Feasibility Study of Autonomous Drone-based IoT Device Management in Indoor Environments

Michael Haus
Technical University of Munich

Jan Krol
Technical University of Munich

Aaron Yi Ding
Delft University of Technology

Jörg Ott
Technical University of Munich

ABSTRACT

Future computing environments are embedded with many sensors for applications like augmented reality. Much of the deployed Internet of Things (IoT) technology is designed to be invisible. To support user's privacy awareness, a map of surrounding sensing devices is beneficial to determine the nature of data collection taking place in any given area. Moreover, security and governance issues are among the challenges IoT poses to organizations which might not know exactly which IoT devices are connected to their network. For instance, many employees bringing their own devices to the workplace. We explore the feasibility to use small COTS drones to create indoor maps of wireless devices. These comprehensive device maps serve as basis for device localization and monitoring to enhance user privacy and network security. We analyze the impact of our device management platform at the drone's energy consumption and evaluate the device detection rate, explored area, and localization error. Due to the restricted battery capacity of the drone, we simulate larger areas with a varying number of IoT devices to highlight the limits of our drone-based device management platform regarding area exploration and reachable devices.

CCS CONCEPTS

• **Networks** → **Network management**; • **Hardware** → *Wireless devices*; • **Security and privacy** → *Network security*; *Privacy protections*.

KEYWORDS

IoT device management, COTS drones, Indoor mapping, Device localization and monitoring, User privacy, Network security

ACM Reference Format:
Michael Haus, Jan Krol, Aaron Yi Ding, and Jörg Ott. 2019. Feasibility Study of Autonomous Drone-based IoT Device Management in Indoor Environments. In *MAGESys '19: ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications, August 19, 2019, Beijing, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

MAGESys '19, August 19, 2019, Beijing, China

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The term "IoT" is an umbrella keyword covering various aspects related to the extension of the Internet and Web into the physical realm [13]. In 2019, an estimated amount of 27 billion connected IoT devices are deployed worldwide [17] and the effective management of these IoT devices becomes challenging due to the scale of deployments. For example, in a harbor logistic warehouse the port authority and different shipping vendors gradually use wireless tags and sensing kits to trace and log their products for transaction and book keeping purposes. Since the position of those sensing devices are often not well tracked and sometimes moved incidentally by maintainers, we need an automatic mechanism to detect and trace them. Moreover, the lack of awareness about spatially distributed IoT assets both limits the services that can be provided and raises concerns with respect to user privacy and system security.

Security and governance issues are among the challenges IoT poses to organizations stemming from the widespread adoption of IoT devices, their diversity, standardization obstacles, and inherent mobility. For instance, smart cameras and smoke detectors enhance security; smart thermostats, smart light bulbs and sockets facilitate power savings; and so forth. Organizations might not know exactly which IoT devices are connected to their network, particularly caused by the trend that employees bringing their own IoT devices (BYOIoT) to the workplace. For example, with the use of wearables in the healthcare and business service/consulting industries. Surveying this BYOIoT trend, also 25–50 % of remote employees connected at least one IoT device to the enterprise network [4]. A situation which threatens the security and integrity of the network and the devices. To support user's privacy awareness, a map of surrounding sensing devices is beneficial to determine the nature of data collection taking place in any given area. It does not protect users against deliberate covert surveillance, but we are able to inform users about the data capture upon approaching a region.

With mapped IoT devices, an up-to-date overview of all distributed devices including their locations and capabilities, we can fully harness IoT deployments while avoiding potential threats in terms of security concerns and user privacy. We build upon our ground work [6, 7] by developing new modules that are dedicated for drone-based IoT device management. This further reduces operational costs to be independent of users and able to autonomously gather data for device maps. We explore the feasibility of using small COTS drones to create indoor maps which consist of Wi-Fi and Bluetooth Low Energy (BLE) devices. These comprehensive device maps serve as basis for device localization and monitoring to enhance network security and user privacy. Via repeated drone flights we can perform device presence detection and our indoor device mapping is able to identify new devices and track changes in the environment.

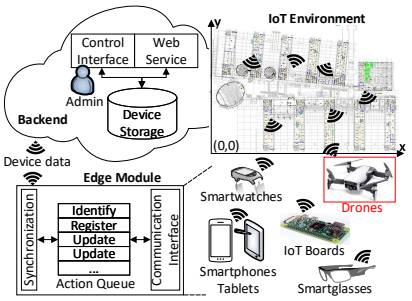
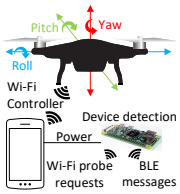


Figure 1: Platform overview for IoT device management. The edge modules run at multiple end-devices which are static user-independent (IoT boards) and mobile user-dependent (smartwatches, smartglasses, smartphones). We focus on drones as mobile user-independent end-devices.

(a) Assembly view



(b) Schematic view

Figure 2: Platform for autonomous device management

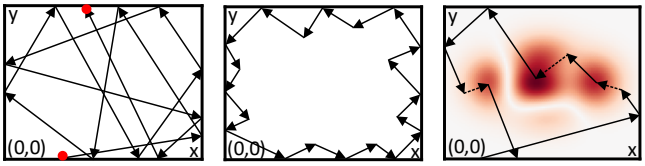
Our contributions are summarized as follows:

(1) We realize a drone-enabled IoT device management to autonomously create indoor maps containing wireless devices. This serves the device monitoring and localization improving user privacy and network security.
(2) We analyze the impact of our device management platform at the drone's energy consumption. Moreover, we identify the best working area exploration strategy in terms of explored area over time, device detection rate, and localization error.
(3) To unwind the restricted battery capacity of the drone, we simulate larger areas with different number of IoT devices to optimize the flight path reaching more devices. After the initial area exploration the device positions are known and we are able to tweak the flight path of the drone.

2 PLATFORM FOR DEVICE MANAGEMENT

To enrich our platform in Fig. 1, we build upon our ground work [6, 7] by developing new modules that are dedicated for drone-based IoT device management. The drone has to be as easy as possible to control autonomously, small enough to fly indoors, and strong enough to carry a smartphone for on-board control and an IoT board for device detection. We use the DJI Mavic Air to analyze the feasibility of using COTS drones for indoor device management.

Drone platform assembly To achieve fully autonomous area exploration, our drone controller and device detection platform flies with the drone as shown in Fig. 2(a), avoiding user involvement and dependency to a ground-control station. The schematic view in Fig. 2(b) shows the smartphone controlling the drone and powering



(a) Random direction **(b) Follow boundary** **(c) Hybrid**

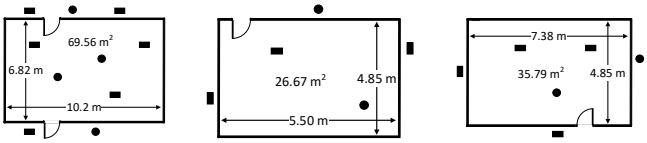
Figure 3: Strategies for area exploration

the Raspberry Pi Zero W. The Raspberry Pi performs the device detection by receiving Wi-Fi probe requests and BLE messages and starts it automatically at startup. Our drone flies at an altitude of 1.8 m to more easily explore areas by avoiding spatial barriers such as chairs, tables. Inspired by the reactive control of [3] at which control decisions are taken only upon observed changes, we actively control the flight of the drone only in case of encountered obstacles. We take advantage of the drone's API and move the drone via virtual sticks which are translated into movement.

Area exploration We have implemented two different area explorations: random direction and boundary following. These are sufficiently simple to realize and fulfill our target to explore the area and recognize devices. With the control mode random direction as illustrated in Fig. 3(a) (start and end position highlighted with red dots), the drone flies up to an obstacle like a wall and randomly chooses another direction until no obstacle blocks the drone. Fig. 3(b) shows the principal working flow of boundary following, the drone flies in its heading until it recognizes an obstacle, then it turns right and left to follow the boundary as closely as possible. For future work, the hybrid control mode in Fig. 3(c) applies by default the random direction strategy. When during movement the drone detects an increasing signal strength, we allow the drone to follow this signal (dotted line). At the peak of the signal strength the drone falls back to random direction control mode.

Device detection For the device detection of Wi-Fi devices we use the Nexmon firmware patch to enable the Wi-Fi monitor mode at the Raspberry Pi Zero W. On this basis, tcpdump collects Wi-Fi probe requests of surrounding devices and concurrently we perform channel hopping to find as many Wi-Fi devices as possible. The discovery of BLE devices occurs via a Python script receiving the messages. The device mapping of all Wi-Fi and BLE devices includes list entries with timestamp, MAC address, and received signal strength indicator (RSSI).

Relative positioning To calculate a relative position of each encountered device and be able to generate device maps, we define a relative coordinate system for our university building. Each room has a reference frame with an origin at which the drone starts exploring the room. From this start position, the drone estimates its own position (x, y) via time, gyroscope (direction), and velocity. Before the flight we perform a time synchronization between smartphone and Raspberry Pi to later identify the device locations via matched log timestamps. During each second of the flight, the smartphone logs the current relative position and time. After the drone's flight we are post processing the gathered data and for each device we select encounters with strongest RSSI, i.e., at this time the drone was nearest to the wireless device. As result, we create



(a) Show room: 6 x Wi-Fi, 4 x BLE **(b) Meeting room: 3 x Wi-Fi, 2 x BLE** **(c) Office room: 4 x Wi-Fi, 2 x BLE**

Figure 4: Test environments for autonomous area exploration of ●: BLE devices and ▬: Wi-Fi devices

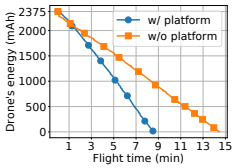


Figure 5: Impact of device management platform at the drone's energy drain

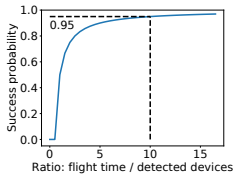


Figure 6: Asses the area exploration of unknown territory (success probability)

a device list with MAC address, relative location (x, y), time, and signal strength.

Limitations Our device detection is limited to active wireless devices, entirely passive devices cannot be recognized. Besides that, the drone's obstacle avoidance restricts indoor area exploration to medium- and large-sized areas. In our experiments, we found that the drone maintains a safety distance of 1.8 m to obstacles, mainly motivated by outdoor usage and weather conditions like wind. Hence, it is not possible to move the drone in small rooms, e.g., $\leq 12\,m^2$, or corridors with a width of two or three meters.

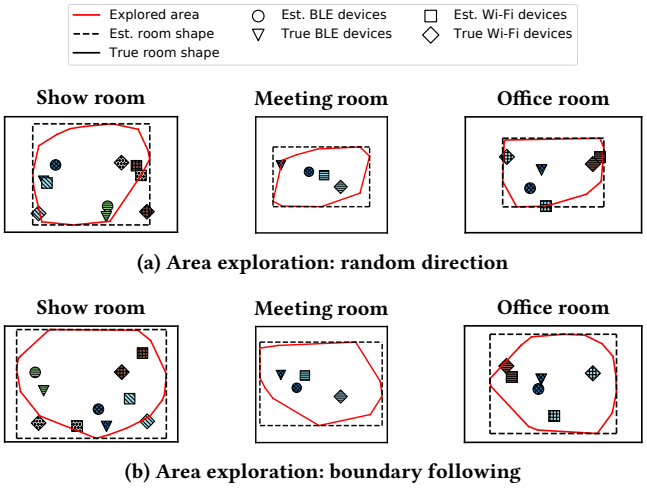
3 EVALUATION

We analyze our drone-based device management regarding initial area exploration to create precise device maps. Thereby, we evaluate the device detection rate, explored area, and localization error over time. After the initial area exploration, we known the device positions to optimize the flight route of the drone to reach more devices within a limited flight time with one battery load. Therefore, we simulate larger areas with a varying number of devices to show the impact of path generation algorithms and hot spots in terms of maximum reachable devices. By hovering at a hot spot, the drone is able to reach multiple IoT devices at once which is more efficient than to fly to each device individually.

3.1 Device Management Platform in Testbed

We have chosen three real-world test environments as shown in Fig. 4 to evaluate our indoor device management platform including explored area, device detection rate, and localization error. We placed a varying number of Raspberry PIs acting as BLE or Wi-Fi device inside and outside of each room representing smartwatches, printers, BLE beacons, IoT sensor boards, and so forth. For an useful localization it is most important to distinguish between devices located inside and outside of the room. Only in larger areas like in our simulation with the university hall (Section 3.2) the device position inside the area gains importance.

Energy consumption The main impact of our device management platform at the theoretical maximum flight time of 21 min is the additional weight of the smartphone with 143 g to control the drone and the Raspberry Pi Zero W with 9 g to gather wireless information. Fig. 5 shows the energy consumption over time with and without our platform: the battery drains at -164.18 mAh/min without our platform which results in a maximum flight time of 14.45 min. In contrast, with our device management platform, the battery drain increases to -277.08 mAh/min which results in a decreased flight time of 8.51 min, a decrease of 41.1 %.



(a) Area exploration: random direction

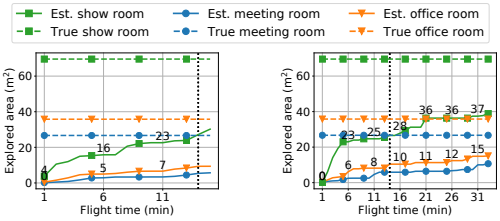
(b) Area exploration: boundary following

Figure 7: Maps of indoor area exploration

Device map For a qualitative evaluation, Fig. 7(a) and Fig. 7(b) show the device mapping for each testbed with identified devices inside the room and their true and estimated positions (for each device an own color and hatch). The area explorations are performing well independent of the room size, most device positions are estimated closely to the true position. The outer relative coordinates from the drone define the convex hull illustrated as explored area. Based on the knowledge that room shapes are mostly rectangular, we take the maximum coordinates from the convex hull in each direction, and lines through these points define the room boundary.

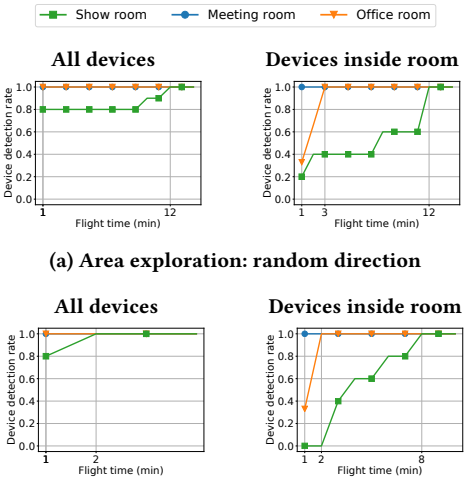
Explored area We evaluate the area explorations illustrated in Fig. 3 where we implemented the random direction and boundary following. Fig. 8 presents the explored area over time compared to the true area of the testbed. The explored area corresponds qualitatively to the actual room size. On average, random direction discovers 0.94 m^2/min and boundary following obtains 0.66 m^2/min. We take the same point in time (14 min) to compare the explored area over time (highlighted via a dotted line). Random direction explores 15.09 m^2 covering 30.32 % of the true room area. On the other hand, the area exploration using the boundary following results in 14.28 m^2 covering 29.76 % of the true area. In general, our area exploration is too slow to discover a reasonable area size within the limited flight time of one battery load.

Device detection We classify devices to be within the room based on the assumption that devices inside the room, nearby the



(a) Random direction (b) Boundary following

Figure 8: Experimental evaluation of explored area



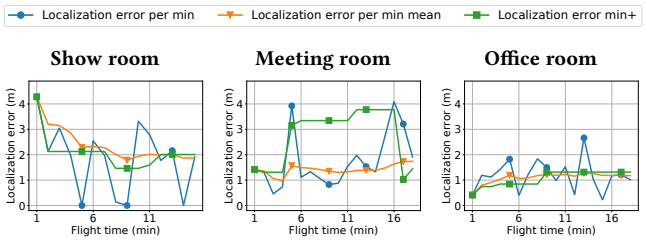
(a) Area exploration: random direction

(b) Area exploration: boundary following

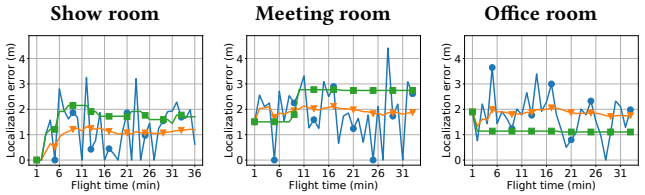
Figure 9: Experimental evaluation of device detection

drone without an obstacle in between, achieve a stronger signal strength compared to devices outside of the room. Over three test environments with varying room size, we experimentally identified RSSI thresholds of -45 dB for Wi-Fi devices and -50 dB for BLE devices to be able to distinguish devices inside and outside of the room. For each testbed, Fig. 9 shows the accuracy and duration until all devices (inside and outside of the room) are recognized and classified to be inside the room. To detect all devices inside and outside of the room, the flight mode with boundary following takes on average 1.33 min compared to a 3.5 times increase of 4.67 min using random direction. In addition, to classify the devices to be within the room, the random direction control mode (5.33 min) lasts 1.5 times longer than the boundary following (3.67 min). The area exploration using boundary following is faster compared to random direction, which allows a good device detection within restricted flight time.

Localization error For each device we take the device position(s) with the strongest signal strength and compute the average localization error over all of them. We calculate three different localization errors by varying the time scale of the input data. First, we compute a localization error for each data collected over one minute. Second, we average the localization errors over data collected within one minute. Third, we compute localization errors by taking only



(a) Area exploration: random direction



(b) Area exploration: boundary following

Figure 10: Experimental evaluation of localization error

one device position with strongest signal strength for each device from current and previous data collected over one minute time intervals (min+). The localization error over time highlights the varying RSSI precision. The mean localization error achieves the smallest positioning error. Using the boundary following control, the localization error results on average in 1.59 m ± 0.19 m compared to the random direction with 1.67 m ± 0.35 m. The second most precise device positions are computed over minute intervals of gathered data. The flight mode using random direction gains a localization error of 1.6 m ± 0.96 m which is similar to the boundary following of 1.61 m ± 0.91 m. With more collected data to estimate device positions, the computed localization error increases. The boundary following control mode results on average in a smaller localization error of 1.74 m ± 0.38 m compared to 1.97 m ± 0.66 m using random direction.

Explore unknown territory Usually we explore areas with an unknown number of IoT devices. To assess when we are done exploring an unknown territory, we are averaging our results for the device detection rate and flight time over different area explorations and testbeds to compute a success probability. Fig. 6 shows whether the area is well enough explored depending on the ratio of flight time and number of detected devices. This means most devices are found and we can classify them to be inside the room, and the localization error is around 2 m.

Summing up, the indoor area exploration using boundary following achieves a superior performance compared to random direction. Boundary following is faster to detect and classify devices and obtains a smaller localization error, the size of the explored area is similar to random direction.

3.2 Optimizing Device Coverage in Simulation

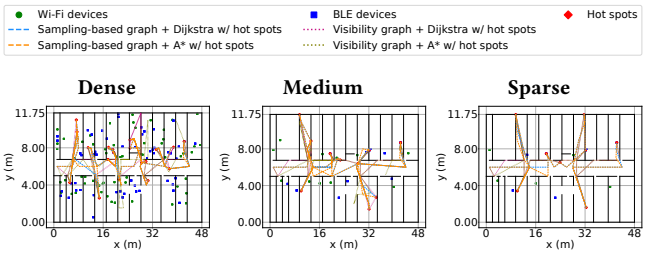
In terms of reachable IoT devices, the restricted battery capacity of the drone is the most severe limitation for our device management platform. After the initial area exploration the device positions are known. On this basis, we simulate larger areas with different

number of IoT devices to optimize the flight path of the drone to reach more devices with one drone's battery load.

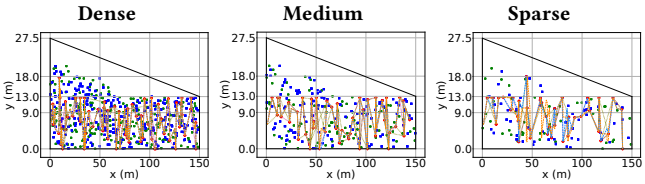
Simulation settings As prerequisite for the simulation, we have measured the drone's velocity and energy drain rate during the flights in our real-world test environments similar to the simulation environments in terms of spatial arrangement. In the university lab with more obstacles the drone achieves a velocity of 0.9 m/s in comparison to 1.5 m/s in the university hall as one large room with fewer spatial barriers. We set the energy limit of the drone to ≥ 35 %, i.e., with the remaining energy the drone is able to fly back to a predefined position to change or recharge the battery. Based on a detailed building map, we have modeled two different simulation environments, our university lab with 564 m^2 and the university hall with 3038 m^2. Fig. 11 highlights our simulation for one exemplary device distribution (we omit the flight paths without hot spots for readability). We use a random device distribution with different number of devices ranging from dense, medium to sparse. For an area of 25 m^2 which matches to a single room, we randomly distribute five devices (dense), two devices (medium), and one device (sparse). Each device is either a Wi-Fi or BLE device and we randomly select a wireless range of [5, 15] m for Wi-Fi devices and [1, 7] m for BLE devices. For each IoT device we simulate a maintenance task by a random waiting time between 5 to 10 s. Table 1 presents the number of devices, the determined hot spots and their ratio for all simulation environments. With more devices, the number of hot spots are increasing while the ratio to the total number of devices is decreasing. With less devices, multiple hot spots are covering only one device.

Exemplary simulation results For our exemplary simulations in Fig. 11, the visibility graph applying Dijkstra path planning performs best with a median distance of 200.1 m in case with hot spots and 489.3 m without hot spots. The visibility graph with A* path planning obtains on average a longer flight route of 176.1 m with hot spots and 600.5 m without hot spots. The sampling-based graph using A* or Dijkstra path planning achieve similar results of 216.3 m with hot spots and 568.9 m without hot spots. The hot spots save on average 61 % of the flight distance.

Path generation To compute the flight path of the drone, we generate two different graphs of IoT devices: sampling-based graph and a visibility graph using the map data from our simulations like in Fig. 11. After the graph construction representing the simulation environment, we apply common Dijkstra and A* path planning to find the shortest flight path of the drone visiting IoT devices. We compare four different path generation approaches: sampling-based graph using Dijkstra and A* path planning, and visibility graph with Dijkstra and A* path planning. Our simulation results over ten rounds in Fig. 12 show the discovered devices by the drone with one battery load compared to the total number of devices (Table 1). Over all simulation runs, the drone's energy limit is 39.5 % ± 2.61 % of the battery capacity with 2375 mAh (≈ 938 mAh). This lowers the average flight time to 303.84 s ± 26.15 s compared to 464.77 s ± 87.73 s without an energy limit leading to a decrease of 34.63 %. The target of our simulation is to find the most efficient path generation, i.e., reach a maximum number of devices by a minimum amount of explored area. Therefore, we consider the ratio between discovered devices (%) and explored area (%) to highlight the best performing path planning for each environment and device distribution. Over



(a) Simulation: university lab (area: 564 m^2, drone velocity: 0.9 m/s)



(b) Simulation: university hall (area: 3038 m^2, drone velocity: 1.5 m/s)

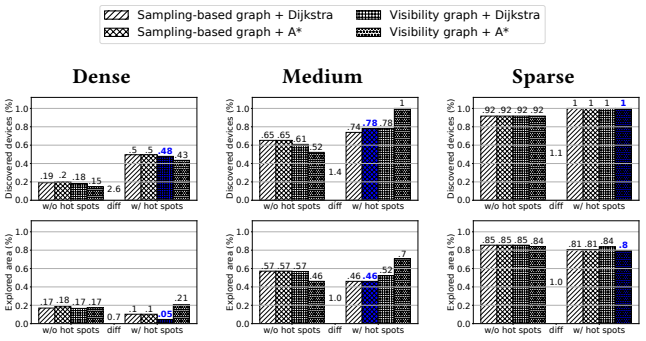
Figure 11: Visualization of exemplary simulation runs

Table 1: Simulation details with number of devices, hot spots, and their ratio over different device distributions

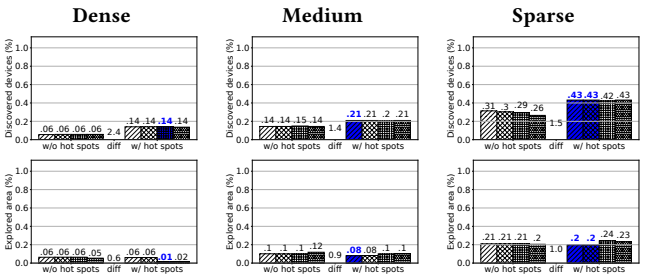
Simulation	University lab			University hall		
Devices	Dense	Mid	Sparse	Dense	Mid	Sparse
Wi-Fi	59	11	5	233	93	46
BLE	56	12	7	287	122	56
Total	115	23	12	520	215	102
Hot spots	19	10	8	65	46	39
Ratio (%)	16.5	43.5	66.7	12.5	21.4	38.2

all simulation runs, the sampling-based graph applying Dijkstra path planning performs best. In comparison, the sampling-based graph using A* achieves a median ratio of 98.55 % of discovered devices and explored area. This is similar to the median ratio of 98.35 % using the visibility graph with Dijkstra path planning. With a median ratio of 87.11 % of discovered devices and explored area the visibility graph applying A* path planning performs worst. The Dijkstra path planning finds mostly the best path even without utilizing a search heuristic as used by A* path planning.

Hot spots To further optimize the flight path and minimizing the flight time, we compute so-called hot spots at which the drone can reach as many devices as possible without movement. We calculate the intersection points among the wireless ranges of all devices to find the positions of the hot spots. The difference factor in Fig. 12 shows the average increase in discovered devices and decrease in explored area. The effect of the hot spots diminishes from a dense to a sparse device distribution. In case of a dense distribution of devices, we achieve the highest impact with an increase of discovered devices by 2.6 (university lab) and 2.4 (university hall). There is no significant performance difference between our two test environments. The university lab consists of 23 rooms with an



(a) Simulation: university lab



(b) Simulation: university hall

Figure 12: Simulation results of two test environments, we highlight the best performing path generation regarding most discovered devices and least explored area

average room size of 25 m^2 and many spatial barriers such as walls in contrast to the university hall as one large room with 3038 m^2.

Over all simulations it is not possible to reach all devices even in case of hot spots with one battery load and an energy limit of 35 % to safely fly back to a predefined position. Only in the university lab with a sparse and medium number of distributed devices we are able to discover all IoT devices. The university hall is too large to be entirely discovered with one battery load. Over all simulations, hot spots increase the number of reachable devices by 39.76 %.

4 USE CASES: PRIVACY AND SECURITY

Our central backend for IoT device management in Fig. 1 provides add-on services via a global map of locally installed wireless devices for enhanced user privacy and network security. The drones as mobile user-independent end-devices identify local devices and transmit this information to our device management backend. To improve user privacy, we synchronize MAC addresses, device positions and device types (Wi-Fi or BLE) to end user devices. Combined with indoor localization at the user device we can inform users about nearby wireless sensing devices which potentially infringing their privacy. Regarding enhanced network security, we are able to distinguish between remote and local wireless devices by comparing network scans with our indoor maps via included MAC addresses. Hence, network administrators can more easily recognize unauthorized wireless devices and via white lists of network devices we can discover locally installed malicious wireless devices.

5 RELATED WORK

Existing work [1, 2, 10, 14, 15] mainly focus on specialized drones, control of the drone, or using energy consuming and computation heavy vision-based data to create indoor maps. In contrast, our device management platform only analyzes the wireless signals to localize devices for indoor mapping.

Users are unaware of the locations and purposes of IoT devices which are infringing their privacy by sensing data in the background. Hence, it is important to catalog wireless devices with detailed information about the locations and capabilities of devices to support awareness about privacy and security concerns [16]. There is a lot of work [8, 11] analyzing network traffic by applying machine learning models like random forest or convolution neural network to identify devices within the network, their device type, and detect anomalous deviations in communication patterns [18]. In our work, we are passively sensing network traffic of Wi-Fi and BLE devices to detect their presence and from the received signal strength (RSSI) we infer the location of the devices.

Maps of surrounding devices can be generated from three sources: 1) authorities, 2) users, and 3) data provided by the infrastructure. Data from the infrastructure removes the need of human effort. For instance, analysis of electric signals to identify home appliances [5], or using network traffic to automatically detect IoT devices [12]. In our case, we create indoor maps together with installed devices and their locations. We need a moving data collector to gather required data and be independent of humans to ensure a uniform quality of the device maps. We have chosen a drone to be most suitable for our intended purpose which can move more freely within indoor areas compared to robots moving on the ground [9] and obstacles on the way like tables, chairs, and staircases.

6 CONCLUSION

By our control design of the drone, the control unit, e.g., smartphone, flies with the drone, we are independent of any ground control station. The drone's battery capacity is the main limiting factor for automated area exploration. In our real-world testbed, we found that the flight control using boundary following for indoor area exploration works best in terms of faster device detection and smaller localization error compared to the random direction strategy. Our simulation revealed that the sampling-based graph applying Dijkstra path planning achieves the best path generation in terms of most discovered devices and least explored area. Hot spots which cover multiple devices at once are significantly improving the number of discovered devices. However, it is still not possible to reach all devices distributed over the entire space with one drone's battery load.

For future work, we plan to implement the hybrid area exploration as described in Fig. 3(c) to gain further insights about the drone's flight path and precision of estimated device locations. To reduce the time to explore areas, we can simultaneously fly multiple drones in different areas and merge the gathered data to generate an overall device map. Besides that, we seek to avoid the additional weight of the smartphone to control the drone by moving the control application to the Raspberry Pi Zero W which is currently only gathering the wireless data like Wi-Fi probe requests and BLE messages for device detection.

REFERENCES

[1] Pompilio Araujo, Rodolfo Miranda, Diedre Carmo, Raul Alves, and Luciano Oliveira. 2017. Air-SSLAM: A Visual Stereo Indoor SLAM for Aerial Quadrotors. *IEEE Geoscience and Remote Sensing Letters* 14, 9 (2017), 1643–1647.

[2] Yingcai Bi, Hailong Qin, Mo Shan, Jiaxin Li, Wenqi Liu, Menglu Lan, and Ben M. Chen. 2016. An Autonomous Quadrotor for Indoor Exploration with Laser Scanner and Depth Camera. In *Proceedings of the 12th IEEE International Conference on Control and Automation (ICCA)*. 50–55.

[3] Endri Bregu, Nicola Casamassima, Daniel Cantoni, Luca Mottola, and Kamin Whitehouse. 2016. Reactive Control of Autonomous Drones. In *Proceedings of the 14th International Conference on Mobile Systems, Applications and Services (MobiSys)*. 207–219.

[4] Calero. 2015. 3 Ways the Internet of Things will Impact Enterprise Security. https://www.calero.com/mobility-service-support/3-ways-the-internet-of-things-will-impact-enterprise-security/

[5] Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. 2010. ElectriSense: Single-Point Sensing Using EMI for Electrical Event Detection and Classification in the Home. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (UbiComp)*. 139–148.

[6] Michael Haus, Aaron Yi Ding, Pan Hui, and Jörg Ott. 2017. Demo: iConfig - What I See is What I Configure. In *Proceedings of the 12th ACM Workshop on Challenged Networks (CHANTS)*. 1–2.

[7] Michael Haus, Aaron Yi Ding, and Jörg Ott. 2017. Managing IoT at the Edge: The Case for BLE Beacons. In *Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects*. 41–46.

[8] Lei Bai, Lina Yao, Salil S. Kanhere, Xianzhi Wang, and Zheng Yang. 2018. Automatic Device Classification from Network Traffic Streams of Internet of Things. *CoRR* abs/1812.09882 (2018).

[9] Ren C. Luo and Chun C. Lai. 2012. Enriched Indoor Map Construction Based on Multisensor Fusion Approach for Intelligent Service Robot. *IEEE Transactions on Industrial Electronics* 59, 8 (2012), 3135–3145.

[10] Aravindh Mahendran, Ayush Dewan, Nikhil Soni, and K. Madhava Krishna. 2013. UGV-MAV Collaboration for Augmented 2D Maps. In *Proceedings of Conference on Advances In Robotics (AIR)*. 1–6.

[11] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. 2017. ProfilIoT: a Machine Learning Approach for IoT Device Identification Based on Network Traffic Analysis. In *Proceedings of the Symposium on Applied Computing (SAC)*. 506–509.

[12] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems (ICDCS)*. 2177–2184.

[13] Daniele Miorandi, Sabrina Sicari, Francesco de Pellegrini, and Imrich Chlamtac. 2012. Internet of Things: Vision, Applications and Research Challenges. *Ad Hoc Networks* 10, 7 (2012), 1497–1516.

[14] Guillaume Pepe, Massimo Satler, and Paolo Tripicchio. 2015. Autonomous Exploration of Indoor Environments with a Micro-Aerial Vehicle. In *Proceedings of the Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. 43–52.

[15] Arindam Saha, Soumyadip Maity, and Brojeshwar Bhowmick. 2018. Indoor Dense Depth Map at Drone Hovering. In *Proceedings of the 25th IEEE International Conference on Image Processing (ICIP)*. 96–100.

[16] Peter Shaw, Mateusz Mikusz, Petteri Nurmi, and Nigel Davies. 2019. IoT Maps: Charting the Internet of Things. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications (HotMobile)*. 105–110.

[17] Statista. 2019. Internet of Things (IoT): Number of Connected Devices Worldwide from 2015 to 2025 (in Billions). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[18] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Minh Hoang Dang, N. Asokan, and Ahmad-Reza Sadeghi. 2018. DÏoT: A Crowdsourced Self-learning Approach for Detecting Compromised IoT Devices. *CoRR* abs/1804.07474 (2018).