta, x^c, y^c, z^c)$ denotes the ith estimated signal strength according to Equation (2) using the estimated path-loss propagation parameters and target location. Essentially, J measures how close the calculated RSSI based on the estimated parameters and location is to the corresponding measured signal strength. Mathematically, our approach aims to solve the following optimization problem:

$$\arg\min_{\alpha, \eta, x^c, y^c, z^c} J(\alpha, \eta, x^c, y^c, z^c). \tag{4}$$

It is in general intractable to obtain a closed-form solution to Equation (4). We instead employ a batch GD algorithm [52], which is simple and effective for solving nonlinear regression problems. Specifically, this algorithm iteratively searches possible model parameters and target locations that minimize the cost function in Equation (3). To do so, it first randomly assigns initial values for the parameters to be learned, and then iteratively performs the following update:

$$\theta_{k+1} = \theta_k - \delta \frac{\partial}{\partial \theta_k} J\left(\alpha_k, \eta_k, x_k^c, y_k^c, z_k^c\right), \tag{5}$$

where θ represents one of the five parameters $(\alpha, \eta, x^c, y^c, z^c)$, δ denotes the learning rate, and k denotes the iteration index. Note that all the parameters need to be updated simultaneously. The partial derivative term $\frac{\partial J}{\partial \theta_k}$ corresponding to the five parameters in Equation (5) can be obtained as follows:

$$
\begin{aligned}
\frac{\partial J}{\partial \alpha_k} &= \frac{1}{M}\sum_{i=1}^{M}\left[\left(P_i^e\left(\alpha_k, \eta_k, x_k^c, y_k^c, z_k^c\right) - P_i^m\right) log_{10}(D_i)\right], \\
\frac{\partial J}{\partial \eta_k} &= \frac{1}{M}\sum_{i=1}^{M}\left(\left(P_i^e\left(\alpha_k, \eta_k, x_k^c, y_k^c, z_k^c\right) - P_i^m\right),\right. \\
\frac{\partial J}{\partial x_k^c} &= \frac{1}{M}\sum_{i=1}^{M}\left[(P_i^e(\alpha_k, \eta_k, x_k^c, y_k^c, z_k^c) - P_i^m)\frac{\alpha(x_k^c - x_i)}{D_i^2 ln10}\right], \\
\frac{\partial J}{\partial y_k^c} &= \frac{1}{M}\sum_{i=1}^{M}\left[(P_i^e(\alpha_k, \eta_k, x_k^c, y_k^c, z_k^c) - P_i^m)\frac{\alpha(y_k^c - y_i)}{D_i^2 ln10}\right], \\
\frac{\partial J}{\partial z_k^c} &= \frac{1}{M}\sum_{i=1}^{M}\left[(P_i^e(\alpha_k, \eta_k, x_k^c, y_k^c, z_k^c) - P_i^m)\frac{\alpha(z_k^c - z_i)}{D_i^2 ln10}\right].
\end{aligned}
\tag{6}
$$

In summary, our approach for the search and learn phase relies on a single model to learn both the model parameters and target position based on the measured data of each drone independently, i.e., there is no exchange of data (neither raw data nor intermediate results) among drones. Its simplicity (i.e., low data computation and zero data communication overhead among drones) enables fast identification of the target within each drone's search area, maximizing the possible covered area and minimizing the mission time. This is critical due to the drones' constrained flight time.

3.3.2 Swarm and Track. Once an ASTRO drone identifies the target, all N drones switch to the swarm and track phase. The goal in this phase is to collaboratively locate and track the target in real-time,[2] while leveraging learned model parameters and target estimation from the previous phase. Because the target could be moving, the key challenge lies in ensuring that the tracking algorithm has a short tracking latency (i.e., low data computation and communication overhead) to enable real-time tracking of the moving target, while at the same time meeting the specified tracking accuracy in the presence of potential environmental disturbances, e.g., wind, potentially failed sensors or drones, and so on.

To address this challenge, we utilize a fast (without the need to exchange raw data) and effective (because the estimated target position at each time step involves corresponding intermediate results from all drones and its outliers is mostly ruled out) tracking algorithm that combines intermediate results from multiple drones at different locations to perform collaborative tracking and a K-means clustering algorithm to suppress the large errors that are commonly observed in path-loss-based propagation models and real-world drone-based mobile sensing experiments. Although there are other well-established noise suppression techniques, such as Kalman filter and its variants, their complexity when applying to nonlinear systems can be prohibitively high [54]. In contrast, the K-means algorithm is simple yet efficient, as will be described in this section and demonstrated in the experimental section. Without loss of generality, we describe the algorithm for the case of 2D tracking; we extended to the 3D tracking case for our implementation.

[2]For other mission objectives, e.g., multi-target missions, a subset of drones can remain in the search phase. For simplicity of exposition, we consider a single target here.

After each of the N drones learns its own propagation parameters, the distance between the ith drone and the target can be calculated directly from a single power measurement, i.e.,

$$D_i = 10^{(P_i - \eta_i)/\alpha_i}, i = 1, 2, \ldots, N. \tag{7}$$

Combined with GPS measurements, in a 2D Cartesian coordinate system, the N drones have the following N nonlinear equations:

$$\begin{aligned} &(x^c - x_1)^2 + (y^c - y_1)^2 = D_1^2, \\ &(x^c - x_2)^2 + (y^c - y_2)^2 = D_2^2, \\ &\ldots \\ &(x^c - x_N)^2 + (y^c - y_N)^2 = D_N^2, \end{aligned} \tag{8}$$

Using multilateration, a unique solution of the target position (x^c, y^c) at each measured time instance from these N nonlinear equations exists if there are at least three independent measurements/equations, i.e., $N \geq 3$. In other words, this algorithm requires three drones for 2D tracking and four drones for 3D tracking in the ideal scenario (i.e., no environmental disturbances, failed sensors or drones, etc.). While additional drones will not help in the ideal scenario, their use can help to rule out noisy tracking in practice. Although there are more advanced localization approaches in the literature such as fingerprinting-based localization [55] and machine-learning strategies [56], they, however, require pre-collecting data with many pre-known target positions, building a large database, and training algorithms. Due to the computation and flight time constraints of the drones, those approaches are impractical in our scenario.

Ideally (i.e., measurements are error-free), the target location can be obtained by merely solving Equation (8) based on each set of measurements. The advantage of such an approach lies in the fact that it relies on a simple algebraic solution and a single set of data at a time, thus leading to a fast response time, i.e., lower data computation and communication overhead, and thus tracking latency. However, the direct results of (x^c, y^c) from solving Equation (8) have a poor accuracy in practical environments, because the data of received signal strength, i.e., P_i in Equation (7), is often noisy due to environment dependent path-loss, fading, and shadowing effects, and Equation (8) merely employs one datum at a time. Furthermore, the accuracy of the resulting estimated target position can be made worse by potential environmental disturbances affecting drones (i.e., wind), failed sensors or drones, and so on. As such, we utilize a K-means clustering algorithm [57] to filter out the noisy results of (x^c, y^c) that are obtained directly from solving Equation (8), for achieving improved localization/tracking accuracy (see the experimental section for more details). In particular, given a set of adjacent results $(\mathbf{L}_1, \ldots, \mathbf{L}_M)$ obtained by solving Equation (8) with $\mathbf{L}_m = (x_m^c, y_m^c)$, K-means clustering partitions the G results into K ($K \leq G$) sets $\mathbf{S} = \{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_K\}$ to minimize the **within-cluster sum of squares (WSS)**. Mathematically, the objective is to find

$$\underset{\mathbf{S}}{arg\,min} \sum_{k=1}^{K} \sum_{\mathbf{L} \in \mathbf{S}_k} \|\mathbf{L} - \mathbf{c}_k\|^2, \tag{9}$$

where c_k denotes the geometric centroid of the kth cluster (i.e., $\mathbf{S}_k$). To determine the best number of clusters for K-means clustering, one commonly used method is the elbow method [57], which is to choose a K value at which the WSS decreases abruptly. We experimentally explore this in Section 4.

To illustrate with $K = 3$, each G adjacent set of results are partitioned into three clusters by the on-drone K-means clustering algorithm, and the centroid of the cluster that has the largest sample size is chosen as the estimated target location at each time instance, as shown in Figure 3. This way, the outliers of the estimated target positions due to the noisy received signal strength,

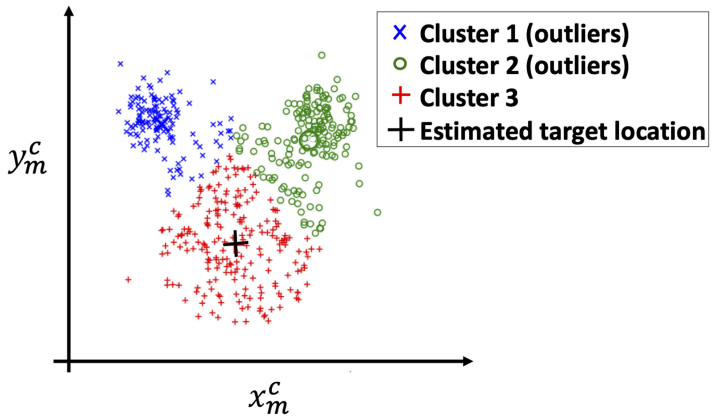


Fig. 3. An example to illustrate how outliers of the estimated target locations are ruled out using the K-means algorithm, in which the geometric centroid of each cluster is marked in bold.

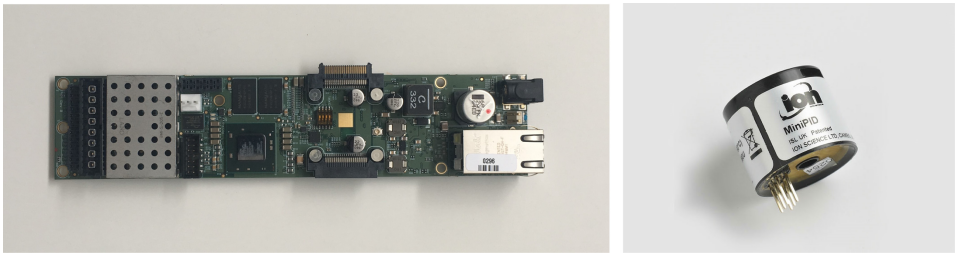
i.e., P_i, in Equation (7), potential environmental disturbances and failed sensors or drones, and so on, can be excluded. Note that we choose the size of the adjacent results, i.e., G, in each K-means clustering step to optimize the tracking accuracy, given the sampling rate and target moving speed. Also, while the search and learn phase involves no exchange of data, the swarm and track phase involves exchanging the intermediate results of drones. In particular, the distance between each drone and the target, i.e., D_i in Equation (7), is passed to a central drone at each time step, which can be chosen arbitrarily to solve Equation (8) for obtaining the estimated target position.

3.4 Discussion

Thus far, we discussed ASTRO in the context of the radio sensing, e.g., via an IRIS software defined radio as shown in Figure 4(a), and described our approaches that feature a combination of domain knowledge aided with machine learning algorithms to ensure fast response time and high localization/tracking accuracy. With these functionalities, ASTRO could be applied to many important real-time applications, such as finding and locating a victim in search and rescue operations via sensing radio signature emitting from the victim's cell phone or locating and tracking radio-collared wildlife to monitor endangered wildlife and ecosystems [58]. Also, the proposed methods described in Section 3.3 can be *generalized* to other sensing missions by modifying the model of the phenomena being sensed, namely, replacing Equation (2) by a domain specific equation. For example, in a gas sensing mission, an equation would be required to map raw measurement data (e.g., a spectral signature for laser spectroscopy [59]) to gas concentration levels in **parts per billion (ppb)** volume and modify remaining steps accordingly.

ASTRO has been extended to multiple works [40, 60] that encompass different sensing missions. For instance, in our recent ASTRO+ work [40], we repurpose ASTRO for environmental air pollution sensing. To do so, we equip the drones with lightweight **Volatile Organic Compounds (VOCs)** gas sensors as shown in Figure 4(b) as well as temperature, humidity, and wind sensors. We then explore the joint impact of weather conditions, namely, temperature and humidity levels, and dynamic airflow from drones on the gas sensors' measurement quality. Leveraging these effects, we design a drone mission planning strategy, experimentally evaluating the results at the ppb level.

Moreover, ASTRO can work complementary with different existing strategies and mechanisms to assist networked drones missions. For example, in complex environments such as urban areas



(a) IRIS software defined radio (b) miniPID2 VOC gas sensor

Fig. 4. Radio sensing (a) in ASTRO and VOC pollutant sensing (b) in ASTRO+.

that have many obstacles, e.g., tall building, and limited/degraded GPS coverage, drones might need to build a map of the environment (e.g., via onboard lidar sensor) to be able to accurately self-position and perform a mission in that environment. In such cases, ASTRO can be implemented along with **Simultaneous Localization and Mapping (SLAM)** methods in robotics [61, 62]. SLAM can focus on building the map of the environment and self-positioning the drones while ASTRO can focus on the mission objective, with the two systems complementing each other.

4 EXPERIMENTAL EVALUATION

In this section, we first describe the experimental setup and then present results, based on over 1,000 h of test flights, to demonstrate the key features of ASTRO. We first evaluate the search and learn phase, discussing parameters estimation accuracy and computation time based on a number of sensed samples. Next, we evaluate the swarm and track phase, analyzing the impact of K-means clustering on error reduction, the impact of drone formation on localization accuracy and performance in tracking a mobile target. Then, we discuss ASTRO in the context of a hiding target and heterogeneous statistics. Last, we describe key challenges and lessons learned over the past 24 months of test flights.

4.1 Evaluation of the Search and Learn Phase

The first phase after launch, i.e., the search and learn phase, is to find the target while learning about the propagation environment. In the first set of experiments, the tracking target is placed in random locations of the stadium's field. We first evaluate the ability of ASTRO to find the target through repeated experiments with different target locations. The drones are launched from the edge of the field. As described in Section 3.3.1, ASTRO drones coordinate to partition the search space into non-overlapping areas. Figure 5 shows an example during the search and learn phase for a case with three drones. In the experiment, the drones labeled D1 (black), D2 (orange), and D3 (blue) are launched from the edge of the field; they coordinate to partition the search space and each drone surveys its assigned area. In the figure, the person holding a transmitter device serves as the target in this example.

4.1.1 Path-loss Exponent Estimation. In the first stage, each drone independently learns the propagation environment, in particular by estimating parameters such as the path-loss exponent α, defined in Equation (2). As drones proceed in a mission, the path-loss exponent estimation propagates into a drone to target ranging accuracy and, ultimately, to target localization estimation. Therefore, accurate α parameter learning is critical in ASTRO. To evaluate it, we consider *Oracle* scheme and *Random* approach. In the Oracle scheme, we consider that drones already know the position of the target and the propagation environment. That scheme serves as ground truth in

Fig. 5. An illustration of the ASTRO drones during the search and learn phase.

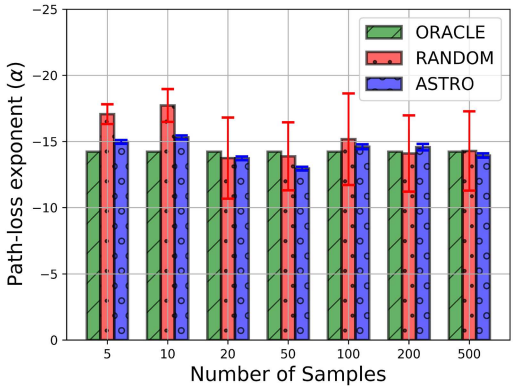


Fig. 6. The path-loss exponent estimation as the number of samples increase.

our evaluation. In the Random approach, contrary to ASTRO, drones do not dedicate computation resources and time to learn α but rather make educated guess depending on the environmental condition of the target. The Random approach helps to analyze the numerical advantages of our proposed learning methods.

As described in Section 3.3.1, the number of samples M in each batch is the key component that impacts the accuracy of the path-loss exponent estimation. In the experiment, each drone observes at least 5 RSSI samples at each spatial point and keeps sensing the target while repositioning in the search and learn phase. In Figure 6, we show the mean and standard deviation of path-loss estimation vs. a varying number of samples for different strategies. Since the Oracle scheme has knowledge of the nearly uniform propagation environment of the football stadium and the target, it consistently indicates α value of around -14.2. In contrast, the Random approach poorly estimates α among all other strategies, although guessing from the limited range of possible α values that correspond to an open space area environment. Because of the lack of learning and adapting based on sensory information, the Random approach also has large uncertainty in the α decision, indicated as a large standard deviation for all of the sample numbers.

Table 1. Expected Computation Time for Learning Parameters Depending on the Computing Platform and Number of Samples

Computing Platform	10 samples	100 samples	1,000 samples	10,000 samples
Raspberry Pi	2.9 s	4.5 s	7.1 s	~1 min
UpBoard	1.5 s	1.7 s	3.0 s	31.2 s
GCS	0.1 s	0.18 s	0.38 s	4.47 s

Unlike the Oracle scheme and Random approach, ASTRO learns and improves path-loss exponent estimation as the number of samples increase in a batch. As shown in the Figure 6, a batch with 50 or less samples is usually insufficient to converge to a consistent α value, fluctuating between overestimation or underestimation depending on the dynamics of the sensory measurements. For example, in this experiment, with 5 samples in a batch ASTRO underestimates path-loss exponent and overestimating it when there are 20 samples. However, as drones fly to more spatial locations and collect more samples, the α estimation improves. The results indicate that, in approximately uniform propagation environment, having more than 50 samples is sufficient to converge to an accurate α estimation. However, increasing the number of samples beyond 100 provides only diminishing marginal improvement as shown in Figure 6.

4.1.2 Computation Time. Because of drones' limited flight duration in mission, it is critical to understand how much time ASTRO drones need to learn the environment and compute parameters such as path loss exponent. For that, the drone solves a batch GD optimization described in Equation (5), where the computation time largely depends on the number of samples in a batch, since the algorithm iterates over every sample per epoch and extracts information from it, with each sample contributing to estimation decision. To evaluate the computation time, we perform experiments with a varying number of samples in a batch, increasing the samples from 10 until 10,000 in multiples of ten. Also, ASTRO is a modular system and can have different computing devices onboard; thus, we execute computation on multiple platforms, namely, on conventional Raspberry Pi and on UpBoard with a more powerful embedded CPU. We also consider the case of offloading the learning parameters to GCS. It provides an additional benchmark of learning time on a desktop PC

Table 1 shows the expected computation time over tens of experiments. Notice that, on Raspberry Pi, it takes around 4.5 s to estimate parameters when there are 100 samples in a batch, while UpBoard needs less than half of that time for the same number of samples. However, for tasks that might require processing samples beyond 1,000, both of these on-drone platforms become computationally costly, for instance, processing time increasing to nearly a minute when there are thousands of samples in a batch. Unlike Raspberry Pi and UpBoard, offloading the task to GCS allows performing estimation in millisecond scale for a number of samples below 1,000 and only 4.47 s for the highest number of samples in the experiment. However, the offloading to GCS also adds the overhead time of transferring data, which will vary depending on the available bandwidth of the channel as well as the size of the transmitted data.

In addition to the number of samples in a batch, there are also generic GD optimization parameters such as learning rate and the maximum number of iterations per epoch. Usually, there are no prior known global optimal values for these parameters [63], but they are rather tuned for a particular application [64], for example, based on the dynamics of the sensory data. In our experiments of on-drone signal-strength sensing, we discover (via a combination of spatio-temporal analysis of experimental RSSI data as well as trial and tuning) that the learning rate of 0.003 and iteration number of 100 is just right, and they also align with the practical recommendation from the

literature [64]. We also perform optimization over multiple epochs to reduce errors and to improve estimation. The aforementioned GD optimization parameters are considered in the computation time results shown in Table 1.

Moreover, in the search and learn phases, each drone independently estimates the location of the target along with path-loss estimation as described in the Equation (4). Note that the localization error in this phase can be as high as several dozens of meters due to lacking multi-lateration from independently working drones. However, drones update and refine their estimation in the next phase (see Section 4.2). The time required for a single drone to find the target and summon the others depends on the size of the area being surveyed, and on average it takes about 12 s during our experiments. Before summoning the other drones, each ASTRO drone estimates the possible location of the target at least two times.

4.2 Evaluation of the Swarm and Track Phase

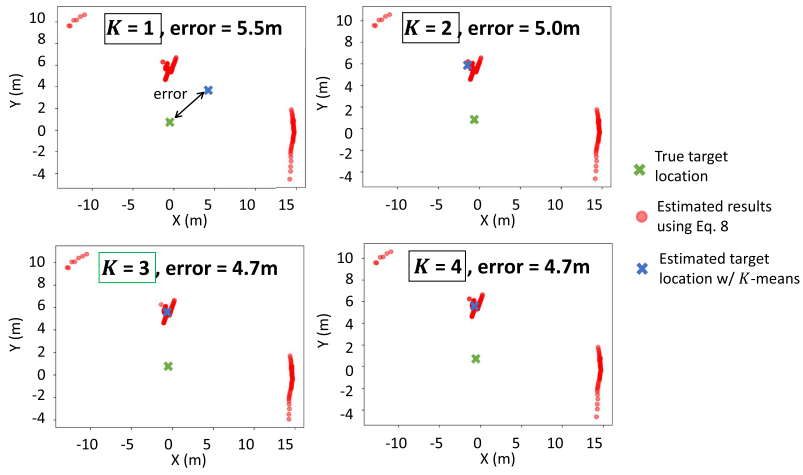
After any drone has identified the target, it informs other drones via the drone-to-drone network to enter the next phase to locate and track the target. In this phase, the drones cooperatively move toward the target. We begin the experiments with a non-mobile target to evaluate the ability of the drones to cooperatively localize it. As described in Section 3.3, as the drones move toward the target, they continue to "learn" and update the parameters obtained in the first phase. Moreover, their estimates are now combined via K-means clustering to provide a single estimate. Any drone (or all drones) can perform this calculation as ASTRO shares all information among all drones.

In this phase, several factors impact the localization accuracy, and the primary one is the number of drones in the network. As described in Reference [4], to find a unique solution of the target using multilateration, a minimum of three drones for 2D tracking and four drones for 3D tracking is needed while additional drones can further improve the tracking accuracy. In Reference [4], we also show that a single drone can only localize a static target mean accuracy of around 50m and two-drone network achieves mean of 25 m accuracy, in both cases large localization error due to lacking multilateration. However, with three drones, the error reduced to sub 10m, and increasing the number of drones beyond three provides only marginal improvement. Based on these findings, further analysis is based on three or more drone missions.

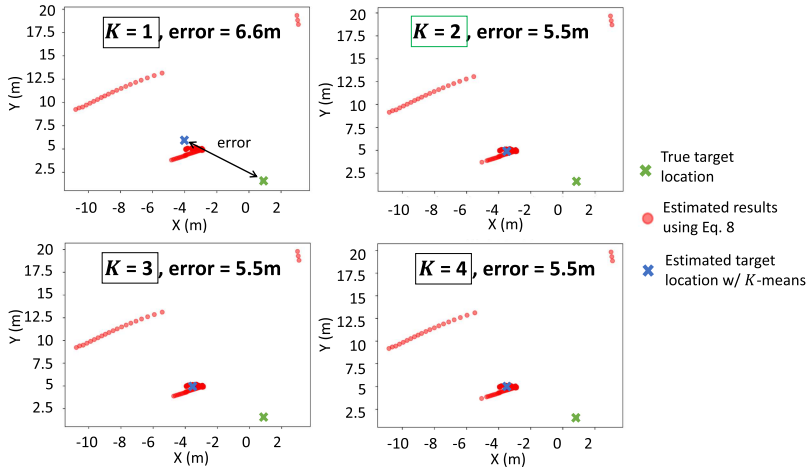
4.2.1 K-means Clustering for Error Reduction. As discussed in Section 3.3.2, ASTRO employs K-means clustering to reduce estimation error as the drones collaboratively localize the target. For example, we observed that using K-means clustering reduces the error from over 15.5 m to less than 10.5 m, resulting in over 30% error reduction in a three-drone mission. Hence, while sufficiently computationally simple to realize on-drone, the employed K-means clustering algorithm provides a significant improvement in the localization performance.

The best value of K, i.e., the one that minimizes localization error, depends on the statistics of the sensed data resulting from the actual environments (i.e., path-loss, fading, shadowing effects, etc.). Figure 7 shows two different cases when the corresponding optimal K value is equal to 3 and 2, respectively. Specifically, for the case corresponding to Figure 7(a), the localization error decreases from 5.5 to 4.7 m when K increases from 1 to 3, and then remains unchanged as K further increases; for the case corresponding to Figure 7(b), which is based on another experimental data set, the localization error decreases from 6.6 to 5.5 m when K increases from 1 to 2, and then stays unchanged as K further increases. While K is ideally set dynamically to adapt to different sensed data "on the fly," here, we consider fixed values.

4.2.2 Impact of Drone Formation on Localization Accuracy. To further improve localization accuracy, it is also important that drones collect independent samples as they fly, essentially observing the target from diverse positions and fusing collective measurements afterward. To study the



(a) The optimal K value is equal to 3

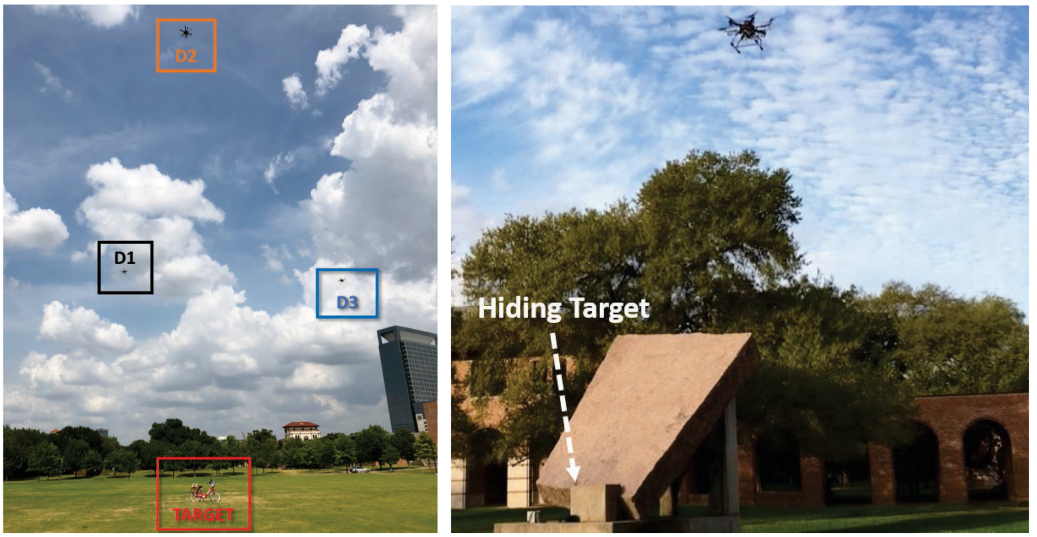


(b) The optimal K value is equal to 2

Fig. 7. A visualization of two different cases when the corresponding optimal K values are equal to 3 and 2, respectively.

impact of sensing independent samples on localization accuracy, we consider two different types of drone formations, namely, circle and orbiting. As the name suggests, in the first one drone fly in a fixed circle around the estimated target while in the second one drones dynamically orbit around the estimated target position. In the first one, we only impose a minimum distance to be maintained between each drone in a circle, therefore drones at each sampling interval collect dependent samples over time. Figure 8(a) shows a picture of an experiment for the case that the drones are in a fixed circle formation. In the orbiting formation, ASTRO drones orbit around the target location, therefore, collecting independent samples over time.

Figure 9 compares the localization error resulting from the two formations. We observe that while the average localization error is nearly 8 m in both cases, orbiting significantly reduces variance due to the spatial smoothing of sample collection. However, orbiting requires overhead



(a) Drones in a circle formation (non-hiding target) (b) Drones localizing the hidden target

Fig. 8. An illustration of (a) ASTRO drones in a circle formation in the swarm and track phase and (b) a ASTRO drone close to the target location, where the target is placed under the inner-most point of a several ton 3.5 m by 6 m slab angled at 45°.

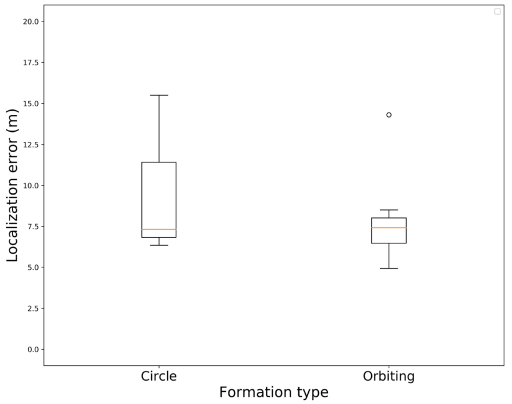


Fig. 9. Impact of ASTRO drones formations on localization accuracy.

to exchange information to keep the drones synchronized. In general, we consider an 8 m error to be adequate for the mission given that (i) the multi-lateration "anchors" (the drones) are mobile and have no *a priori* data on the propagation environment and (ii) while the drone locations are known via GPS, GPS itself has an inherent error.

4.2.3 Tracking a Mobile Target. Thus far, we have considered a static target. Here, we perform experiments with three drones finding and tracking a mobile target. In this case, a human subject walks along the field of the stadium holding the SDR transmitter. He uses the field's hash marks to attempt to maintain an approximate velocity of 2 m/s. The subject turns around when reaching the end of the field. A video recording of an example experiment is also available [65].

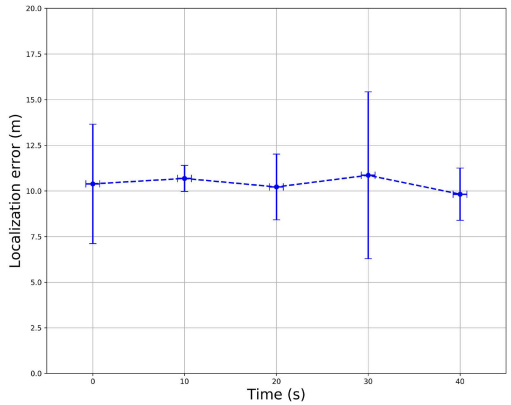


Fig. 10. Localization accuracy over time with a mobile target.

Figure 10 shows localization accuracy over time, based on multiple experiments with a mobile target. Notice that despite that the target's location is changing and the drones must continually move to track it, on average, ASTRO maintains similar error throughout the mission. Specifically, we observed that staying close to the target and roughly in the same formation (typically within 10 m total distance with an altitude of 8 m), signal variations, the machine learning algorithm, and even wind effects, often send a drone several meters "off course." Nonetheless, the prediction error, which combines all drone measurements and incorporates such errors, remains around 10 m.

4.3 Hiding Target and Heterogeneous Statistics

Unlike all the previous experiments where the target was placed in an open football, here we consider a hidden target. Specifically, we place the target in a quad surrounded by buildings and hiding under a several tons, 3.5 by 6 m slab angled at 45°. Figure 8(b) shows a picture of the slab sculpture with a drone in-view while performing a mission. In the experiments, the target is placed under the slab, at the inner-most point of the structure. Thus, the drones must contend with lacking a line-of-sight path to the target and multiple reflections from nearby buildings and the slab, and each can measure different propagation parameters depending on whether they face the opening of the slab.

In these experiments, we utilize three drones and launch each from opposite ends of the quad. Hence, the three drones have very different views of the target. In the experiment corresponding to Figure 11, Drone D1 has a view into the interior of the slab, whereas D2 and D3 do not. After the initial phase, D1 summons the other two drones to enter the swarm and localize phase. The other two drones will subsequently also move to the area where they can receive the signal directly from the opening of the slab. As they move, their model parameters are retrained and the average final localization error for the collaborative localization is 8 m in this scenario. Due to the environmental obstacles (trees, buildings, sculpture, etc.), the drones do not surround the target geographically in a circle, but rather they are all approximately 10 m from the target location toward the opening, due to their objective of minimizing localization error.

4.4 Lessons Learned and Challenges

Over the several years of designing ASTRO and performing thousands of hours of in-the-field test flights, we have learned many significant lessons. Acquired expertise also paved the way for new exciting research, e.g., References [40, 60].

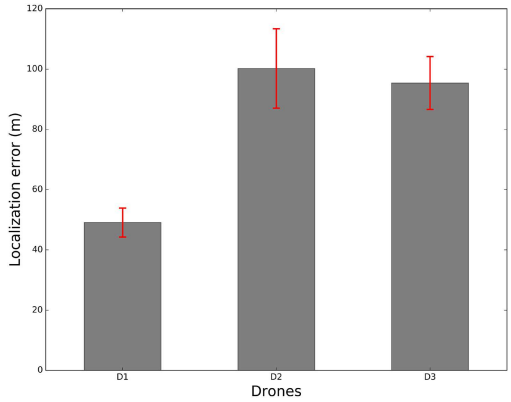


Fig. 11. The per-drone localization error of the three ASTRO drones with a hiding target under the slab *before* collaboration.

Robust Platform. One of the first challenges of ASTRO was the design choice of the platform. The objective was to have a robust platform that will enable control over different components of the system, providing flexibility to explore and implement new research ideas. On one side, commercially available platforms such as Parrot, DJI, and so on, offer stable solutions, but are closed and single-purpose, leaving very limited space for extensions. Therefore, we built a custom system by developing open-source software and open hardware. We also designed it to be modular, which enables easy extension and modification to accommodate different research purposes. In this process, it has been crucial that the selected components are reliable, and at the same time, meet the weight and power budget requirements. Indeed, lower weight improves takeoff/climb and landing performance during a mission. Given a maximum payload of ~1.5 kg and ~500 g of battery, it is crucial to carefully budget the payload. Regarding the power budget, a good rule of thumb when selecting the battery, is to use 1,000 mAH (milliamp hours) per motor.

Maintenance. Another fundamental task regards the maintenance of the multi-drone system. Drones are indeed different from classical tech gadgets that at most require battery checks. Each drone is comprised of many hardware and software components, and they require consistent maintenance and update at different levels. Before *every flight*, it is also essential to check the integrity of the drone frame and propellers, calibration of the sensors (such as Flight Control), and the condition of batteries. Ideally, any malfunctions should be detected before in-the-field flights to avoid crushes. Overall, during our tests, we broke 5 hexa frames, 13 motors, 10 arms, more than 10 pairs of propellers, more than 7 electronic speed controllers, 6 Raspberry Pi, 3 Flight Controllers, and uncountable 3D printed cases and wires.

Safety First. Performing a multi-drone test flight is challenging and requires the utmost precautions. In addition to having a pilot license, at least one person should take responsibility for each drone, holding a corresponding drone controller and being ready to interrupt autonomous flight and land the drone if necessary. Also, weather condition needs to be monitored during the test flight day; while rain might damage electronics of the drone, a strong wind could potentially drift drones to crush to each other (if flying close) or nearby trees/buildings. Above all, when flying multiple drones in a crowded environment, it important to have visual contact with the drones at all times (no flight in the dark) and try to avoid high-speed flight to have sufficient time to react and land the drones if needed.

5 CONCLUSION

We presented the design, implementation, and experimental evaluation of ASTRO, a modular end-to-end system for distributed sensing missions with autonomous networked drones. ASTRO drones can adapt flight patterns in real-time according to sensor data, do not require prior training data, and can use on-board machine learning methods even if all drones lose contact with the ground control. We implement the key components of ASTRO and demonstrate its capabilities through a series of sensing missions to find and track a mobile or hiding target.

REFERENCES

[1] Z. Liu, Y. Chen, B. Liu, C. Cao, and X. Fu. 2014. HAWK: An unmanned mini-helicopter-based aerial wireless kit for localization. *IEEE Trans. Mobile Comput.* 13, 2 (2014), 287–298.

[2] Antonio Loquercio, Ana I. Maqueda, Carlos R. del Blanco, and Davide Scaramuzza. 2018. DroNet: Learning to fly by driving. *IEEE Robot. Autom. Lett.* 3, 2 (2018), 1088–1095. DOI: 10.1109/LRA.2018.2795643

[3] Wenguang Mao, Zaiwei Zhang, Lili Qiu, Jian He, Yuchen Cui, and Sangki Yun. 2017. Indoor follow me drone. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'17)*.

[4] Riccardo Petrolo, Yingyan Lin, and Edward Knightly. 2018. ASTRO: Autonomous, sensing, and tetherless networked drones. In *Proceedings of ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet'18)*.

[5] Luca Mottola and Kamin Whitehouse. 2016. Mobile systems research with drones. *GetMobile: Mobile Comp. and Comm.* 20, 4 (2016).

[6] ArduPilot. [n.d.]. Retrieved from https://github.com/ArduPilot/ardupilot.

[7] Endri Bregu, Nicola Casamassima, Daniel Cantoni, Luca Mottola, and Kamin Whitehouse. 2016. Reactive control of autonomous drones. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'16)*.

[8] Evan Ackerman. 2016. Amazon to test delivery drone autonomy in the U.K. *IEEE Spectrum* (2016). https://spectrum.ieee.org/automaton/robotics/drones/amazon-to-test-delivery-drone-autonomy-in-the-uk.

[9] TopTenDrone.com. [n.d.]. Top 10 Best Drones with Follow-me Mode—2020. Retrieved from http://www.top10drone.com/top-10-drones-follow-me-mode/.

[10] S. Paschall and J. Rose. 2017. Fast, lightweight autonomy through an unknown cluttered environment. In *Proceedings of the IEEE Aerospace Conference*.

[11] dji.com [n.d.]. Matrice 100: Quadcopter for developers. Retrieved from https://www.dji.com/matrice100.

[12] Ashutosh Dhekne, Mahanth Gowda, and Romit Roy Choudhury. 2017. Extending cell tower coverage through drones. In *Proceedings of the International Workshop on Mobile Computing Systems and Applications (HotMobile'17)*.

[13] Jingjing Wang, Chunxiao Jiang, Zhu Han, Yong Ren, Robert G. Maunder, and Lajos Hanzo. 2017. Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones. *IEEE Vehic. Technol. Mag.* 12, 3 (2017), 73–82.

[14] Evsen Yanmaz, Saeed Yahyanejad, Bernhard Rinner, Hermann Hellwagner, and Christian Bettstetter. 2018. Drone networks: Communications, coordination, and sensing. *Ad Hoc Netw.* 68, C (2018), 1–15. Advances in Wireless Communication and Networking for Cooperating Autonomous Systems.

[15] İlker Bekmezci, Ozgur Koray Sahingoz, and Csamil Temel. 2013. Flying ad-hoc networks (FANETs). *Ad Hoc Netw.* 11, 3 (2013), 1254–1270.

[16] Bow-Nan Cheng and Scott Moore. 2012. A comparison of MANET routing protocols on airborne tactical networks. In *Proceedings of the IEEE/AFCEA Military Communications Conference (MILCOM'12)*.

[17] Lin Lin, Qibo Sun, Shangguang Wang, and Fangchun Yang. 2012. A geographic mobility prediction routing protocol for ad hoc UAV Network. In *Proceedings of the IEEE Globecom Workshops*.

[18] Jiyeon Lee, Kangho Kim, Seungho Yoo, Albert Y. Chung, Joon Yeop Lee, Seong Joon Park, and Hwangnam Kim. 2016. Constructing a reliable and fast recoverable network for drones. In *Proceedings of the IEEE International Conference on Communications (ICC'16)*.

[19] Seungho Yoo, Kangho Kim, Jongtack Jung, Albert Y. Chung, Jiyeon Lee, Suk Kyu Lee, Hyung Kyu Lee, and Hwangnam Kim. 2017. Empowering drones' teamwork with airborne network. In *Proceedings of the of IEEE International Conference on Advanced Information Networking and Applications (AINA'17)*.

[20] X. Zhou, Q. Wu, S. Yan, F. Shu, and J. Li. 2019. UAV-enabled secure communications: Joint trajectory and transmit power optimization. *IEEE Trans. Vehic. Technol.* 68, 4 (2019), 4069–4073. DOI: http://dx.doi.org/10.1109/TVT.2019.2900157

[21] Anwen Wang, Xiang Ji, Dajun Wu, Xuedong Bai, Nana Ding, Jing Pang, Shaofeng Chen, Xiaojiang Chen, and Dingyi Fang. 2017. GuideLoc: UAV-assisted multitarget localization system for disaster rescue. *Mobile Info. Syst.* (2017), 1267608:1–1267608:13. DOI: https://doi.org/10.1155/2017/1267608

[22] Andreas Birk, Burkhard Wiggerich, Heiko Bülow, Max Pfingsthorn, and Sören Schwertfeger. 2011. Safety, security, and rescue missions with an unmanned aerial vehicle (UAV). *J. Intell. Robot. Syst.* 64, 1 (2011), 57–76.
[23] J. E. Gomez-Balderas, G. Flores, L. R. García Carrillo, and R. Lozano. 2013. Tracking a ground moving target with a quadrotor using switching control. *J. Intell. Robot. Syst.* 70, 1 (2013), 65–78.
[24] T. Kersnovski, F. Gonzalez, and K. Morton. 2017. A UAV system for autonomous target detection and gas sensing. In *Proceedings of the IEEE Aerospace Conference*.
[25] H. Ahmadi and R. Bouallegue. 2015. RSSI-based localization in wireless sensor networks using regression tree. In *Proceedings of the IEEE International Wireless Communications and Mobile Computing Conference (IWCMC'15)*.
[26] Y. Wang, X. Xu, and X. Tao. 2009. Localization in wireless sensor networks via support vector regression. In *Proceedings of the International Conference on Genetic and Evolutionary Computation (ICGEC'09)*.
[27] L. Gogolak, S. Pletl, and D. Kukolj. 2011. Indoor fingerprint localization in WSN environment based on neural network. In *Proceedings of the IEEE International Symposium on Intelligent Systems and Informatics (SISY'11)*. DOI: http://dx.doi.org/10.1109/SISY.2011.6034340
[28] Jun Zheng and Asghar Dehghani. 2012. Range-free localization in wireless sensor networks with neural network ensembles. *J. Sensor Actuator Netw.* 1, 3 (2012), 254–271. DOI: http://dx.doi.org/10.3390/jsan1030254
[29] I. T. Haque and C. Assi. 2015. Profiling-based indoor localization schemes. *IEEE Systems Journal* 9, 1 (March 2015), 76–85. DOI: http://dx.doi.org/10.1109/JSYST.2013.2281257
[30] J. Proft, J. Suarez, and R. Murphy. 2015. Spectral anomaly detection with machine learning for wilderness search and rescue. In *Proceedings of the IEEE MIT Undergraduate Research Technology Conference (URTC'15)*.
[31] Y. Bazi and F. Melgani. 2018. Convolutional SVM networks for object detection in UAV imagery. *IEEE Trans. Geosci. Remote Sens.* 99 (2018), 1–12. DOI: http://dx.doi.org/10.1109/TGRS.2018.2790926
[32] Y. Yin, X. Wang, D. Xu, F. Liu, Y. Wang, and W. Wu. 2016. Robust visual detection-learning-tracking framework for autonomous aerial refueling of UAVs. *IEEE Trans. Instrument. Measure.* 65, 3 (Mar. 2016), 510–521. DOI: http://dx.doi.org/10.1109/TIM.2015.2509318
[33] J. Chen, X. Miao, H. Jiang, J. Chen, and X. Liu. 2017. Identification of autonomous landing sign for unmanned aerial vehicle based on faster regions with convolutional neural network. In *Proceedings of the Cloud and Autonomic Computing Conference (CAC'17)*.
[34] D. Zhou, J. Zhou, M. Zhang, D. Xiang, and Z. Zhong. 2017. Deep learning for unmanned aerial vehicles landing carrier in different conditions. In *Proceedings of the International Conference on Applied Research in Computer Science and Engineering (ICAR'17)*.
[35] C. Patruno, M. Nitti, A. Petitti, E. Stella, and T. D'Orazio. 2018. A vision-based approach for unmanned aerial vehicle landing. *J. Intell. Robot. Syst.* (Sep. 2018). DOI: http://dx.doi.org/10.1007/s10846-018-0933-2
[36] S. Chung, A. Paranjape, P. Dames, S. Shen, and V. Kumar. 2018. A survey on aerial swarm robotics. *IEEE Trans. Robot.* 34 (4). pp. 837–855.
[37] Stephanie Gil, Swarun Kumar, Dina Katabi, and Daniela Rus. 2016. Adaptive Communication in Multi-robot Systems Using Directionality of Signal Strength. *Robot. Res.* 114 (2016), 57–77.
[38] Manh Duong Phung, Cong Hoang Quach, Tran Hiep Dinh, and Quang Ha. 2017. Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection. *Autom. Construct.* 81 (2017), 25–33.
[39] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng. 2019. When UAV swarm meets edge-cloud computing: The QoS perspective. *IEEE Netw.* 33, 2 (2019), 36–43. DOI: http://dx.doi.org/10.1109/MNET.2019.1800222
[40] Ahmed Boubrima and Edward W. Knightly. 2020. Robust mission planning of UAV networks for environmental sensing. In *Proceedings of ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet'20)*.
[41] Astroserver. [n.d.]. SkyNet software platform, Rice University. Retrieved from http://astroserver.rice.edu:8025/skynet.
[42] Docker. [n.d.]. Retrieved from https://www.docker.com/.
[43] ArduPilot. [n.d.]. Open Autopilot software for drones and other autonomous systems. Retrieved from https://ardupilot.org/.
[44] Panda Wireless. [n.d.]. Panda Wireless PAU06 300 Mbps Wireless N USB Adapter. Retrieved from http://www.pandawireless.com/Specs%20%7C%20Panda%20Wireless.html.
[45] D. Seither, A. Konig, and M. Hollick. 2011. Routing performance of wireless mesh networks: A practical evaluation of BATMAN advanced. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN'11)*.
[46] MAVLink. [n.d.]. Lightweight messaging protocol for communicating with drones and between onboard drone components. Retrieved from https://mavlink.io/en/.
[47] Gustavo S. C. Avellar, Guilherme A. S. Pereira, Luciano C. A. Pimenta, and Paulo Iscold. 2015. Multi-UAV routing for area coverage and remote sensing with minimum time. *Sensors* 15, 11 (2015), 27783–27803.
[48] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. 1999. The anatomy of a context-aware application. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom'99)*.

[49] L. Girod and D. Estrin. 2001. Robust range estimation using acoustic and multimodal sensing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*.
[50] D. Niculescu and Badri Nath. 2003. Ad hoc positioning system (APS) using AoA. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'03)*.
[51] G. A. F. Seber and C. J. Wild. 2003. *Nonlinear Regression*. Wiley.
[52] Ethem Alpaydin. 2010. *Introduction to Machine Learning* (2nd ed.). The MIT Press.
[53] T. S. Rappaport. 2001. *Wireless Communications. Principles and Practice* (2nd ed.). Prentice Hall.
[54] S. J. Julier and J. K. Uhlmann. 2004. Unscented filtering and nonlinear estimation. *Proc. IEEE* 92, 3 (2004), 401–422.
[55] Hazem Sallouha, Alessandro Chiumento, and Sofie Pollin. 2017. Localization in long-range ultra narrow band IoT networks using RSSI. In *Proceedings of the IEEE International Conference on Communications (ICC'17)*. IEEE, 1–6.
[56] Sandy Mahfouz, Farah Mourad-Chehade, Paul Honeine, Joumana Farah, and Hichem Snoussi. 2014. Target tracking using machine learning and Kalman filter in wireless sensor networks. *IEEE Sensors J.* 14, 10 (2014), 3715–3725.
[57] J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*.
[58] Martin Wikelski, Roland W. Kays, N. Jeremy Kasdin, Kasper Thorup, James A. Smith, and George W. Swenson. 2007. Going wild: What a global small-animal tracking system could do for experimental biologists. *J. Exp. Biol.* 210, 2 (2007), 181–186.
[59] Sune Svanberg, Guangyu Zhao, Hao Zhang, Jing Huang, Ming Lian, Tianqi Li, Shiming Zhu, Yiyun Li, Zheng Duan, Huiying Lin, and Katarina Svanberg. 2016. Laser spectroscopy applied to environmental, ecological, food safety, and biomedical research. *Optics Express* 24, 6 (2016), A515–A527.
[60] Zhambyl Shaikhanov, Ahmed Boubrima, and Edward W. Knightly. 2020. Autonomous drone networks for sensing, localizing and approaching RF targets. In *Proceedings of IEEE Vehicular Networking Conference (VNC'20)*.
[61] Patrik Schmuck and Margarita Chli. 2017. Multi-uav collaborative monocular slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'17)*. IEEE, 3863–3870.
[62] Renaud Dubé, Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. 2017. An online multi-robot slam system for 3d lidars. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'17)*. IEEE, 1004–1011.
[63] Russell Reed and Robert J. MarksII. 1999. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press.
[64] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*. Springer, Berlin, 437–478.
[65] Google. [n.d.]. A video from an experiment in which ASTRO drones are in "Swarm and Track" and tracking a mobile target via sensing RSSI. Retrieved from https://drive.google.com/drive/folders/1FG0sKIAp_UB79fxNIhTjoKCbMvRpTzpb?usp=sharing.

Received February 2020; revised November 2021; accepted May 2021