IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 April 2025, accepted 22 May 2025, date of publication 28 May 2025, date of current version 11 June 2025.

*Digital Object Identifier 10.1109/ACCESS.2025.3574556*

RESEARCH ARTICLE

# Ultra-Lightweight Collaborative SLAM for Robot Swarms

**VLAD NICULESCU[1], TOMMASO POLONELLI[1], (Senior Member, IEEE), MICHELE MAGNO[1], (Senior Member, IEEE), AND LUCA BENINI[1,2], (Fellow, IEEE)**

[1]Department of Information Technology and Electrical Engineering (D-ITET), ETH Zürich, 8092 Zürich, Switzerland
[2]Department of Electrical, Electronic, and Information Engineering (DEI), University of Bologna, 40136 Bologna, Italy

Corresponding author: Tommaso Polonelli (tommaso.polonelli@pbl.ee.ethz.ch)

This work was supported in part by the BRAINSEE Project funded by the Armasuisse Science and Technology of the Swiss Confederation under Grant 8003528831, and in part by the European Space Agency (ESA)-Open Space Innovation Platform (OSIP) under Grant I-2023-03429.

**ABSTRACT** A key requirement in robotics is the ability to simultaneously self-localize and map a previously unknown environment, relying primarily on onboard sensing and computation. Achieving fully onboard accurate simultaneous localization and mapping (SLAM) is feasible for high-end robotic platforms, whereas small and inexpensive robots face challenges due to constrained hardware, therefore frequently resorting to external infrastructure for sensing and computation. The challenge is further exacerbated in swarms of robots, where coordination, scalability, and latency are crucial concerns. This work introduces a decentralized and lightweight collaborative SLAM approach that enables mapping on virtually any robot, even those equipped with low-cost hardware and only 1.5 MB of memory, including miniaturized insect-size devices. Moreover, the proposed solution supports large swarm formations with the capability to coordinate hundreds of agents. To substantiate our claims, we have successfully implemented collaborative SLAM on centimeter-size drones weighing 46 g. Remarkably, we achieve a mapping accuracy below 30 cm, a result comparable to high-end state-of-the-art solutions while reducing the cost, memory, and computation requirements by two orders of magnitude. Our approach is innovative in three main aspects. First, it enables onboard infrastructure-less collaborative mapping with a lightweight and cost-effective ($20) solution in terms of sensing and computation. Second, we optimize the data traffic within the swarm to support hundreds of cooperative agents using standard wireless protocols such as ultra-wideband (UWB), Bluetooth, or WiFi. Last, we implement a distributed swarm coordination policy to decrease mapping latency and enhance accuracy.

**INDEX TERMS** Collaborative SLAM, mapping, nano-drone, UAV, swarm.

## SUPPLEMENTARY MATERIAL

Supplementary video at https://youtu.be/uh-Iys90agU
Project's code at https://github.com/ETH-PBL/Nano-C-SLAM

## I. INTRODUCTION

Nowadays, swarms of autonomous robots find applications in many sectors, from industry to civil markets, including biomedical and healthcare [1]. Key tasks such as perception or mapping can be carried out more effectively and at lower latency by a swarm than by a single agent [2]. However, the design of a collaboration scheme between the agents of a swarm is still an unsolved challenge in many robotics applications [3], [4]. The core principle of swarm coordination relies on a set of shared rules based on local-global sensory inputs and communication with neighboring agents. In addition, not relying on centralized processing is crucial for increasing robustness, ensuring that the failure of a single robot does not compromise the entire mission execution [5], [6], [7]. Moreover, achieving autonomous and coordinated robot navigation in real-world environments poses significant scalability challenges in designing a shared coordination scheme that is feasible for hundreds of cooperative agents [8].

The associate editor coordinating the review of this manuscript and approving it for publication was Angelo Trotta.

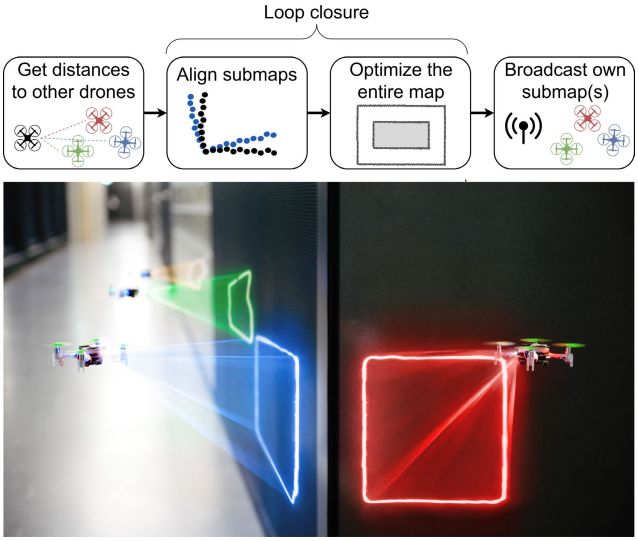



**FIGURE 1.** Illustration of a swarm of miniaturized UAVs using our C-SLAM system to map a real environment.

### A. TECHNICAL CHALLENGES FOR INFRASTRUCTURELESS ROBOT SWARMS

Aerial light shows featuring hundreds of quadcopters and the concept of highly automated warehouses predominantly relying on robotic sorters underscore the current technological landscape [9], [10], [11]. However, those systems mostly follow preprogrammed trajectories calculated on a base station [12], often relying on external infrastructure for computation, localization, and communication [8], [13]. Navigation and cooperation of a swarm of robots in unexplored and GNSS-denied environments, without the support of any external framework, remains an open research challenge [5], [14]. In particular, the design of an efficient and robust infrastructure-free and decentralized solution for agile collaborative robots, supporting navigation, distributed sensing, and mapping is a challenging research goal in the robotics field [15], [16], [17], [18]. The ongoing industrial and social revolution is driving the deployment of robots in everyday life, beyond heavily controlled environments, such as automated production lines [19]. Therefore, heterogeneous and cost-effective robotic platforms will soon need to operate in a variety of environments, often uncontrolled, in proximity to humans, animals, and within residential spaces [5], [20]. In this context, enabling the use of robots across various applications, reducing the hardware costs associated with sensing and computation, is crucial [21]. Undoubtedly, the current societal viewpoint of autonomous and intelligent machines as exclusive and costly needs to shift towards seeing them as accessible, plug-and-play, and cost-effective tools [22]. Moreover, in the context of specific application scenarios, such as space exploration and first-aid, hundreds of droids will be deployed in unexplored and infrastructure-less scenarios without the possibility of human intervention [20], pushing even more the need for lightweight, cost-effective, and distributed solutions for autonomous and heterogeneous swarms of miniaturized robots [23], [24].

### B. RELATED WORKS AND COMMERCIAL SOLUTIONS FOR DISTRIBUTED SLAM

Multi-robot distributed simultaneous localization and mapping (SLAM), in which a group of autonomous agents jointly create and maintain a shared map of the surroundings, is one of the essential elements to enable distributed cooperation and optimal task planning [7], [24], [25]. Typical SLAM methods produce high-resolution 3D maps [26], exploiting vision-based solutions based on stereo cameras or light detection and ranging (LiDAR) depth sensors [27]. Numerous SLAM systems have been integrated into UAVs, reporting mapping errors as low as 12 cm [28]. While these standard approaches offer a robust benchmark for evaluating our system's performance, they are generally very demanding in terms of computational and memory resources, requiring expensive system-on-module (SoM) platforms integrating CPUs and GPUs with gigabytes of local memory [29], [30]. In contrast, [31] shows a preliminary lightweight SLAM system that employs similar loop closure and map optimization techniques. However, it operates in a centralized manner and is limited to a single agent, thereby constraining its mapping area to the drone's flight time. Moreover, its hand-crafted exploration algorithm is restricted to only vertical and horizontal movements.

Other limitations of multi-agent systems include the system hardware cost, sensing, and the wireless bandwidth required to exchange real-time information with other agents in the swarm. State-of-the-art (SoA) dense vision-based SLAM (vSLAM) approaches require the transfer of up to 3.5 GB of data in a centralized scheme, or ∼150 MB per robot in a distributed and communication-efficient scheme [17], [32]. Therefore, an increased number of robots would saturate a WiFi network with only a few agents, in the range between 3 and 10, generating a fundamental scalability bottleneck [17], [32]. Overall, there has yet to be a fully distributed and highly scalable SLAM system addressing all the associated challenges of supporting large numbers of collaborative agents operating with limited onboard resources without relying on an external infrastructure for sensing and/or computation [33]. These challenges encompass communication and computational requirements alongside the hardware costs for sensing and data processing [24].

### C. CONTRIBUTION

This work approaches the research challenge of collaborative SLAM (C-SLAM) with a novel approach in every aspect, from sensing to computation, in challenging real-world environments such as indoor areas filled with obstacles. The primary contribution is a fully distributed C-SLAM system for multi-robot dense mapping using sparse sensing. Our system enables a swarm of collaborative robots to estimate a 2D map of the environment in real-time.

Each agent runs entirely onboard the SLAM algorithm to process depth-inertial sensor data based on local state estimation and a low-power, cost-effective, and lightweight 64-pixel depth camera. Wireless communication can be supported by any commercial standard, including WiFi and Bluetooth low energy (BLE) [34], while for the scope of this work we employ ultra-wideband (UWB) to also enable infrastructure-less intra-swarm ranging (i.e., distance estimation). A distributed ranging procedure is utilized to perform inter-robot collision avoidance. Each swarm agent performs real-time local mesh updates to correct mapping drift through multi-robot loop closure. During the loop closure process, the system first aligns sub maps using the iterative closest point (ICP) algorithm [35] and then uses those alignments as constraints to correct the entire map. The steps of this process are illustrated in Figure 1. Moreover, the proposed implementation is modular, allowing different agents to join/leave the swarm dynamically, supporting heterogeneous robots.

While LiDARs paired with depth-based SLAM proved to yield high mapping accuracy, they have a high cost of a few hundred dollars. Thus, mounting such a unit onboard every robot in a swarm would dramatically increase the total cost. We demonstrate the possibility of enabling mapping with similar SoA accuracy using $5 time-of-flight (ToF) $8 \times 8$ multizone depth sensors and a low-power commercial-off-the-shelf (COTS) microcontroller. So far, SLAM, and especially C-SLAM, has been a prerogative for high-end platforms featuring GPUs coupled with several gigabytes of memory that cost hundreds of dollars. On the other hand, our highly optimized C-SLAM pipeline can run in real-time on COTS ultra-low-power microcontrollers featuring 1.5 MB of RAM. With a power budget of $\sim$100 mW and a cost below $10, our C-SLAM engine can correct the map in about 250 ms after every loop closure. The required power budget, including sensing and computation, is below 1 W. Therefore, our work provides mapping capabilities to a wide range of robotic platforms, including inexpensive, lightweight, and resource-constrained drones. As a numerical example, our system can map an area of 100 $m^2$ using four drones, with each drone requiring less than 200 kB of RAM and achieving a mean localization error of approximately 15 cm.

In addition to the theoretical formulation and simulation-based studies, our work demonstrates with practical and extensive field experiments how a team of ultra-constrained centimeter-size unmanned aerial vehicles (UAVs) can collaboratively map a generic environment relying only on onboard capabilities for mapping, communication, localization, and sensing. Figure 1 illustrates our swarm of nano-UAVs [23] deployed in the field, performing collaborative mapping of an indoor environment. Our novel contributions can be summarized as follows: *(i)* a distributed lightweight C-SLAM framework that runs onboard and performs depth-based loop closure and distributed trajectory optimization. *(ii)* A communication scheme that enables the robots to coordinate, manage loop closure information of other agents, and minimizes the amount of exchanged data. *(iii)* A navigation strategy that attempts to uniformly distribute the swarm in an unexplored environment. *(iV)* The whole project, together with the hardware, the firmware, the navigation policy, and the mathematical formulation, is released open-source.

While our experimental evaluation is performed with UAVs, our C-SLAM system can be deployed on any robot that can accommodate a payload of approximately 10 g, a power budget of 1 W, and that is equipped with odometry capabilities. Note that the odometry capabilities can even be achieved using low-cost and low-resolution sensors [36], such as the PMW3901 camera that features a resolution of $35 \times 35$ pixels. These sensors do not need to provide high accuracy, as the odometry errors are corrected by the C-SLAM algorithm.

## II. ALGORITHMS

In this section, we introduce the ultra-lightweight C-SLAM algorithm and the exploration strategy that the robots use to improve the mapping coverage in unknown environments. Furthermore, we expound upon the C-SLAM scheme and the mechanism used by each swarm agent to rectify its trajectory and align maps to ensure global consistency. Our solutions can be paired with any robotic platform, as long as it provides depth measurement capabilities enabled by sensors, such as the $8 \times 8$ STMicroelectronics VL53L8CX.

In the following, we assume a system equipped with four VL53L8CX depth sensors, each oriented differently (i.e., front, back, left, right) to provide omnidirectional coverage. Although the depth ToF sensors output information in matrix format (i.e., depth across vertical and horizontal directions), our system performs 2D mapping and therefore we need to reduce the depth measurements to one plane. For this purpose, we reduce each matrix to a depth row by selecting the median pixel from each column. Given that the sensors employed to demonstrate the effectiveness of our solutions have a resolution of $8 \times 8$, this translates to 8 pixels after the reduction or a total of 32 pixels for all four sensors.

### A. COLLABORATIVE SLAM SCHEME

Our mapping system employs graph-based SLAM, which represents the robot trajectory as a pose graph. The nodes are poses sampled at discrete times (i.e., $\boldsymbol{x}_k$) and the edges are relative measurements between the poses. Each pose $\boldsymbol{x}_k = (x_k, y_k, \psi_k)$ is expressed in the world frame and consists of the 2D position and heading, where k represents the timestamp. Furthermore, we note as $\boldsymbol{z}_{i,j}$ the relative measurement from poses $i$ to $j$. Any two consecutive nodes within the graph of a robot are connected by an edge characterized by the 3-element vector $\boldsymbol{z}_{i,i+1}$ provided by the drone's state estimator – named odometry edge. At every instant $k$, the robot not only stores a new pose but also acquires 32 distance measurements from the four depth sensors, which constitute a depth frame. The poses and their associated depth frames (i.e., sparse measurements) are sufficient to produce the map

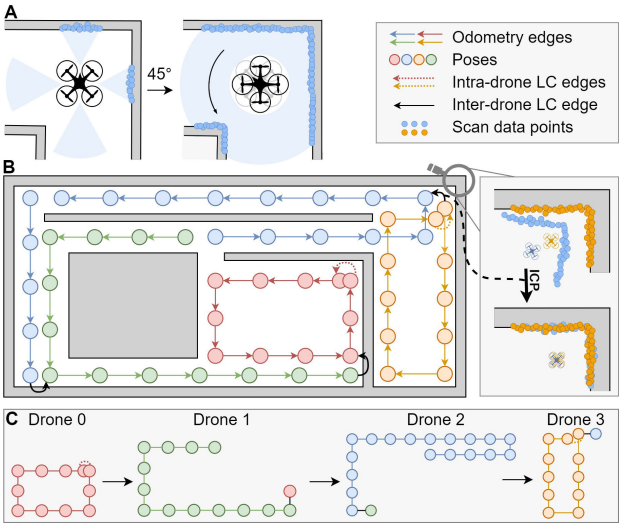



**FIGURE 2.** The C-SLAM scheme. (A) Illustration of how a drone acquires a scan once reaching a texture-rich location. (B) Composite visualization of individual pose graphs and how they are connected through inter-drone loop closure edges. (C) The cascaded distributed SLAM optimization, illustrating the graph optimized by each drone.

by projecting the distance measurements in the world frame (dense mapping).

However, the odometry edges are affected by errors, and computing the pose values using forward integration would cause the trajectory to drift over time. Consequently, our C-SLAM uses a correction mechanism that compares observations of the environment in revisited locations. These observations are small map tiles produced by aggregating 20 consecutive depth frames, which we define as a *scan*. Figure 2-A illustrates the process of creating a scan by sequentially appending the projected depth frames. We chose the number 20 because it offers a good trade-off between having sufficient overlapping information for robust scan-matching and keeping the computational load manageable. Moreover, since the cumulative field of view (FoV) of the four depth sensors is 180°, the drone rotates in place by at least 45° during the scan acquisition to ensure full environmental coverage. Thus, a *scan* refers to the entire set of 20 registered depth frames, providing a comprehensive 360° representation of the environment. For a depth frame acquisition rate of 7.5 Hz – as used in this work – acquiring a scan takes about 2.66 s. Each scan is matched with a pose, representing the robot position right before the acquisition starts. Using scan-matching, our system determines the optimal rotation and translation that overlaps one scan over the other. Since the poses and their associated scans are acquired at the same time, the transformation resulting from scan-matching also applies to the poses. In this way, when the drone revisits a location, scan-matching determines an accurate rigid body transformation relative to a pose that was previously acquired in that location, to perform loop closure.

We distinguish two types of loop closures: *(i)* intra-drone loop closures, occurring when the compared scans belong to the same drone, and *(ii)* inter-drone loop closures, occurring when scan-matching ovelaps scans belonging to two different robots. The intra and inter-drone loop closures are essential for ensuring the accuracy of the local-global maps and for maintaining overall consistency on the reference frame. The information provided by scan-matching is incorporated in the graph as an additional loop closure edge. Then, our system uses pose-graph optimization (PGO) [31], [37], propagating the loop closure information through the whole graph to correct all previously acquired poses. In our work, we employ the ICP algorithm to perform scan-matching [35].

$$X^* = \arg\min_x \Big( \sum_i \underbrace{e_{i,i+1}^T \Omega_{\text{odom}} e_{i,i+1}}_{\text{odometry edges}} + \sum_{i,j} \underbrace{e_{i,j}^T \Omega_{\text{LC}} e_{i,j}}_{\text{intra LC edges}} + \sum_{i,l} \underbrace{e_{i,l}^T \Omega_{\text{LC}} e_{i,l}}_{\text{inter LC edges}} \Big) . \quad (1)$$

Thus, the scan-matching mechanism facilitates not only the correction of a robot's pose in relation to its own previous pose (i.e., intra-drone loop closure) but also in relation to the pose of another drone (i.e., inter-drone loop closure). Figure 2-B shows an example of the graphs associated with multiple agents, where each agent maps a different environmental area. Nonetheless, an overlapping segment is necessary to establish connections between the drones' respective graphs. Within our distributed optimization scheme, each agent exclusively engages in inter-loop closures with drones possessing lower IDs. Furthermore, the graph each drone optimizes consists of its own poses alongside the external poses. To ensure coherence across all graphs, external poses remain unaltered during PGO, serving solely as reference points (i.e., anchor points) for optimizing self poses. In other words, each drone aligns its map relative to those with lower IDs. This decoupling mechanism enables each drone to independently optimize its own graph while preserving global consistency. Alternatively, other approaches optimize the joint graph maintaining the bidirectional constraints between drones [17], [38]. While still distributed, these methods are iterative and require neighboring drones to exchange the values of common poses after each iteration. This introduces additional communication overhead that would slow the optimization process and impact the available radio bandwidth. The choice of decoupling is further validated by [7], which employs a similar strategy as ours and reports no loss in precision compared to fully joint optimization methods.

Equation 1 shows the optimization problem of the distributed PGO, which is solved onboard by each agent leveraging the approach from [31]. Let $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots\}$ be the set of all poses in the graph. We note as $\hat{\boldsymbol{z}}_{i,j}$ the prediction of an edge measurement, which is the edge measurement computed out of two poses $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Therefore, solving the problem in Equation 1 reduces to determining the pose values that ensure consistency between the edge measurements $\boldsymbol{z}_{i,j}$ and the predicted measurements $\hat{\boldsymbol{z}}_{i,j}$. The maximum

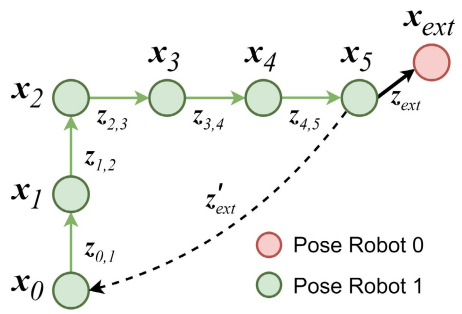



**FIGURE 3.** An example graph consisted of 6 internal poses and one external pose, showing how an external pose is transformed into an internal constraint.

likelihood solution associated with Equation 1 is equivalent to minimizing the sum of squared differences $\boldsymbol{e}_{i,j} = z_{i,j} - \hat{z}_{i,j}$. Each term $\boldsymbol{e}_{i,j}$ is paired with an edge and weighted by a diagonal information matrix $\Omega$. When computing the errors associated with the odometry edges, $j = i+1$ as the odometry edges are always connecting consecutive poses. Moreover, the external poses $\boldsymbol{x}_k$ used to compute the inter-drone loop closure edges remain unchanged during the optimization.

### B. DERIVING INTER-DRONE LOOP CLOSURE CONSTRAINTS

As prerequisites for the optimization, each drone requires the value of the first pose (the take-off ground truth position) and the edge values. Moreover, PGO never changes the first pose $\boldsymbol{x}_0$, which anchors the graph of each drone. The graph each robot optimizes consists of the poses derived from its own trajectory and the poses from other robots that are involved in loop closure with the respective robot. Figure 3 shows a simple example of a graph, with 6 poses acquired directly from the robot and one external pose $\boldsymbol{x}_{ext}$ involved in loop closure with the pose $\boldsymbol{x}_5$. While the odometry edges $z_{0,1}, z_{1,2}, \ldots, z_{4,5}$ are calculated with the aid of the internal state estimator, the inter-drone loop closure edge $z_{ext}$ is provided by running ICP scan-matching on the scans associated to poses $\boldsymbol{x}_5$ and $\boldsymbol{x}_{ext}$.

$$Z_{\text{ext}} = (\underbrace{Z_{\text{ICP}} X_5}_{X'_5})^{-1} X_{\text{ext}} \tag{2}$$

$$Z'_{\text{ext}} = (\underbrace{Z_{\text{ICP}} X_5}_{X'_5})^{-1} X_0 \tag{3}$$

Let $X_i$ denote the homogeneous coordinates of pose $\boldsymbol{x}_i$ and $Z_i$ the homogeneous representation of edge $z_i$ [39]; capital letters denote homogeneous variables, while lowercase letters denote their Euclidean counterparts. Let $z_{ICP}$ be the relative transformation that overlaps the two scans associated with $\boldsymbol{x}_5$ and $\boldsymbol{x}_{ext}$ (i.e., the result of ICP). Thus, $\boldsymbol{Z}_{ICP}$ allows computing the loop closure edge $\boldsymbol{Z}_{ext}$ from pose $\boldsymbol{x}_5$ to $\boldsymbol{x}_{ext}$ as in Equation 2. We recall that the $\boldsymbol{x}_{ext}$ should act as a reference for the graph of Robot 1 and correct the value of $\boldsymbol{x}_5$ so that their scans overlap - and therefore also the maps of the two robots.

Consequently, $\boldsymbol{x}'_5$ from Equation 2 represents the corrected value of pose $\boldsymbol{x}_5$. However, incorporating the pose $\boldsymbol{x}_{ext}$ and edge $\boldsymbol{z}_{ext}$ into the graph would not result in the transformation of $\boldsymbol{x}_5$ into $\boldsymbol{x}'_5$. This is because PGO would also adjust $\boldsymbol{x}_{ext}$, which is intended to remain fixed as an anchor. To mitigate this issue, we transform the loop closure edge $\boldsymbol{z}_{ext}$ into an edge from $\boldsymbol{x}_5$ to $\boldsymbol{x}_0$ – denoted $\boldsymbol{z}'_{ext}$ in Figure 3 and computed as in Equation 3. This exploits the fact that the first pose $\boldsymbol{x}_0$ never changes during PGO. Note that the structure is very similar to Equation 2, but now the transformation is relative to $\boldsymbol{x}_0$ and not to $\boldsymbol{x}_{ext}$. With this method the external poses exist in the graph only virtually, as they only yield additional edges to $\boldsymbol{x}_0$. Thus, inter-drone loop closures do not add other nodes – crucial since PGO scales quadratically with node count.

### C. UPDATE INTER-CONSTRAINTS

PGO is a process that occurs several times during a mapping mission. Because inter-drone loop closure edges rely on external poses from drones with lower IDs, these edges must be updated when external poses change. For instance, if $\boldsymbol{z}'_{ext}$ is computed from $\boldsymbol{x}_{ext}$ and $\boldsymbol{x}_{ext}$ later changes, Robot 1 must update $\boldsymbol{z}'_{ext}$. Rather than retransmitting the entire scan, which would incur high communication and computation overhead, we instead transmit only the updated pose (denoted $\boldsymbol{x}_{ext\_new}$) to update $\boldsymbol{z}'_{ext}$ based on the difference between $\boldsymbol{x}_{ext\_new}$ and the old pose $\boldsymbol{x}_{ext}$ – as shown in Equation 4.

$$Z'_{ext} = \left(X_{ext\_new} X_{ext}^{-1} Z_{ICP} X_5\right)^{-1} X_0 \,. \tag{4}$$

We highlight that Equation 4 is formulated specifically for the example introduced here, but it generalizes for any inter-drone loop closure edge. Thus, after performing PGO, each robot broadcasts the updated pose values involved in loop closure to the other robots – which use Equation 4 to update their inter-drone loop closure edges.

## III. COMMUNICATION AND RANGING PROTOCOL

The exploration algorithm requires knowledge of the relative positions and distances to all the other swarm agents. Furthermore, the C-SLAM scheme requires scans from the other drones to perform inter-drone loop closures. Figure 4 shows the multi-stage communication protocol that orchestrates how the drones exchange information, where the time spent in each stage is shown below the stage number. The protocol is implemented using UWB for the scope of this work, which is used for both data transmission and ranging. However, it can be implemented leveraging alternative commercial standards, such as WiFi and BLE [34]. Moreover, the protocol is token-based, implying that only one drone can transmit at a time, avoiding packet collision effects and, consequently, decreasing wireless traffic. After a drone receives the token, it first performs ranging with all drones of higher ID. We employ the double-sided two-way-ranging (IEEE 802.15.4a-2007) [15], [40], which involves a pairwise

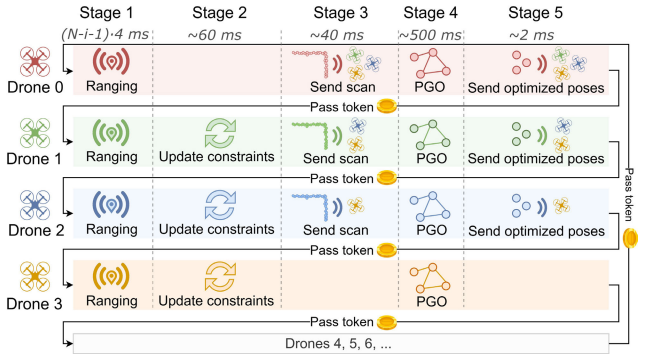



**FIGURE 4. The communication protocol, supporting a variable number of swarm agents while minimizing data traffic.**

point-to-point distance estimation. Furthermore, the drones also embed their current positions (from the pose graph) in the ranging messages. After drone $i$ completes Stage 1, it is aware of all distances and positions of the drones with ID $i+1$ or higher.

In Stage 2, a drone verifies if it has any received external scans in the buffer. Then, it pairs each external scan with the internal scan acquired at a distance smaller than a threshold $D_{th} = 1$ m. If multiple internal scans meet this condition for a given external scan, the system chooses the one acquired the earliest. For each pair, it runs ICP and derives a new inter-drone loop closure edge. However, to filter false loop closures, our system discards ICP results exhibiting a relative rotation exceeding 45°. This measure helps prevent cases where two scans, although spatially close, are captured from opposite sides of a thin wall and do not correspond to the same location. In addition, the system also checks the average Euclidean distance between two point clouds overlapped by ICP. If this distance exceeds 10 cm, the scan matching is considered unsuccessful, indicating that the scans are very noisy or do not correspond to the same location.

In Stage 3, the drone checks if a new scan has been acquired (by itself) since the last time it had the token, and if this is the case, it broadcasts the scan to all drones with higher IDs. The other drones store the received scan in a buffer and process it during Stage 2 the next time they acquire the token. Stage 4 consists of optimizing the pose graph. This is always initialized by drone 0 when it completes a certain number of loop closures (i.e., three in our experiments) or periodically at a defined time interval. To initiate the cascaded PGO, drone 0 sends an extra message to drone 1 along with the token handoff. Subsequently, each drone mirrors this action, relaying the message forward in sync with the token's progression to notify all drones to perform PGO during their next turn. After every PGO, the drone also sends the new values of the scan poses to the other drones with higher IDs - performed within Stage 5. The size of the updated poses (i.e., 12 B per pose) is negligible compared to the scan size.

We mention that while the communication protocol iterates continuously through the drones in the swarm, it is only Stage 1 which is executed every time the drone acquires the token. An arbitrary drone $i$ requires $N-i-1$ distance measurements in Stage 1 to communicate with all higher ID drones, where N is the total number of drones. This results in a total of at most $N(N-1)/2$ communications within a whole round, where each ranging requires about 4 ms. Stage 2 is typically performed several times per minute, depending on how often external scans are received and, therefore, how many texture-rich locations are in the environment. The time spent in this stage depends quasi-linearly on the number of drones and it is dominated by the execution time of ICP (typically below 60 ms per scan-matching). Since PGO does not need to happen too frequently, Stages 4 and 5 are typically executed once per minute in a normal indoor scenario.

The designed system exhibits robust fail-safe characteristics, capable of addressing radio disruptions, hardware malfunctions, or temporary absence of swarm agents. All data messages are acknowledged, and in the event of a missing response, retransmission is tried up to three times. In instances where a drone, designated by ID $i$, attempts to pass the token to drone $i+1$ without acknowledgment, there is an automated procedure to redirect the token to drone $i+2$, and so on. Additionally, any drone in the swarm will automatically reclaim the token if more than $2(i+1)s$ have passed since the last time it had the token. This mechanism preserves the functionality of the protocol even when the token-bearing drone falls outside of the communicable range or experiences a failure.

The representation in Figure 4 shows only a simplified version of the protocol. However, after each token acquisition, a drone enters a brief listening period, spanning a random interval between 0 – 20 ms to ascertain if another drone transmits during this time (i.e., has the token). Should the transmitting drone have a lower ID, the current drone discards the token. If not, it proceeds with the subsequent stages. This mechanism mitigates the situations where more drones have the token simultaneously. For example, if the drone carrying the token moves out of the swarm's radio range, another drone will claim the token. When the missing drone returns to the swarm, our protocol determines which drone keeps the token.

## IV. EXPLORATION ALGORITHM

We introduce our lightweight exploration strategy used for the scope of this paper, which is formulated as a state machine and governs the behavior of individual robots. This strategy performs a pseudo-random exploration of a given environment without requiring a target destination. However, unlike other random exploration algorithms [8], this strategy prioritizes wall following and corner visiting [41], which encompass texture-rich areas favorable to loop closures. Regardless of the state, the goal of the exploration algorithm is to provide the body velocities (i.e., $v_x$, $v_y$), and the heading angular rate $\omega$. Figure 5-A illustrates the three states of the state machine, where each robot starts in the *Cruise* mode, characterized by a cyclic execution of three steps. In the initial step, the algorithm adjusts the forward velocity $v_x$ linearly,

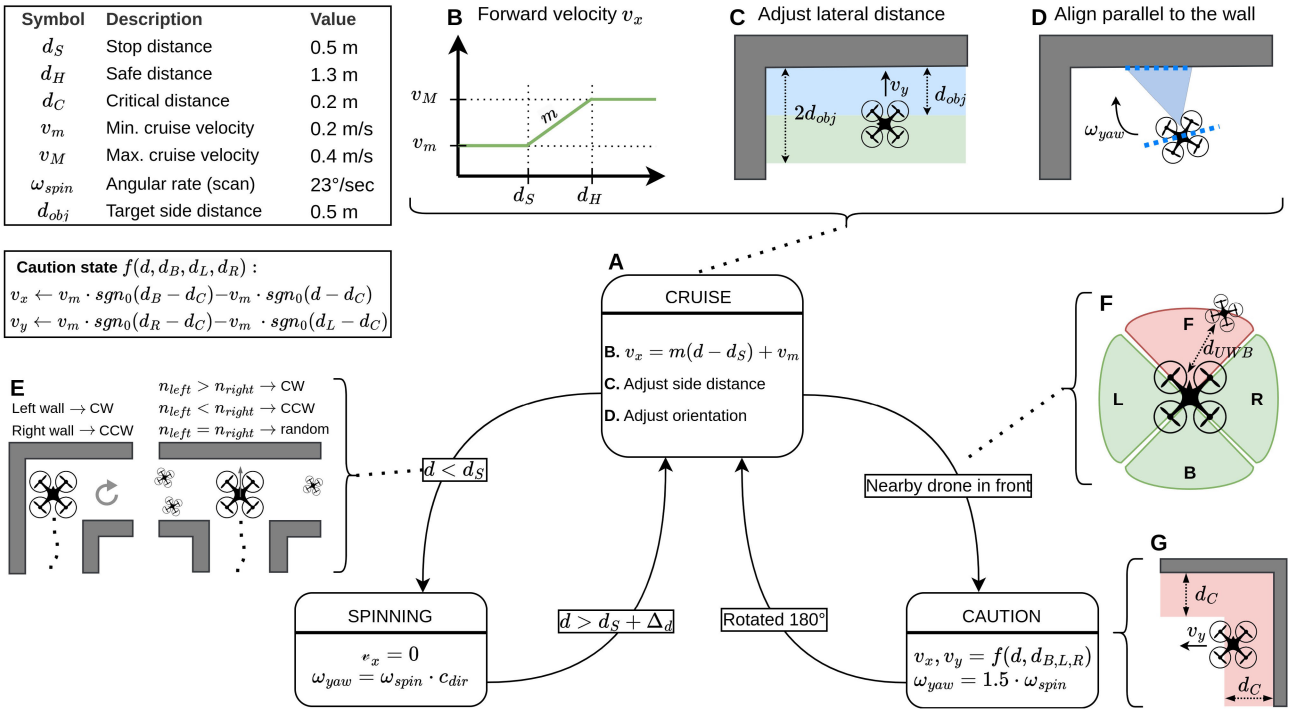
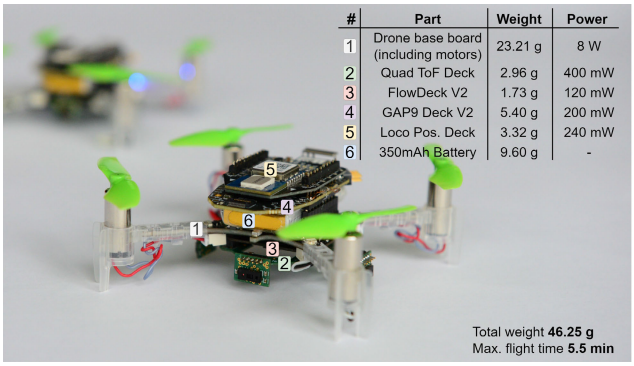



**FIGURE 5.** The state machine of the exploration algorithm illustrating the behavior in each state and the transition conditions.

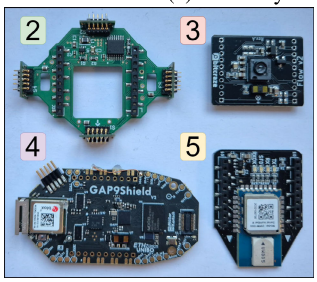
dependent on the frontal distance $d$, clipped with predefined lower and upper bounds $v_m$ and $v_M$. Note that distance $d$ is computed as the minimum value provided by the frontal depth sensor. The velocity curve is shown in Figure 5-B, where m represents the slope of the linear region. The second step verifies if there is a side obstacle within a lateral distance of $2d_{obj}$. A proportional controller then adjusts $v_y$ to position the drone at $d_{obj}$ from the detected side obstacle, as shown in Figure 5-C. Figure 5-D illustrates the final step of the *Cruise* state, where the robot checks for the presence of an object on either side. A line detector assesses if the depth measurements align with the expected readings from a flat surface. Based on the slope of the line, the system aligns the robot's orientation parallelly. If an object is detected on both sides, priority is given to the right. If an object is detected beyond the distance $d_{obj}$ or not detected at all, the commanded angular rate $\omega$ is set to 0. The first step of the *Cruise* state ensures smooth deceleration upon detecting a frontal obstacle, while subsequent steps facilitate effective wall following.

When a frontal obstacle is detected, and $d$ becomes smaller than a threshold $d_S$, the system goes into the *Spinning* state. The direction of spinning is determined by evaluating $d_L$ and $d_R$ distances on the left and right, respectively. In the event of an obstacle detected to the left (i.e., $d_L < 2d_{obj}$), the robot rotates clockwise (CW) as shown on the left side of Figure 5-E; conversely, if an obstacle is detected to the right, the robot rotates counter-clockwise (CCW). In scenarios where both sides are unobstructed, and the robot is situated at an intersection (i.e., Figure 5-E - right side), it counts how many other agents are on its left and right relative to its OX-axis (shown in grey) and turns towards the side with fewer robots. This mechanism favors a uniform robot distribution within the environment and increases the area coverage rate. In the event of an equal number of robots on both sides, a random decision is made. When $d$ becomes higher than $d_S + \Delta_d$, with $\Delta_d = 0.3$ m, a pathway ahead is again clear, and the system makes the transition back to the *Cruise* state. If the drone spends at least 2 s in the *Spinning* state, it indicates that it is facing a corner or a texture-rich area and has rotated more than 46°, thereby covering the entire lateral FoV. This is an automated mechanism to identify regions where a scan should be acquired.

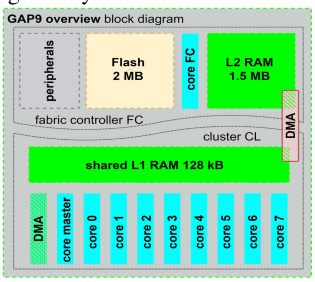
If the mission is performed by one drone only, the system would only loop between *Cruise* and *Spinning* states. The transition to the *Caution* state can only be triggered by the interaction with another robot. Figure 5-F shows how each robot virtually splits the surroundings into four areas. If there is an agent within the frontal (F) area at a distance lower than $2d_{obj}$, i.e., $d_{UWB} < 2d_{obj}$, the system goes into the *Caution* state. In finding the presence of another agent within the frontal zone, the robot utilizes the received position estimates, while for determining proximity within a distance $2d_{obj}$, it solely relies on UWB measurements. In the *Caution* state, the robot turns around 180Â°. During spinning, it also checks if an obstacle (i.e., wall or another agent) is located closer than a critical distance $d_C$, and sets $v_x$ and $v_y$ accordingly through the function $f(d, d_B, d_L, d_R)$ to push itself away from the obstacle, as illustrated in Figure 5-G. Thus, to avoid




(a) The fully integrated system.




(b) Top view of the four decks.




(c) Architecture of the GAP9.

**FIGURE 6. (a) The hardware platform used in our experiments, based on the Crazyflie v2 nano-drone. (b) The plug-in decks used by our system, that stack on the base drone PCB. (c) The constitutive blocks of the GAP9 SoC from Greenwaves, showing the 8+1 cluster cores, and the fabric controller.**

collisions, when two or more robots encounter one another head-on, they each turn around and continue exploring. Note that the distances $d_B$, $d_R$, and $d_L$ represent the minimum distances from the back, right, and left areas, respectively. When the robot completes spinning, it returns to the *Cruise* state.

## V. HARDWARE SETUP

Our mapping system is designed with flexibility in mind, aiming to provide compatibility with a wide range of robotic platforms. The primary requirement is the availability of depth sensors. As a result, the algorithm and implementation can be adjusted to support alternative hardware setups, such as various processors or sensing components. For our demonstration, we have chosen the commercial-of-the-shelf (COTS) nano-UAV Crazyflie 2.1 from Bitcraze. This option allows us to demonstrate the effectiveness of our solution even on highly constrained platforms. Consequently, our results can be easily replicated using commercially available hardware.

The open-source firmware of Crazyflie 2.1 offers functionalities for flight control, state estimation, radio communication, and setpoint commander. The drone's primary printed circuit board (PCB) serves also as frame, housing essential electronics such as an inertial measurement unit (IMU), a radio transceiver Nordic nRF51822, and an STM32F405 microcontroller unit (MCU). The latter operates at a maximum clock frequency of 168 MHz and has 192 kB of RAM, with over 70% of these resources already utilized by the firmware for control and estimation tasks. Additionally, the drone is equipped with extension headers for adding extra decks (i.e., plug-in boards).

We have incorporated the commercial FlowDeck v2, which utilizes a downward-facing optical flow camera and a single-zone ToF ranging sensor. These components enable velocity and height measurements, fused by the onboard extended Kalman filter (EKF) for position and heading estimation. Since there is no magnetometer involved, the heading estimation relies on the information provided by the gyroscope. Moreover, we use the Loco Positioning deck featuring IEEE 802.15.4 UWB communication and ranging, via the COTS Decawave DWM1000 module. The DWM1000 enables ranging with an accuracy of about 10 cm [42]. However, the distance information provided by UWB is not utilized by the state estimator, which relies solely on the aforementioned sensors. In addition, we include two custom-designed decks: one housing four depth ToF sensors to enhance the drone's environmental sensing capabilities, and the second deck featuring the GAP9 system-on-chip (SoC). This serves as a co-processor, extending the computational capabilities of Crazyflie 2.1. Therefore, computationally intensive tasks such as running ICP scan-matching, graph-based SLAM, scan computation, or map generation are executed on the GAP9. In this configuration, the total weight at take-off is 46.25 g, which includes all the hardware utilized for the scope of this paper. The fully integrated system, highlighting our custom hardware, is depicted in Figure 6a, while Figure 6b shows the decks mounted on the drone. The STM32 MCU serves as the manager for all processes running onboard the nano-UAV, including sensor data acquisition. While it does not handle heavy computations itself, it is responsible for delegating these tasks to GAP9 via SPI communication. During the flight, the STM32 retrieves depth data from the four ToF sensors and the current pose from the internal state estimator, transmitting this information to GAP9.

### A. THE CO-PROCESSOR DECK FEATURING THE GAP9

We use the co-processor deck proposed in [43], which weighs 5.4 g and incorporates the GAP9 SoC – based on the parallel ultra-low-power (PULP) paradigm [44], and developed by Greenwaves Technologies. Figure 6c illustrates the primary components of the GAP9. The GAP9 SoC features 10 RISC-V-based cores, organized into two power and frequency domains.

The first domain is the fabric controller (FC), housing a single core capable of operating at speeds of up to 400 MHz, paired with 1.5 MB of RAM (L2 memory). The FC serves as the main processor of the SoC, managing communication with peripherals and coordinating on-chip memory operations. The second domain is the cluster (CL), comprising nine RISC-V cores capable of operating at speeds of up to 400 MHz. These cores are specifically designed to

handle highly parallelizable and computationally intensive workloads. Within the CL, one core acts as a "leader core," receiving tasks from the FC and distributing them to the other eight cores in the cluster, which execute the computation. The CL cores share the same instruction cache, enabling efficient execution of identical code on different data. The CL is accompanied by 128 kB of L1 memory (shared among the CL cores), with transfers between L2 and L1. Moreover, the CL features a neural engine that provides hardware-level acceleration for operations like convolutions, batch normalization, or ReLU activations, supporting the execution of quantized deep learning models.

The GAP9 interfaces with the STM32 via SPI and handles all the intensive computation required by PGO and scan-matching.

### B. CUSTOM QUAD TOF DECK

The VL53L5CX is a lightweight multi-zone 64-pixel ToF sensor, weighing only 42 mg. This sensor offers a maximum ranging frequency of 15 Hz at an 8 × 8 pixel resolution. It provides a maximum range of 4 m, covering a FoV of 45° (the complete sensor characterization is conducted in [23]). Moreover, the VL53L5CX provides a pixel validity matrix alongside the 64 depth measurements, automatically identifying and flagging noisy or out-of-range measurements. To facilitate the utilization of multi-zone ranging sensors on the Crazyflie 2.1 platform, a custom deck was employed specifically to accommodate four VL53L5CX ToF sensors, as depicted in Figure 6b. The four ToF sensors face the front, back, left, and right directions, enabling obstacle detection within a cumulative FoV of 180°. The custom deck weighs 2.96 g and its hardware design is released as open source.

## VI. PROTOCOL IMPLEMENTATION

The following section explains in detail how the pose-graph data exchanged by the communication protocol is stored and managed by the system. For every timestamp, the system acquires a new pose and adds it to the graph. A pose consists of three variables – namely, the 2D position $(x_k, y_k)$ and heading $\psi_k$ – which are obtained from the onboard EKF that is part of the open-source drone firmware. Since the pose information is paired with depth measurements from four depth sensors, we refer to the structure that encompasses both pose and depth information as an *augmented pose*. These augmented poses are continuously stored in the *augmented pose table* that is represented in memory as an array. As shown in Figure 7-(a), each augmented pose has a size of 84 B, with its largest field consisting of 32 depth measurements stored as *int16*. The pose itself comprises three *float* elements. In addition to the pose and depth data, each augmented pose is labeled with a unique ID and a timestamp. The rate at which the system adds new augmented poses is limited by the data rate of the depth sensors, which is 15 Hz. However, since nano-drones typically fly at speeds below 2 m/s [23], we selected a pose acquisition rate even lower, at 7.5 Hz. Decreasing the acquisition rate further could result in gaps in the map point cloud, if the drone approaches its maximum speed.

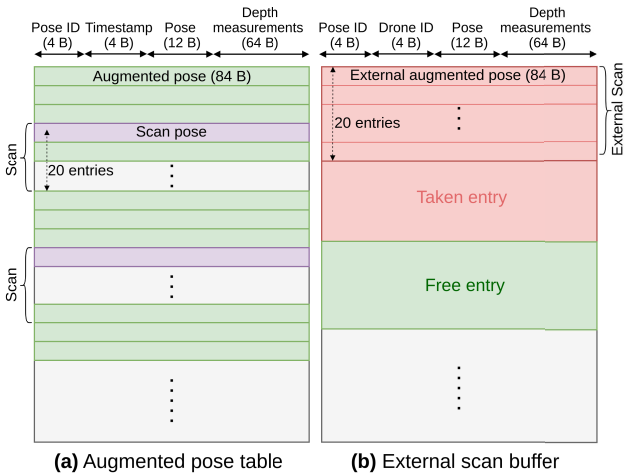



**FIGURE 7. Graphical representation of the main data structures that store the pose graph.**

The system also stores the IDs of the scan poses – i.e., the first pose of each scan. Once the exploration algorithm identifies a transition from the *Spinning* to the *Cruise* state, it checks if it spent more than 2 s in the *Spinning* state (corresponding to a rotation of 46°). If this was the case, it labels the augmented pose $k-20$ as a scan pose. Whenever the system needs to fetch a scan, it goes into the table at the target pose ID and fetches 20 subsequent entries. In this way, there is no need to allocate separate memory for the scans, and redundancy is avoided.

In addition to the augmented pose table, the system allocates substantial memory for storing external scans received from other drones in the *external scan buffer* shown in Figure 7-(b). Each entry in this buffer holds 20 external augmented poses, as drones exchange scans in this format. An external scan typically stays in its dedicated buffer until it is matched with one of the internal scans through inter-drone loop closure. After completing an inter-drone loop closure, the corresponding entry in the external scan buffer is freed, and the resulting constraint is added to the *external constraints list*. Each entry in this list contains the poses involved in the inter-drone loop closure, the ID of the other drone, and the relative measurement $Z'_{\text{ext}}$ derived as described in Section II-B. Additionally, it contains $Z_{\text{ICP}}$, enabling updates to $Z'_{\text{ext}}$ when an external pose changes (e.g., due to PGO) following Equation 4.

Algorithm 1 complements the communication protocol diagram illustrated in Figure 4 and shows how it interacts with the augmented pose table and the scan buffer at every stage. The execution time of Stage 1 increases linearly with the number of drones $N$, with drone 0 taking the longest because it must perform ranging with every other drone. Stage 2 operates in $\mathcal{O}(n_{ext} \times n_{int})$ time, since for every external scan, it searches for a matching internal scan, where $n_{ext}$ and $n_{int}$ denote the number of external and internal scans, respectively. Whenever deriving external constraints in Stage 2, the system

**Algorithm 1** Pseudo-Code of the Communication Protocol

```
while mission_in_progress do
    if has_token() then
        // Stage 1: Ranging and Position Update
        for drone_id ← current_id + 1 to N − 1 do
            (dist, pos) ← do_ranging_with(drone_id)
            update_states(dist, pos, drone_id)
        end for
        // Stage 2: Update External Constraints
        for all ext_scan ∈ scan_buffer do
            int_scan_match ← find_match(ext_scan)
            if int_scan_match ≠ null then
                edge ← icp(int_scan_match, ext_scan)
                add_to_external_constraints(edge)
            end if
        end for
        // Stage 3: Broadcast New Scan & Intra Loop Closures
        if new_scan_acquired() then
            for drone_id ← current_id + 1 to N − 1 do
                send_scan(last_scan, drone_id)
            end for
            for all int_scan ∈ int_scan_buffer do
                scan_match ← find_match(int_scan)
                if scan_match ≠ null then
                    edge ← icp(scan_match, int_scan)
                    add_to_internal_constraints(edge)
                end if
            end for
        end if
        // Stage 4: Pose-Graph Optimization
        if do_pgo is true then
            run_pgo()
            // Stage 5: Broadcast Updated Poses
            for drone_id ← current_id + 1 to N − 1 do
                send_scan_poses(drone_id)
            end for
        end if
        pass_token_to_next_drone()
    else        // Listening Mode (No Token)
        if ranging_request() is true then
            (dist, pos, drone_id) ← do_ranging()
            update_states(dist, pos, drone_id)
        end if
        if ext_scan_received() is true then
            add_to_ext_scan_buffer(ext_scan)
        end if
        if ext_pose_upd_received() is true then
            update_ext_constraint_list(pose_upd)
        end if
        if token_received() is true then
            claim_token()
        end if
    end if
end while
```

checks potential matches between external scans in the external scan buffer and internal scans by comparing if the Euclidean distance between their scan poses is smaller than $D_{th}$. The same mechanism applies when acquiring a new scan in Stage 3, for deriving internal constraints. However, since at most one internal scan is acquired in one iteration, the matching procedure among all internal scans runs in linear time relative to their total number. Despite its complexity,

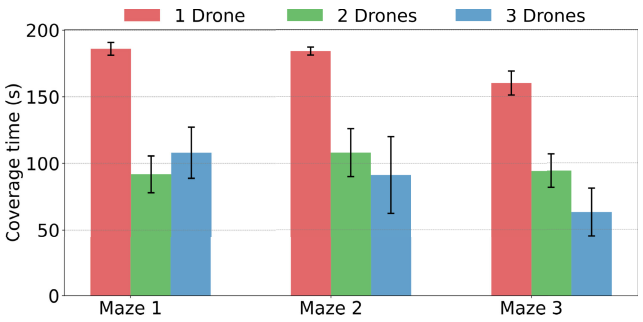



**FIGURE 8.** The coverage time for one, two, and three drones, illustrating the effectiveness of collaborative exploration.

the process of finding matches among scans is fast, due to the relatively small number of external and internal scans – typically just a few dozen at most. Conversely, executing ICP and PGO is more computationally demanding, as described in Section VII-D, because their complexity scales quadratically with the scan size and the number of poses, respectively.

Whenever PGO is performed, the values of the poses are updated in the augmented pose table. Then, the scan poses are fetched and broadcasted to the other drones with higher ID in Stage 5. After a drone passes the token, it only listens for UWB messages and does not perform computation-heavy operations. While listening, it can respond to a ranging request, process external scans by adding them to the external scan buffer, or handle pose updates to revise the external constraints list based on Equation 4.

In our setup, the STM32 is responsible for acquiring data from the depth sensors, running the exploration algorithm, exchanging data via UWB, and tracking the positions and distances to other drones. However, it does not handle large data structures, such as the augmented pose table, external scan buffer, and external constraints list, which are stored in the GAP9's L2 memory. As a result, when scans or poses need to be sent to another drone, the STM32 acts as a bridge, retrieving these large data blocks from the GAP9 over SPI and transmitting them via UWB. The GAP9 handles both the storage and computation for all tasks related to the pose graph, as well as running algorithms such as PGO and ICP scan-matching. As a realistic numerical example, consider a mission time of 5 min. In this case, each drone would need to store up to 2250 augmented poses, amounting to 189 kB. The external scan buffer, being smaller, has a size that depends on the number of external scans that need to be buffered. In an unfavorable scenario where no inter-drone loop closures occur within a minute, and assuming each drone captures up to five scans per minute, a swarm of four drones would need to buffer 20 external scans during that minute. This corresponds to 400 external augmented poses, requiring 33.6 kB of storage.

## VII. EXPERIMENTAL RESULTS

We demonstrate the efficacy of our system by conducting practical field experiments within both controlled and realistic environments, encompassing general indoor areas.

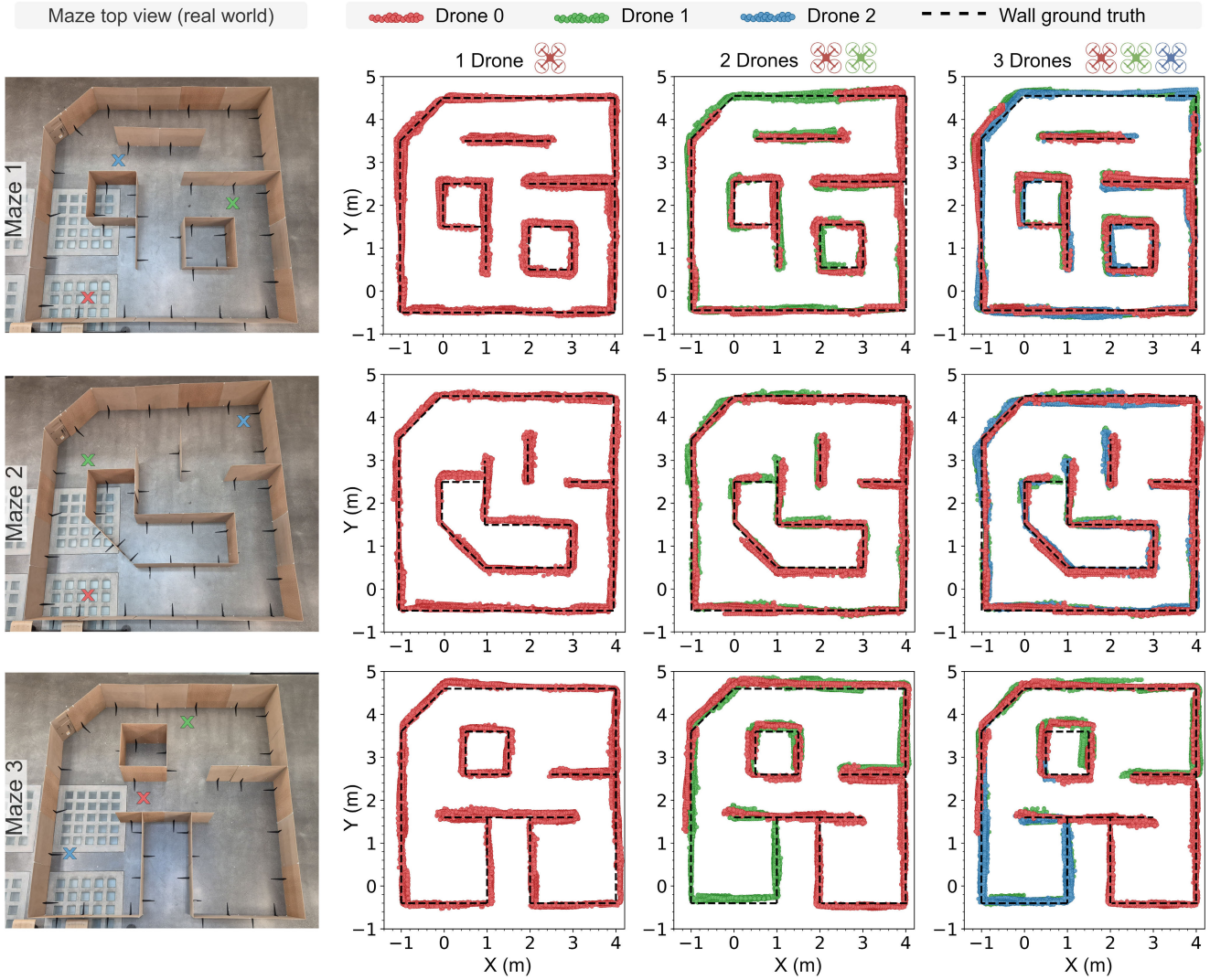



**FIGURE 9.** Mapping results in a controlled environment. The first column illustrates top view photos of each maze, taken in-field. The other columns show the maps produced at the end of the mission in each maze for one, two, and three drones, illustrating how the individual maps are aligned and merged by our C-SLAM algorithm.

Moreover, C-SLAM is compared with SoA recent works in terms of accuracy and scalability, demonstrating how hundreds of robotic agents can collaborate, sharing data and scans over standard wireless protocols. Within the scope of our mapping mission, a variable number of drones, between one and four, explore different environments, employing our exploration and C-SLAM algorithms. Furthermore, a scalability test varying the swarm size between 2 and 200 agents has been conducted in a separate experiment. During flight, drones utilize UWB radio communication to convey their respective positions to one another, as well as the distances between them. Our evaluation is centered on assessing how the coverage time scales with the number of drones, the absolute trajectory error (ATE) of each drone within the swarm, and the mapping accuracy of the collective global map.

### A. EXPERIMENTS IN A CONTROLLED ENVIRONMENT

First, we evaluate our system in a controlled environment consisting of mazes built out of chipboard panels of 1 m × 0.8 m. The goal is to evaluate the coverage time and mapping accuracy, observing how these metrics are influenced by varying the number of drones. To demonstrate the adaptability of our system to different spatial configurations, we evaluated its mapping efficacy across several mazes with distinct geometries. The Vicon Vero 2.2 motion capture system installed in our testing arena provides the ground truth for the evaluation. Figure 9 illustrates the three distinct mazes subjected to this analysis, with the take-off positions of the drones marked by an "×" of three distinct colors. After take-off, the drones embark on their exploration and mapping tasks, propelled by the underlying exploration algorithm. For each configuration, Figure 8 shows the time

the drones require to cover all accessible areas within the maze. As anticipated, the longest duration correlates with the experiments utilizing a single drone. Moreover, introducing a second drone yields a significant reduction in coverage time by 48%, 43%, and 41% for Maze 1, Maze 2, and Maze 3, respectively. However, transitioning to a configuration with three drones does not yield a proportional decrease in coverage time; reductions are observed at only 50% and 60% for Maze 2 and Maze 3, respectively, when compared to the single-drone setup. This suggests a saturation trend when increasing the number of agents per square meter. For Maze 1, the coverage time increases by 18% compared to the two-drones experiment. This phenomenon can be attributed to the drones intersecting paths twice in Maze 1 (i.e., going into *Caution* state), as opposed to once and not at all in Mazes 2 and 3, respectively. Since intersections compel the drones to reverse direction and retrace their previously covered path, they increase the coverage time. This highlights the importance of adjusting the drone count based on the size of the explorable area.

**TABLE 1. The ATE (in cm) for each robot in the swarm, highlighting the positioning precision for each configurations.**

| | 1 Drone | 2 Drones | | 3 Drones | | |
|---|---|---|---|---|---|---|
| | #1 | #1 | #2 | #1 | #2 | #3 |
| Maze 1 | 15.7 | 15.9 | 13.3 | 16.3 | 24.0 | 19.4 |
| Maze 2 | 12.8 | 12.5 | 16.2 | 12.7 | 13.4 | 17.1 |
| Maze 3 | 16.5 | 15.0 | 21.5 | 15.6 | 22.7 | 13.8 |

**TABLE 2. The mapping error (in cm) for each configuration.**

| | 1 Drone | 2 Drones | 3 Drones |
|---|---|---|---|
| Maze 1 | 6.7 | 9.5 | 10.2 |
| Maze 2 | 6.4 | 6.8 | 7.2 |
| Maze 3 | 7.3 | 11.3 | 12.0 |

Table 1 shows the ATE for each configuration, which varies in the range of 12.5 cm – 24 cm, with the highest values for the configuration of Maze 1 and three drones. Given that rotational movements worsen the ATE, due to yaw angle observability limitations of nano-UAVs operating indoors, and each encounter between drones necessitates a 180° spin, it logically follows those configurations characterized by a higher frequency of intersections exhibit increased ATE values – this effect is nearly negligible during scans because the rotation angle is significantly smaller. Furthermore, Maze 3 also shows a relatively high ATE, regardless of the number of drones. In this case, it is not the drone intersection that increases the ATE but the geometry of the maze, as the two "rooms" at the bottom are only connected through an upper path. The absence of loop closures in the lower section of Maze 3 results in distortion of the trajectory, thereby influencing the ATE. Within the scope of this work, we specifically selected a constrained drone platform featuring limited odometry capabilities that restrict the trajectory estimation accuracy to demonstrate how C-SLAM provides stability and robustness even under poor observability conditions.

**TABLE 3. Number of intra-drone loop closures per experiment. Inter-drone loop closures are shown in bold.**

| | 1 Drone | 2 Drones | | 3 Drones | | |
|---|---|---|---|---|---|---|
| | #1 | #1 | #2 | #1 | #2 | #3 |
| Maze 1 | 6 | 8 | 4 (**3**) | 3 | 2 (**6**) | 4 (**6**) |
| Maze 2 | 7 | 7 | 8 (**5**) | 7 | 8 (**5**) | 7 (**5**) |
| Maze 3 | 4 | 9 | 4 (**4**) | 9 | 7 (**3**) | 15 (**3**) |

Figure 9 (right side) presents the mapping outcomes for each maze aimed at evaluating the effect of multiple inter-drone loop closures on the map's integrity. The metric used to assess the overall C-SLAM accuracy is the absolute mapping error. This metric first calculates the projection to the closest straight line (or its extension) for every point in the map. Then, the mapping error is computed as the root-mean-squared-error (RMSE) of all the projections. The mapping RMSE does not only depend on the state estimation error, but it also correlates with the shape of the environment and the measurement noise of the depth measurements. The overall swarm mapping error spans from 6.4 cm to 12 cm, as shown in Table 2. The results are aligned with Table 1, yielding the smallest overall mapping error for Maze 2, and the largest error for Maze 3. This is expected because each drone has its own biases, and anchoring the maps in a finite number of points may not completely align them.

Table 3 shows how many loop closures are performed in each mission. While the numbers are comparable among configurations, we notice a larger amount of intra-drone loop closures when mapping Maze 3 with three drones. This is due to the nature of the exploration policy, as once a drone reaches one of the bottom "rooms" of Maze 3, it is likely to perform several loops until it exits the area. This leads to multiple intra-drone loop closures performed in a short time. The maps presented in Figure 9 underwent a filtering phase, where points with a low density of neighbors were removed, assuming they likely represent outliers corresponding to nearby drones rather than real objects. This approach enhances the map's accuracy by ensuring it reflects only points associated with physical features, based on the premise that genuine features are typically surrounded by a cluster of points.

### B. EXPERIMENTS IN A REAL-WORLD INDOOR ENVIRONMENT

We also prove the effectiveness of our C-SLAM in a realistic environment. For this purpose, we map an 18 m × 10 m area in an office environment employing a swarm of four nano-UAVs, depicted in Figure 6-(a). Based on the results illustrated in Figure 8, the optimal number of drones was inferred from the point at which coverage time saturates, indicating an optimal number of robots per area unit. Furthermore, the number of agents used in the experiment closely aligns with recent collaborative mapping studies [17].


**FIGURE 10.** Samples from the real environment where the mapping mission is conducted, highlighting the obstacles.

Two scenes of the environment are illustrated in Figure 10, while the obstacles are labeled in Scene 1. This environment is selected as the worst-case scenario, with few favorable areas for loop closure and a thick concrete wall affecting wireless communication. The proposed experimental setup pushes the limits of C-SLAM, enabling distributed mapping on ultra-constrained nano-robotic platforms. Since there is no trajectory ground truth in this experiment, we represent in Figure 11-A the onboard estimated trajectories that were corrected by the C-SLAM algorithm to show the effectiveness of our exploration algorithm in distributing the drones through the whole environment. They initiate their flight from positions denoted by '×', achieving complete area coverage in approximately 60 s.

Figure 11-B shows how the map looks at the end of the mission, where the data points produced by each drone are represented with a distinct color. Additionally, the locations where the two environmental snapshots were captured are indicated. Notably, the experiment exhibits accurate corner alignment attributed to inter-drone loop closures. The most significant discrepancy is noted adjacent to the longest wall at the bottom of the area, approximately 15 m, where the absence of loop closures over extended distances results in substantial accumulated trajectory drift, thereby inducing ATE errors that are not fully rectifiable. Conclusively, the experiment leads to a mapping error of 29.7 cm. It is important to note that, in contrast to the maze-based experiments characterized by a more confined area, in this experiment, the drones frequently encountered UWB radio out-of-range situations limiting the relative intra-swarm localization, predominantly due to the obstructions of concrete walls found in the line of sight. Despite these challenges, our system demonstrated robustness against connection-drop conditions, and it consistently demonstrated the ability to recover from such states when the drones returned within the communication range.

### C. SCALABILITY STUDY

Our system prioritizes scalability, hardware cost, and computational load. Based on the intra-swarm data communication and localization discussed in Section III, a scalability study is presented in Figure 12a and Figure 12b, demonstrating the possibility of supporting up to 200 robots. We define

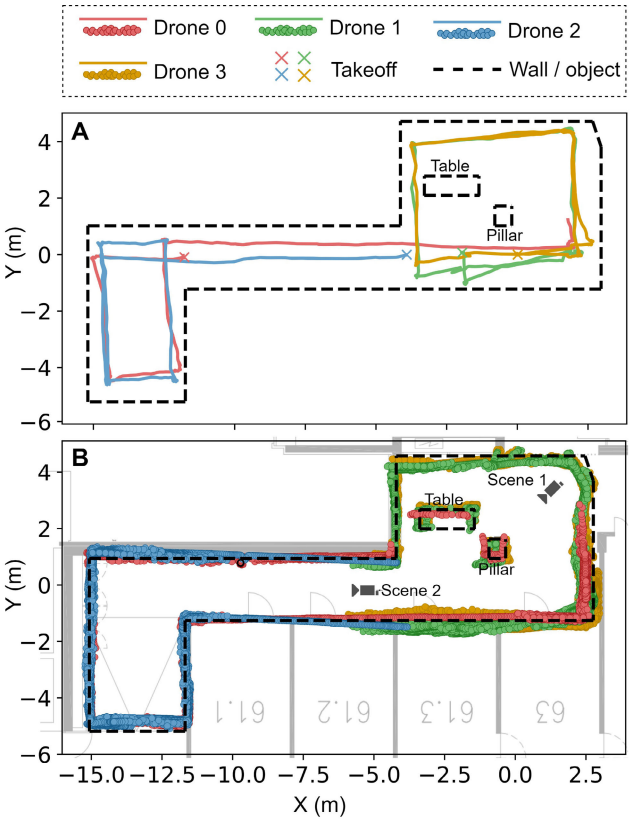
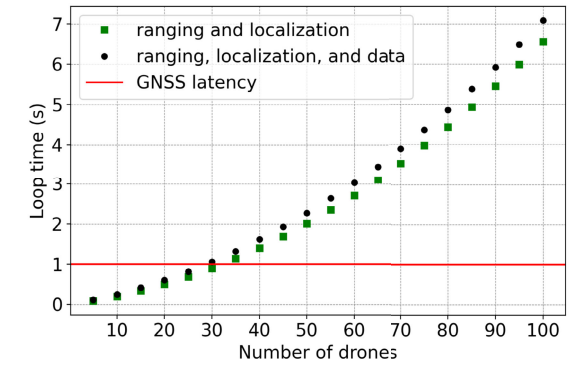


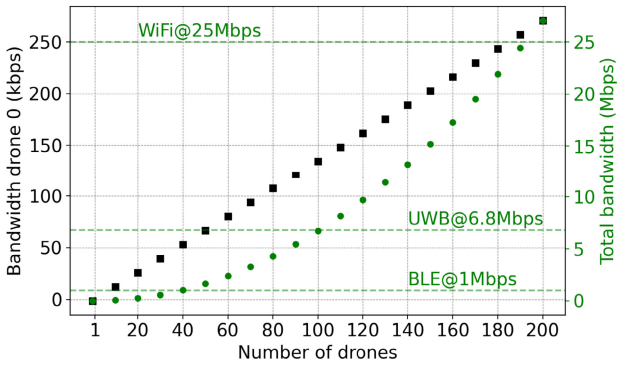

**FIGURE 11.** Mapping results in a real world environment. (A) The trajectories of each drone, color-coded according to the legend for clear differentiation (B) The collaborative map created by the drone swarm.

loop time as the total time necessary for the drones in the swarm to communicate their positions to each other, the distances between them, and the scans. In the communication scheme we propose, only one drone transmits at a time, and the transmission scheduling for each drone occurs in increasing order based on its ID. Therefore, a large loop time would impact the update rate of the positions, decreasing the robustness of collision avoidance. Figure 12a shows how the loop time depends on the number of drones. The position and distance updates are relevant for inter-drone collision avoidance, while the scan transfer is associated with mapping and loop closure. Consequently, Figure 12 illustrates the loop time with and without considering the influence of the scans. Given that the scan acquisition frequency is usually considerably lower than the loop frequency, this study presumes that each drone sends at most one scan per loop and no more than 20% of the drones in the swarm send a scan within the same loop. We highlight that a latency comparable to GNSS-based localization is obtained for a swarm size of 30 drones. Moreover, while previous works [32] achieve a 2.7 s latency with a swarm of 10 drones (data transmission only), our system allows a swarm of 55 robots with the same latency (data transmission and localization).

We highlight that the loop time in Figure 12a mainly depends on transmitting the drones' positions and distances.




(a) Loop time.




(b) Necessary radio bandwidth.

**FIGURE 12.** Scalability study. (a) The loop time, including UWB ranging and localization alone and with the effect of transferring scans. (b) The radio bandwidth per drone (drone 0 has the largest bandwidth) and for the whole swarm, showing the nominal data rate of common wireless technologies.

However, combining an absolute positioning system with our C-SLAM pipeline would alleviate the need to wirelessly measure the relative drone positions. In such a scenario, the only scalability limitation would lie in the necessary bandwidth for transferring the scans. Figure 12b shows the bandwidth requirement as a function of the number of drones. Furthermore, we represent both the total swarm bandwidth and the bandwidth required by drone 0 (i.e., the worst case). In addition, the figure shows that conventional communication technologies [34] such as WiFi, UWB, and BLE can accommodate about 190, 100, and 40 drones, respectively. Although the WiFi protocol (Wi-Fi 6 - IEEE 802.11ax) can reach up to 600 Mbps, in line with the scope of this work, we selected a cheap and miniaturized WiFi module (NINA-W10 from u-blox) supporting up to 25 Mbps. Within this investigation, we consider an average number of five scans per minute and a scan size of 2 kB. The possibility of pairing our setup with BLE enables even teams of insect-size robots to collaboratively map a previously unknown environment [21].

### D. ONBOARD EXECUTION TIME AND ENERGY ANALYSIS

We provide an analysis of the execution time of the adopted algorithms. Our focus lies on evaluating ICP and PGO onboard the robotic platform and only relying on its computing capabilities. Table 4 (top) presents the execution time of ICP as a function of the scan size. We sweep the scan size in the range of 128 - 1024 points and obtain an execution time in the range of 3 ms – 138 ms. In this work, we select a scan size of 640, as the best trade-off between scan-matching accuracy and execution time. Note that the average power consumption (in the range of 122 mW – 178 mW) increases with the scan size, because a larger scan results in a longer activity of the GAP9 CL. Our C-SLAM algorithm optimizes the trajectory of each individual robot relying on the graph-based SLAM proposed in [31]. This algorithm uses a hierarchical approach, which firstly splits the graph into multiple subgraphs, and then optimizes each subgraph individually. Since the memory constraints limit the maximum size of the graph to about 440 poses when performing graph optimization, the hierarchical method allows to optimize much larger graphs (up to about 3000 poses). The subgraph split is typically performed using a distance threshold, and each subgraph corresponds to a travelled distance of about 0.8 m. The optimization itself has a quadratic complexity, but since the size of the subgraphs is approximately constant, the execution time of the whole hierarchical optimization is linear in the number of subgraphs, and therefore in the total number of poses (assuming an equal number of poses per subgraph). This is proven by the execution time measurements shown in Table 4 (bottom). Furthermore, the average power consumption is approximately constant in all configurations - about 70 mW.

The results in Section VII-A show that the overall mapping time decreases nearly linearly with the addition of more drones - except for cluttered environments, where frequent drone encounters slow down the coverage. In the following, we also conducted a study to analyze how per-drone energy consumption varies with swarm size. Table 5 shows the energy used by each drone over one minute as the swarm size ranges from 10 to 100. We report only the energy for computation and UWB communication, as these components change with the number of drones. As the swarm size grows,

**TABLE 4.** Execution analysis of execution time, average power and energy of the ICP and graph-based SLAM algorithms onboard the GAP9 SoC.

| ICP analysis | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Scan size | 128 | 256 | 384 | 512 | 640 | 768 | 896 | 1024 |
| Time (ms) | 3 | 10 | 21 | 36 | 55 | 79 | 107 | 138 |
| Power (mW) | 122 | 150 | 165 | 170 | 172 | 175 | 176 | 178 |
| Energy (mJ) | 0.54 | 1.8 | 3.8 | 6.5 | 10 | 14 | 19 | 25 |
| PGO analysis | | | | | | | | |
| Poses | 800 | 1000 | 1200 | 1400 | 1600 | 1800 | 2000 | 2200 |
| Time (ms) | 206 | 259 | 320 | 374 | 431 | 485 | 534 | 593 |
| Energy (mJ) | 14.4 | 18.1 | 22.4 | 26 | 30.2 | 34 | 38 | 41.5 |

**TABLE 5.** Energy consumption per drone (computation and communication) over one minute for various swarm sizes.

| Number of drones | 10 | 20 | 30 | 40 | 50 | 60 | 80 | 100 |
|---|---|---|---|---|---|---|---|---|
| Energy (J) | 0.5 | 0.9 | 1.3 | 1.7 | 2.1 | 2.5 | 3.3 | 4.2 |

**TABLE 6. Comparison of our system with commercial alternatives across cost, weight, and onboard capabilities. Most companies provide mapping functionalities as plugins – additional sensor platforms or software; thus, the table groups drone(D)-plugins(P) from the same manufacturer in rows separated by a horizontal continuous line. The total cost and weight is the sum of each group. The following acronyms are used: Subscription-Based (S-B), Software Solution (S-S).**

| Cat. | Product | Cost (€) | Weight | Onboard | Sensor Type | Swarm Capability | GNSS-Denied Operation |
|---|---|---|---|---|---|---|---|
| P | DJI Zenmuse L | 13.5 k | 900 g | ✓ | LiDAR | ✗ | ✗ |
| P | DJI Terra | S-B | S-S | ✗ | - | ✗ | ✗ |
| D | DJI M600 | 5.2 k | 9.5 kg | ✗ | Camera | ✗ | ✗ |
| D | DJI M300 | 10 k | 6.3 kg | ✗ | Camera | ✗ | ✗ |
| D | Q. Sys. VTOL Trinity | 20 k | 5 kg | ✓ | Camera | ✗ | ✗ |
| P | Q. Sys. Qube 640 | 73 k | 949 g | - | LiDAR | ✗ | ✗ |
| D | ModalAI Starling 2 | 3 k | 500 g | ✗ | Stereo Cameras + ToF | ✗ | ✓ |
| D | Flyability Elios 3 | 35 k | 1.6 kg | ✓ | Cameras + LiDAR | ✗ | ✓ |
| D | Skydio X10 | - | 1.25 kg | ✓ | Cameras + Thermal | ✗ | ✓ |
| P | Skydio 3D Scan | S-B | S-S | ✗ | - | ✗ | ✓ |
| D | Crazyflie 2.1 | 270 | 30 g | ✗ | - | ✓ | ✓ |
| P | **This work** | 10 | 16 g | ✓ | 4× ToF | ✓ | ✓ |

The prices and feature comparisons presented in this table are estimations derived from publicly available information as of March 2025. They are intended for informational purposes only and should not be interpreted as official statements or commitments from the respective companies. Actual product specifications, capabilities, and pricing may vary. For the most accurate and up-to-date information, please consult the official sources or authorized representatives of the respective companies.

the drone energy increases roughly linearly from 0.5 J to 4.2 J. However, this amount is still two orders of magnitude lower than the motors' consumption [23]. This shows that the extra energy for computation and communication is negligible, keeping total power per drone nearly constant and enhancing scalability.

## VIII. DISCUSSION

So far, accurate collaborative SLAM has been a prerogative of powerful and expensive robotic platforms such as GPUs-based ones, limiting swarm formations to few units due to the high volume of exchanged data necessary for loop closure and map alignment [15]. Moreover, the majority of existing SLAM systems rely on external infrastructure support for localization and computation [11], [29], such as GNSS or motion capture systems.

Recent SoA works on onboard and decentralized solutions mainly focus on maximizing the mapping accuracy but do not address latency and scalability [6], [45]. Despite relying on high-end platforms, recent works reach an average global map update in the range between 0.1 s to 7 s [29], [31]. Moreover, such a large data volume bounds the minimum latency, which in some cases reaches up to 5 s due to computation and data transmission overhead [32]. Despite these powerful computational platforms, more similar to a server than an embedded system, the CPU load level reaches a mean of 93% [32] or half of the total load for autonomous car competitions [30] only to perform 2D mapping. Under these circumstances, SoA works on distributed and collaborative SLAM [17] feature a mapping error in the range of 20 cm to 25 cm for indoor operation. We achieved a comparable error of about 30 cm, but our system requires less than 1.5 MB of RAM and a sensor setup that costs about $20.

In our experiments, the drones are aware of their initial take-off locations which are pre-programmed. Although initial robot positions might not be known in all real-world applications (e.g., in cases of random deployment), our work focuses on inter-drone collaboration and maintaining the global map's consistency over time. To tackle the initial localization challenge, our system can be integrated with existing solutions; for instance, the authors in [42] propose a relative localization algorithm based on pairwise UWB range measurements.

### A. MAXIMUM MAPPABLE AREA

The size of the pose graph each drone stores and optimizes depends on the distance it traveled (i.e., internal poses). However, the memory footprint PGO requires depends on both nodes and edges in the graph, the latter being impacted by the number of loop closures. Thus, the maximum graph size that can be optimized depends on the size of the environment and its geometry - paired with how many loop closures arise. For example, an environment where a robot acquires 1400 poses and performs five loop closures results in ∼250 ms execution time onboard a GAP9 SoC. The largest graph an individual drone can optimize has about 3000 poses, within the 1.5 MB constraint, corresponding to an environment of about 400 $m^2$. Thanks to our exploration strategy promoting the distribution of drones across distinct regions, the distributed C-SLAM solution scales almost linearly with the number of robots. Moreover, the maximum graph capacity aligns with the drone's maximum flight time of 7 min [23], which requires storing about 3150 poses at an augmented pose rate of 7.5 Hz - as used in this work. Thanks to our system's energy-efficient, lightweight design, deploying it on slightly larger drones with increased battery capacity proportionally extends mapping time and coverage with negligible extra payload weight or computational overhead – paving the way for longer-endurance missions without compromising real-time onboard processing.

### B. COMPARISON WITH LIDAR-BASED MAPPING

Although the ToF sensors we employ in this work are lightweight, energy-efficient, and cost-effective, conventional LiDARs maintain superiority in terms of accuracy. Primarily, LiDARs commonly feature an extended opera-

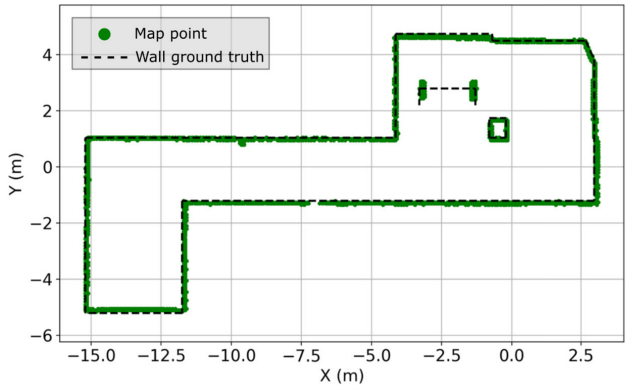



**FIGURE 13.** The map is obtained by mapping the same environment from Section VII-B with a Unitree A1 Robot Dog equipped with a LiDAR and using Cartographer as SLAM engine.

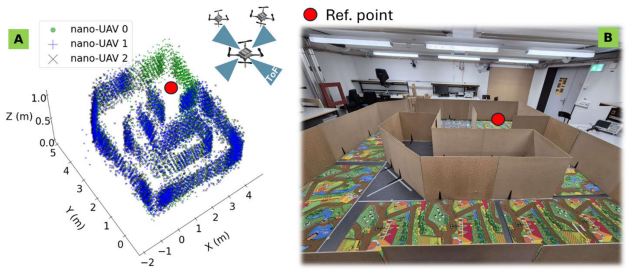



**FIGURE 14.** (A) The generated 3D map acquired from three nano-UAVs. (B) The environment mapped in (A).

tional range, facilitating the mapping of distant environmental regions. Conversely, our ToF depth sensor is constrained by a maximum range of 4 m. Moreover, LiDARs frequently boast superior angular resolution, leading to measurement precision that is less dependent on distance magnitude, allowing for higher accuracy scans and scan-matching. However, to precisely quantify the differences in mapping accuracy, we map again the environment from Figure 10 using a ground legged robot (i.e., Unitree A1) equipped with Hokuyo UTM-30LX-EW 2D LiDAR configured at 20 Hz. The processing is conducted on an Intel NUC10i7FNKN with $10^{th}$ gen 6-core i7 CPU, alongside a NVIDIA Jetson Xavier NX. In this configuration, the computing power is 45 W, total RAM memory 32 GB, and the computing cost in the range of $1500 USD. In contrast, our collaborative SLAM requires orders of magnitude lower hardware specifications, 100 mW, 1.5 MB, $10 USD, respectively. The same applies to the sensor platform. To complete the price-wise comparison with other commercial solutions, we also provide Table 6, which clarifies the cost-effectiveness and uniqueness of our fully distributed onboard mapping system.

Figure 13 shows the map obtained with the Robot Dog-based system and Cartographer,[1] a SoA solution for real-time SLAM. Note that the dashed ground truth line, which is not mapped, corresponds to the table plate located at a height of 1 m, outside the LiDAR's field of view. Despite the strong difference in computational and sensing power, the resulted map has an accuracy of 10.05 cm, which is in the same order of magnitude as the result achieved by our mapping system (i.e., 29.7 cm). However, the accuracy difference is mainly impacted by the lower precision of the optical flow-based odometry used by the nano-UAVs, which is inferior to the odometry of the ground robots. More in detail, the legged robot's odometry accuracy is mainly impacted by encoders, while for the optical-flow-based odometry used by our nano-UAVs, the errors depend on multiple factors such as illumination, reflections, texture, which vary within the testing area. These errors are therefore not deterministic and more difficult to correct or model.

[1] github.com/cartographer-project/cartographer

### C. ENABLING MAPPING IN 3D

The C-SLAM pipeline we propose supports only 2D mapping and 2D loop closure, an optimal tradeoff between mapping accuracy and memory footprint. However, the raw depth measurements from the depth sensors are acquired in matrix form, with a total of 64 pixels distributed in a resolution of $8 \times 8$. While our 2D approach reduces the measurements from matrix to row format, a 3D map can be obtained by projecting all matrix measurements in the world frame. In this setup, the loop closure and ICP are still performed in 2D, while the generated graph map will result in a 3D perspective. An example is provided in Figure 14, where a maze environment mapped concurrently with three nano-UAVs and its respective 3D map are depicted. In this configuration, the scan size becomes 6× larger, due to the necessity of storing and transmitting 64 pixels for each sensor. Table 7 shows the structure of a depth frame for both scenarios and how the dimension changes when performing 3D mapping compared to the 2D scenario.

**TABLE 7.** Memory structure of a depth frame.

| | 2D Mapping | | 3D Mapping | |
|---|---|---|---|---|
| | Format | Size | Format | Size |
| Pose ID | *int32* | 4 B | *int32* | 4 B |
| Timestamp | *int32* | 4 B | *int32* | 4 B |
| Pose | $3 \times$ *float32* | 12 B | $3 \times$ *float32* | 12 B |
| Depth | $4 \times 8 \times$ *int16* | 64 B | $4 \times 64 \times$ *int16* | 512 B |
| **Total** | | 84 B | | 532 B |

## IX. CONCLUSION

This paper introduces a decentralized and lightweight collaborative SLAM approach, tailored for low-cost robotic platforms, including those with extremely limited hardware resources. Our solution supports the coordination of large robot swarms, effectively managing the complexities associated with scalability by optimizing the data traffic and the exploration strategy. Moreover, by reducing the sensing and computation costs to approximately $20 per unit, our solution offers a cost-effective alternative to traditional high-end

SLAM systems, opening up new possibilities for deploying SLAM in budget-constrained applications. To efficiently manage the computation required for SLAM, our system distributes the intensive tasks across RISC-V parallel low-power platforms, operating onboard each individual robot in real-time. Furthermore, the collaborative exploration strategy that we propose decreases the coverage time while ensuring accurate mapping.

We evaluated our solution in various environments using a swarm of centimeter-size UAVs weighing only 46 g, achieving a mapping accuracy below 30 cm. This level of precision is comparable to high-end SoA solutions, yet it comes with drastically reduced cost, memory, and computational requirements.

## ACKNOWLEDGMENT

The authors thank Davide Plozza and Cristian Cioflan for their support during field tests. The GAP9 Deck V2 is designed by Victor Javier Kartsch Morinigo and Hanna Müller with the support of Greenwaves Technologies.

## REFERENCES

[1] Y. Zhou, B. Rao, and W. Wang, "UAV swarm intelligence: Recent advances and future trends," *IEEE Access*, vol. 8, pp. 183856–183878, 2020.

[2] B. Zhou, H. Xu, and S. Shen, "RACER: Rapid collaborative exploration with a decentralized multi-UAV system," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1816–1835, Mar. 2023.

[3] B. Amjad, Q. Z. Ahmed, P. I. Lazaridis, M. Hafeez, F. A. Khan, and Z. D. Zaharis, "Radio SLAM: A review on radio-based simultaneous localization and mapping," *IEEE Access*, vol. 11, pp. 9260–9278, 2023.

[4] Q. M. Rahman, P. Corke, and F. Dayoub, "Run-time monitoring of machine learning for robotic perception: A survey of emerging trends," *IEEE Access*, vol. 9, pp. 20067–20075, 2021.

[5] M. Dorigo, G. Theraulaz, and V. Trianni, "Reflections on the future of swarm robotics," *Sci. Robot.*, vol. 5, no. 49, p. 4385, Dec. 2020.

[6] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1686–1705, Mar. 2023.

[7] P.-Y. Lajoie and G. Beltrame, "Swarm-SLAM: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems," *IEEE Robot. Autom. Lett.*, vol. 9, no. 1, pp. 475–482, Jan. 2024.

[8] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Sci. Robot.*, vol. 4, no. 35, p. 9710, Oct. 2019.

[9] E. Soria, F. Schiano, and D. Floreano, "Predictive control of aerial swarms in cluttered environments," *Nature Mach. Intell.*, vol. 3, no. 6, pp. 545–554, May 2021.

[10] V. S. Varadharajan, D. St-Onge, B. Adams, and G. Beltrame, "Swarm relays: Distributed self-healing Ground-and-Air connectivity chains," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5347–5354, Oct. 2020.

[11] R. J. Amala Arokia Nathan, I. Kurmi, and O. Bimber, "Drone swarm strategy for the detection and tracking of occluded targets in complex environments," *Commun. Eng.*, vol. 2, no. 1, p. 55, Aug. 2023.

[12] V. Krátký, A. Alcántara, J. Capitán, P. Štepán, M. Saska, and A. Ollero, "Autonomous aerial filming with distributed lighting by a team of unmanned aerial vehicles," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7580–7587, Oct. 2021.

[13] Y. Tian, Y. Chang, L. Quang, A. Schang, C. Nieto-Granda, J. P. How, and L. Carlone, "Resilient and distributed multi-robot visual SLAM: Datasets, experiments, and lessons learned," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 11027–11034.

[14] G. C. H. E. de Croon, J. J. G. Dupeyroux, S. B. Fuller, and J. A. R. Marshall, "Insect-inspired AI for autonomous robots," *Sci. Robot.*, vol. 7, no. 67, p. 6334, Jun. 2022.

[15] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, "Swarm of micro flying robots in the wild," *Sci. Robot.*, vol. 7, no. 66, p. 5954, May 2022.

[16] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "EGO-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 4101–4107.

[17] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2022–2038, Aug. 2022.

[18] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame, "DOOR-SLAM: Distributed, online, and outlier resilient SLAM for robotic teams," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1656–1663, Apr. 2020.

[19] P. E. Dupont, B. J. Nelson, M. Goldfarb, B. Hannaford, A. Menciassi, M. K. O'Malley, N. Simaan, P. Valdastri, and G.-Z. Yang, "A decade retrospective of medical robotics research from 2010 to 2020," *Sci. Robot.*, vol. 6, no. 60, p. 8017, Nov. 2021.

[20] P. Arm, G. Waibel, J. Preisig, T. Tuna, R. Zhou, V. Bickel, G. Ligeza, T. Miki, F. Kehl, H. Kolvenbach, and M. Hutter, "Scientific exploration of challenging planetary analog environments with a team of legged robots," *Sci. Robot.*, vol. 8, no. 80, p. 9548, Jul. 2023.

[21] V. Iyer, A. Najafi, J. James, S. Fuller, and S. Gollakota, "Wireless steerable vision for live insects and insect-scale robots," *Sci. Robot.*, vol. 5, no. 44, p. 839, Jul. 2020.

[22] A. Grau, M. Indri, L. Lo Bello, and T. Sauter, "Robots in industry: The past, present, and future of a growing collaboration with humans," *IEEE Ind. Electron. Mag.*, vol. 15, no. 1, pp. 50–61, Mar. 2021.

[23] H. Müller, V. Niculescu, T. Polonelli, M. Magno, and L. Benini, "Robust and efficient depth-based obstacle avoidance for autonomous miniaturized UAVs," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4935–4951, Dec. 2023.

[24] M. S. Talamali, A. Saha, J. A. R. Marshall, and A. Reina, "When less is more: Robot swarms adapt better to changes with constrained communication," *Sci. Robot.*, vol. 6, no. 56, p. 1416, Jul. 2021.

[25] F. Berlinger, M. Gauci, and R. Nagpal, "Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm," *Sci. Robot.*, vol. 6, no. 50, p. 8668, Jan. 2021.

[26] K. Ebadi et al., "Present and future of SLAM in extreme environments: The DARPA SubT challenge," *IEEE Trans. Robot.*, vol. 40, pp. 936–959, 2024.

[27] Y. Cai, F. Kong, Y. Ren, F. Zhu, J. Lin, and F. Zhang, "Occupancy grid mapping without ray-casting for high-resolution LiDAR sensors," *IEEE Trans. Robot.*, vol. 40, pp. 172–192, 2024.

[28] G. SONUGÜR, "A review of quadrotor UAV: Control and SLAM methodologies ranging from conventional to innovative approaches," *Robot. Auto. Syst.*, vol. 161, Mar. 2023, Art. no. 104342.

[29] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 1689–1696.

[30] N. Baumann, E. Ghignone, J. Kühne, N. Bastuck, J. Becker, N. Imholz, T. Kränzlin, T. Y. Lim, M. Lötscher, L. Schwarzenbach, L. Tognoni, C. Vogt, A. Carron, and M. Magno, "ForzaETH race stack—Scaled autonomous head-to-head racing on fully commercial off-the-shelf hardware," *J. Field Robot.*, vol. 42, no. 4, pp. 1037–1079, Jun. 2025.

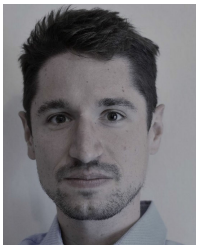
[31] V. Niculescu, T. Polonelli, M. Magno, and L. Benini, "NanoSLAM: Enabling fully onboard SLAM for tiny robots," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13584–13607, Apr. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10343110/

[32] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, "Edge robotics: Edge-Computing-Accelerated multirobot simultaneous localization and mapping," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 14087–14102, Aug. 2022.

[33] T. Polonelli, C. Feldmann, V. Niculescu, H. Müller, M. Magno, and L. Benini, "Towards robust and efficient on-board mapping for autonomous miniaturized UAVs," in *Proc. 9th Int. Workshop Adv. Sensors Interfaces (IWASI)*, Jun. 2023, pp. 9–14.

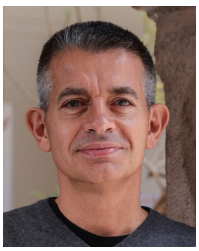
[34] M. A. Jamshed, K. Ali, Q. H. Abbasi, M. A. Imran, and M. Ur-Rehman, "Challenges, applications, and future of wireless sensors in Internet of Things: A review," *IEEE Sensors J.*, vol. 22, no. 6, pp. 5482–5494, Mar. 2022.
[35] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In defense of point-to-point ICP-simple, accurate, and robust registration if done the right way," *IEEE Robot. Autom. Lett.*, vol. 8, no. 2, pp. 1029–1036, Feb. 2023.
[36] S. Fuller, Z. Yu, and Y. P. Talwekar, "A gyroscope-free visual-inertial flight control and wind sensing system for 10-mg robots," *Sci. Robot.*, vol. 7, no. 72, Nov. 2022, Art. no. abq8184.
[37] T. Fan and T. D. Murphey, "Majorization minimization methods for distributed pose graph optimization," *IEEE Trans. Robot.*, vol. 40, pp. 22–42, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10286063/
[38] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone, "Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1127–1134, Apr. 2020.
[39] A. Mueller, "Modern robotics: Mechanics, planning, and control [Bookshelf]," *IEEE Control Syst. Mag.*, vol. 39, no. 6, pp. 100–102, Dec. 2019.
[40] F. Shan, J. Zeng, Z. Li, J. Luo, and W. Wu, "Ultra-wideband swarm ranging," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.
[41] B. P. Duisterhof, S. Li, J. Burgués, V. J. Reddi, and G. C. H. E. de Croon, "Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 9099–9106.
[42] D. Schindler, V. Niculescu, T. Polonelli, D. Palossi, L. Benini, and M. Magno, "A relative infrastructure-less localization algorithm for decentralized and autonomous swarm formation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2023, pp. 5288–5295.
[43] H. Müller, V. Kartsch, and L. Benini, "GAP9Shield: A 150GOPS AI-capable ultra-low power module for vision and ranging applications on nano-drones," in *Proc. Eur. Robot. Forum*, 2024, pp. 292–297.
[44] D. Rossi, F. Conti, M. Eggiman, A. D. Mauro, G. Tagliavini, S. Mach, M. Guermandi, A. Pullini, I. Loi, J. Chen, E. Flamand, and L. Benini, "Vega: A ten-core SoC for IoT endnodes with DNN acceleration and cognitive wake-up from MRAM-based state-retentive sleep mode," *IEEE J. Solid-State Circuits*, vol. 57, no. 1, pp. 127–139, Jan. 2022.
[45] S. Zhong, Y. Qi, Z. Chen, J. Wu, H. Chen, and M. Liu, "DCL-SLAM: A distributed collaborative LiDAR SLAM framework for a robotic swarm," *IEEE Sensors J.*, vol. 24, no. 4, pp. 4786–4797, Feb. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10375928/

**VLAD NICULESCU** received the master's degree in robotics, systems, and control from ETH Zürich, in 2019, where he is currently pursuing the Ph.D. degree in electrical engineering with the Integrated Systems Laboratory. During his bachelor's and master's period, he has competed more than ten international student competitions. His current research interests include developing localization, mapping, and autonomous navigation algorithms that target ultra-low-power platforms which can operate onboard nano-drones. He was the Electrical Lead of the Student Project Swissloop, which won the second place; and the Innovation Award in the SpaceX Hyperloop Pod Competition 2019.



**TOMMASO POLONELLI** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Bologna, Bologna, Italy, in 2017 and 2020, respectively. He is currently a Postdoctoral Researcher with ETH Zürich, Zürich, Switzerland. He has collaborated with several universities and research centers, such as University College Cork, Cork, Ireland, and Imperial College London, London, U.K. He has authored over 40 papers in international journals and conferences. His research interests include wireless sensor networks, the IoT, autonomous unmanned vehicles, power management techniques, structural health monitoring, and the design of ultra-low power battery-supplied devices with onboard intelligence.

**MICHELE MAGNO** (Senior Member, IEEE) received the master's and Ph.D. degrees in electronic engineering from the University of Bologna, Italy, in 2004 and 2010, respectively. Since 2013, he has been with ETH Zürich, Switzerland, and has become a Visiting Lecturer or a Professor with several universities, such as the University of Nice Sophia, France; Enssat, Lannion, France; the University of Bologna; and Mid University Sweden; where he is currently a Full Visiting Professor with the Electrical Engineering Department. He is currently a Senior Scientist with the Department of Information Technology and Electrical Engineering (D-ITET), ETH Zürich. Since 2020, he has been leading the D-ITET Center for project-based learning, ETH Zürich. He has authored more than 220 papers in international journals and conferences. His current research interests include smart sensing, low-power machine learning, wireless sensor networks, wearable devices, energy harvesting, low-power management techniques, and extension of the lifetime of batteries-operating devices. He is an ACM Member. Some of his publications were awarded as best papers awards at IEEE conferences. He also received awards for industrial projects or patents.



**LUCA BENINI** (Fellow, IEEE) received the Ph.D. degree from Stanford University. He is currently the Chair of digital circuits and systems with ETH Zürich and a Full Professor with the Universitá di Bologna. His research interests include energy-efficient parallel computing systems, smart sensing micro-systems, and machine learning hardware. He is a fellow of ACM and a member of the Academia Europaea. He was a recipient of the 2016 IEEE CAS Mac Van Valkenburg Award, the 2020 EDAA Achievement Award, and the 2020 ACM/IEEE A. Richard Newton Award.