COLLECTIVE BEHAVIOR

Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment

K. N. McGuire[1]*, C. De Wagter[1], K. Tuyls[2], H. J. Kappen[3], G. C. H. E. de Croon[1]*

Swarms of tiny flying robots hold great potential for exploring unknown, indoor environments. Their small size allows them to move in narrow spaces, and their light weight makes them safe for operating around humans. Until now, this task has been out of reach due to the lack of adequate navigation strategies. The absence of external infrastructure implies that any positioning attempts must be performed by the robots themselves. State-of-the-art solutions, such as simultaneous localization and mapping, are still too resource demanding. This article presents the swarm gradient bug algorithm (SGBA), a minimal navigation solution that allows a swarm of tiny flying robots to autonomously explore an unknown environment and subsequently come back to the departure point. SGBA maximizes coverage by having robots travel in different directions away from the departure point. The robots navigate the environment and deal with static obstacles on the fly by means of visual odometry and wall-following behaviors. Moreover, they communicate with each other to avoid collisions and maximize search efficiency. To come back to the departure point, the robots perform a gradient search toward a home beacon. We studied the collective aspects of SGBA, demonstrating that it allows a group of 33-g commercial off-the-shelf quadrotors to successfully explore a real-world environment. The application potential is illustrated by a proof-of-concept search-and-rescue mission in which the robots captured images to find "victims" in an office environment. The developed algorithms generalize to other robot types and lay the basis for tackling other similarly complex missions with robot swarms in the future.

INTRODUCTION

Swarms of tiny autonomous flying robots hold great promise. Tiny flying robots can move in narrow spaces, can be so cheap that they may become disposable, and are safe in the presence of humans (*1*, *2*). Moreover, whereas the individual robots may be inherently limited in their abilities both in terms of cognition and in terms of actions, together they may solve very complex problems. This kind of problem-solving ability is abundant in nature. Two well-known examples are the shortest path finding by swarms of ants (*3*) and collective selection of profitable food resources by honeybees through waggle dances (*4*).

The core principle of swarm robotics is that the individual robots obey relatively simple control rules, merely based on their local sensory inputs and local communication with their neighbors. This principle fits well with the limited resources of tiny robots. Moreover, not relying on any central processing promises a high robustness of the system. A single failing robot will not endanger task execution because its role will be fulfilled by one of the many other robots. In addition, together, small robots will potentially be able to perform tasks—such as surveillance, construction, or exploration—quicker and more robustly. In the past few decades, a large body of research investigating swarm robotics has formed. For instance, in the Swarm-Bots project, controllers have been evolved for small driving robots to complete tasks such as gap crossing, which required them to attach themselves to each other to form a bridge, and movement of objects bigger than each individual (*5*). Moreover, swarms of robots have been demonstrated in applications ranging from constructing small preplanned structures (*6*) to forming shapes with their own bodies (*7*, *8*) and to performing tasks such as dispersion, aggregation, and surveillance (*9*).

Concerning flying robot swarms, the major challenge lies in achieving autonomous robot navigation and coordination between the robots in real-world environments. There have been impressive shows with many simultaneously flying robots, such as Intel's Shooting Star drones (*10*), which were used in the 2017 Super Bowl halftime and the 2018 Winter Olympics. However, those robots purely followed preprogrammed Global Positioning System (GPS)–based trajectories, so they did not make local decisions based on their surroundings. In contrast, in (*11*, *12*) and, more recently, (*13*), swarms of flying robots performed coordinated swarming behaviors together in outdoor environments. In the latter study, the main behavioral parameters were optimized with an evolutionary process such that robots stayed together, even in the presence of no-fly zones. The studies (*11*–*13*) all still crucially relied on GPS. The flying robots communicated their GPS locations to each other to determine the relative locations to other robots that serve as input to the local controllers. In all above studies, the swarms essentially flew in open environments or, in the case of (*13*), had access to a global map of no-fly zones.

Navigation of a swarm of tiny flying robots in a cluttered, GPS-denied environment has been an unsolved problem. The major challenge derives from the highly restricted nature of these tiny robots. The mainstream solution to navigation consists of simultaneous localization and mapping (SLAM) based on camera images (*14*) or laser range finders (*15*). However, typical, metric SLAM methods make detailed three-dimensional (3D) maps, which is very demanding in terms of computational and memory resources. State-of-the-art SLAM methods, like large-scale direct monocular SLAM (LSD-SLAM) (*16*), often need to be computed by an external ground station computer (*17*). Multirobot SLAM, in which a group of robots jointly creates and maintains a shared map of the environment (*18*), places an

[1]Faculty of Aerospace Engineering, Delft University of Technology, Delft, Netherlands. [2]Department of Computer Science, University of Liverpool, Liverpool, UK. [3]Faculty of Science, Radboud University, Nijmegen, Netherlands.
*Corresponding author. Email: k.n.mcguire@tudelft.nl (K.N.M.); g.c.h.e.decroon@tudelft.nl (G.C.H.E.d.C.)

additional load on the communication bandwidth. One can also opt for the slightly more efficient visual inertial odometry (*19*). However, this is subject to drift. To illustrate the challenge of navigating with tiny flying robots, we show (Fig. 1) the processing power of the robot used in our experiments (Bitcraze's Crazyflie 2.0) beside two recent, state-of-the-art–embedded processing units used in SLAM approaches. Flying robots like the Crazyflie use about 7 W to fly. To not substantially affect their flight time, the processing should therefore use only a fraction of this power. The Crazyflie carries an STM32F4 microprocessor, with a clock speed of 168 MHz and 192 kB of random-access memory (RAM). Typical state-of-the-art robots used for autonomous flight [e.g., (*20*, *21*)] use processors like the NVIDIA TX2, which has a six-core central processing unit, each with a clock speed of 2 GHz, a 256-core NVIDIA graphics processing unit, and 8 GB of RAM. Hence, we need to solve the navigation problem with orders of magnitude less memory and processor speed. This calls for a completely different navigation strategy.

One potential approach to efficient navigation is to draw inspiration from biology, for instance, by looking at honeybee navigation strategies. Honeybees navigate by combining path integration with landmark recognition (*22*). Path integration is well understood and can be implemented with very limited systems, as in the recently presented AntBot (*23*). Whereas walking insects can count their steps for path integration, flying insects rely more heavily on the integration of optical flow (*24*). Path integration alone does not suffice for navigation because it drifts over time. This drift can be cancelled by means of landmark detection and visual homing, but it is not obvious how landmark recognition is performed by biological systems. The dominant model is the snapshot model (*25*), in which pictures are stored of the surroundings and later compared with the visual inputs. Unfortunately, current implementations of landmark recognition still require substantial processing and memory [e.g., (*26*)], making it unsuitable for navigation by tiny robots. Moreover, they mostly thrive on texture-rich environments, which are commonly found in nature but not in repetitive man-made environments.

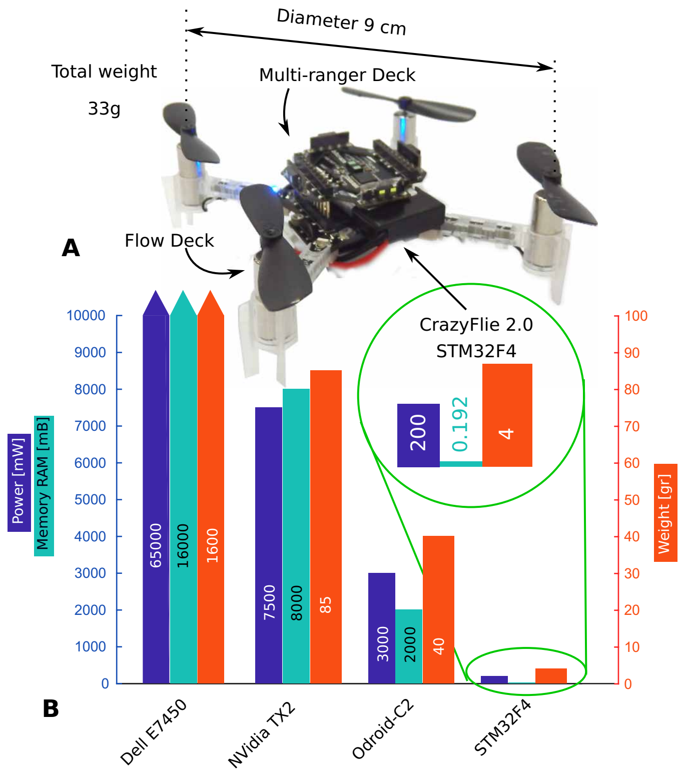


Fig. 1. Hardware specifications and comparison. (**A**) Crazyflie 2.0 with the flow and multi-ranger expansion decks and (**B**) the autopilot (STM32F4) compared with the specifications of the NVIDIA TX2, the Odroid-C2, and a laptop (Dell Latitude E7450). Note that the Dell specifications do not fit within the chart (as indicated with the top triangles).

Biological systems provide interesting suggestions for arriving at the minimal requirements for navigation. The maps created by metric SLAM can be used for navigating from any point to any other point in the map. The navigation strategies followed by insects suggest that it may be possible to save on computation and memory by requiring less accurate maps. Biological navigation strategies show a parallel with topological SLAM (*27*), in which a robot only stores important landmarks and their relations in terms of distance and direction. This no longer allows a robot to travel anywhere in the explored space with high accuracy, but this may not be necessary for successful behavior. In some cases, it may only be important to explore and come back to the "nest," i.e., to only perform accurate homing. A navigation strategy that only demands homing and does not rely on computationally complex visual navigation has strong potential for downscaling to tiny robots.

A minimal navigation solution

The main contribution of this article is a minimal autonomous navigation solution for a swarm of tiny flying robots to explore an unknown, unstructured environment and subsequently to come back to the departing point. "Exploration" here means to move through as large a part of an unknown environment as possible, with a goal to gather application-dependent information. The proposed navigation solution was implemented in a swarm of tiny flying robots and shown to work in a large real-world indoor environment that has no external infrastructure for exact positioning. Moreover, in the same environment, we illustrate how the solution enabled a specific proof-of-concept search-and-rescue exploration mission, in which the swarm gathered images to find "victims" in the environment.

Particularly, we introduce the swarm gradient bug algorithm (SGBA). As the name suggests, the method is inspired by "bug algorithms" (*28*–*30*), which originated as simple maze-solving algorithms. The core concept is that navigating from A to B is performed not by planning in a global map with known obstacles but by reacting to obstacles as they come within range of the sensors. This way of dealing with obstacles results in a highly computationally efficient navigation. However, existing bug algorithms in the literature remain rather theoretical and are not suitable for application to navigation in real, GPS-denied environments because they typically rely on either a known global position or perfect odometry. For example, in (*31*, *32*), real-world robots used their wheel odometry for navigation within an indoor environment; nevertheless, the testing environments were too small to experience the full extent of the possible odometry drift. A flying robot typically relies on visual odometry and, due to the vibrations and texture dependence, is even more prone to odometry inaccuracies than a driving robot. When realistic levels of odometry drift were introduced, the navigation performance of bug algorithms from the literature dropped steeply (*33*).

The bug algorithm proposed in this article, SGBA, departs substantially from existing bug algorithms because it has been designed explicitly for allowing a swarm of tiny robots to explore a real-world, GPS-denied environment. Figure 2B shows the main concept: A swarm of robots departs their base station for outbound travel. Each robot has a different preferential direction toward which it will try to go. When robots encounter obstacles, they follow the obstacles' contours. This process is called "wall following" in the bug algorithm literature. When a robot's preferred direction is free of obstacles again, it will continue to follow its preferred direction. As soon as the robot's battery is around 60%, it starts inbound travel (Fig. 2C). To come back to the original location, the robots use a mix of (coarse) odometry and, on longer time scales, an observable gradient to the base station. In our experiments, we used the received signal strength intensity (RSSI) to a radio beacon located at the base station. Because bug algorithms do not make maps, there is a danger that they get stuck in loops without realizing it. For instance, this can happen in rooms where there is only one way out, which may be missed when the robot leaves the wall in its preferred direction, just to navigate to the opposite wall again. Hence, during both outbound and inbound travel, the odometry was also exploited to detect short-term loops that could result in robots getting stuck in a particular part of the environment. The robots also needed to avoid each other and to communicate their desired direction to each other. In the experiments, we used wireless onboard inter-robot communication to both these ends. Specifically, for the intra-swarm collision avoidance, the inter-robot RSSI was used instead of communicating a global position (which is not known by the robots). Moreover, when robots noticed the presence of other robots in the direction of their preferred heading, they adapted their preference, thus enhancing the exploration for the outbound flight.

A simplified version of the finite state machine (FSM) of SGBA can be found in Fig. 2A. The SGBA method and the FSM are presented in more detail in Materials and Methods. The innovation of SGBA lies in its suitability for the real-world properties of tiny, exceptionally computationally restricted robots and in the combination of the various subcomponents. Many of the subcomponents themselves have already been proposed in the literature. For instance, traveling toward a wireless beacon with a bug algorithm was also proposed in (*34*, *35*). However, the proposed methods in those studies were too sensitive to the real-world noise of RSSI measurements. Because of the difficulties of real-world interference, refraction, and scattering of the signal, the experiments eventually involved an infrared beacon instead, which was visible from all locations in the environment. This setup would not be useful in a real scenario. The work in (*36*) used the gradient of real-world, noisy RSSI values to guide exploration on a real robot (without a bug algorithm type of behavior). However, the platform still required a full SLAM method on board because the precise positions of the RSSI samples were needed to estimate the location of the Wi-Fi source. In contrast to these methods, we have implemented a home beacon search tactic that dealt with real-world, noisy, 2.4-GHz, Wi-Fi RSSI values and did not rely on exact positioning. Another asset is SGBA's use of multiple robots. The idea of using multiple robots for bug algorithms was first forwarded and studied in simulation in (*37*). However, they used it to explore the local obstacle boundary and not for efficiently exploring the environment. The swarming mechanisms of SGBA involve (i) imprinting of different initial preferential directions, (ii) collision avoidance, and (iii) adaptation of preferential directions when robots notice that their preferred direction overlaps too much with that of another robot. More elaborate swarming mechanisms are possible, but the results show that these straightforward mechanisms, which do not require accurate relative positions between the robots, already significantly increase exploration efficiency.

From the explanation above, it can be deduced that SGBA requires five main functionalities: (i) following a given direction; (ii) wall following; (iii) odometry; (iv) inter-robot detection, communication, and avoidance; and (v) a gradient-based search toward the departure point. Although we mainly focused on flying robots in this article, the five functionalities can be implemented with various types of hardware and software on different types of robots. We used flying robots for the real-world experiments and driving robots in

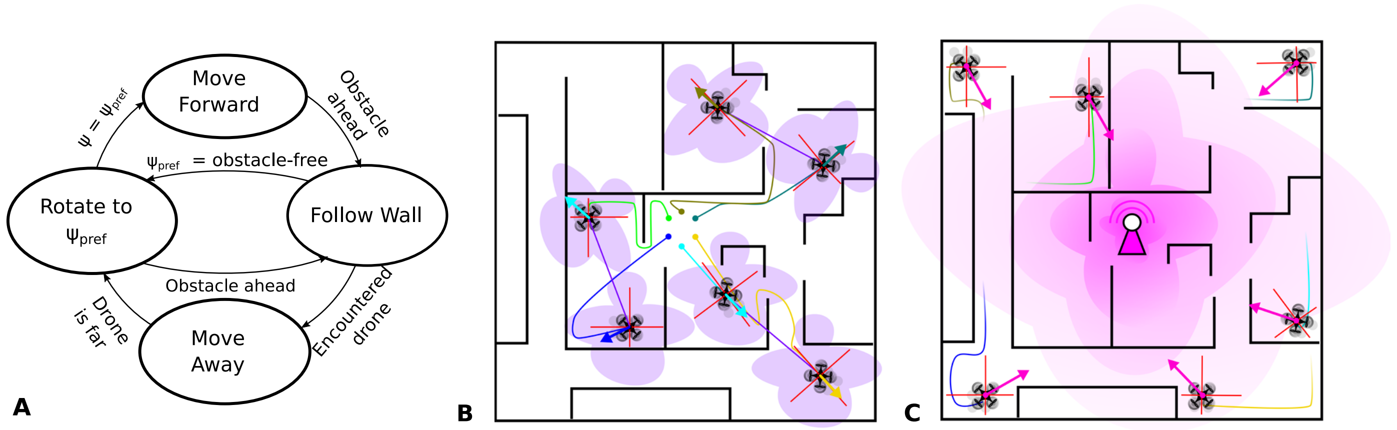


Fig. 2. Main concept of the SGBA. (**A**) A simplified state machine of SGBA derived from the one presented in Materials and Methods. (**B**) Outbound travel of SGBA. The purple shading illustrates the local signal strength around each drone used for intra-swarm avoidance. (**C**) Inbound travel. The pink shading represents the signal strength of the wireless beacon at the ground station to which the drones navigate. The interswarm avoidance is still active on the inbound flight but is not depicted. The fuchsia arrow at each drone's position illustrates the robot's estimated direction to the beacon.

the simulation experiments detailed below. The difference in implementation of SGBA includes, for instance, that functionality (iii) is performed with optical flow–based odometry on the flying robot and wheel-based odometry on the driving robot. Hence, the SGBA algorithm can be applied to different types of mobile robots with limited resources, as long as they are endowed with the above required functionalities.

RESULTS

We performed both simulation and real-world experiments to gauge the performance of SGBA as an autonomous navigation solution for exploration missions. Navigation here means spreading out in the environment, covering the environment as much as possible, and coming back to the departure point. The two main performance metrics are (i) the area coverage and (ii) the return rate of the robots. With these metrics, we mainly assessed the navigational characteristics of SGBA because these are important to exploration missions in general. We varied the number of robots to investigate the advantages of the swarming aspect of SGBA. After the main simulation and real-world experiments focusing on the navigation performance, we also investigated a search-and-rescue scenario in which the robots had to find possible victims in the environment. For this scenario, the robots carried onboard cameras and secure digital (SD) cards for storing images of the environment because transmitting live streams was not feasible. When returning to the base station, the robots could upload the images to the base station, and a human end user could look at the images to find the victims. Hence, only returning robots provided useful information on the task. Because this will be the case in many real-world exploration scenarios where SGBA is a suitable method, we considered as a third performance metric, (iii) the area covered by returning robots—termed "coverage returned." In the specific search-and-rescue experiment, we evaluated whether the victims were present in the images.

Simulation experiment results

We first implemented the SGBA in simulation. The goal of the simulation experiments was to investigate the performance of the algorithm in many artificially generated environments. Moreover, in simulation, we could perform extensive experiments for gathering sufficient statistics on trends such as the relation between the performance and the number of robots. As a simulator, we have chosen ARGoS (Fig. 3A) because it has especially been developed for multirobot systems (*38*). A Robot Operating System (ROS) environment was used to connect the SGBA controller, the automatic environment generator, and the simulator together, in a similar manner as in (*33*), by using ARGoS for ROS (*39*) [all code repositories for the simulation experiments can be found in (*40*)]. In simulation, the robots were adapted ARGoS foot-bots, which were originally modeled on the MarXbot (*41*). These ground-based nonholonomic robots are different from the airborne holonomic robots used in the real-world experiments (see text S5 for an overview of how we implemented SGBA's five functionalities on the simulated foot-bot and on the flying robots used in the real-world experiments). The use of ground-based robots in the simulation experiments illustrates that SGBA can be applied to different types of robots.

The simulated robots started around the home beacon in the middle of the environment. With SGBA, each of them sequentially started moving into their preferred direction, which in this case were the angles 45°, 135°, −135°, and −45° [the modulus of the robot's identification number (ID) from 4 determines the preferred direction]. In simulation, the outbound travel lasted for 5 min. After spreading out into the environment, the simulated robots tried to head back to the home beacon within another 5 min (10 min of total simulation time), for which they used the noisy and locally perturbed RSSI of the base station beacon. Figure 3 (B and C) shows two examples of the simulated experiments with four and six robots. We experimented with 2, 4, 6, 8, and 10 robots per simulated environment. Per test configuration, 100 environments were produced with the procedural environment generator, as developed in (*33*). The coverage statistics can be found in Fig. 3D and the return rates in Fig. 3E. Despite their extremely restricted onboard resources, 10 small robots were able to explore, on average, 90% of a simulated 20 m by 20 m environment in 10 min [dark blue bar in Fig. 3D, example trajectory in Fig. 3 (B and C)].

The utility of the collective aspect of SGBA is shown by the dark blue bars in Fig. 3D; adding more robots leads to a higher coverage in the same amount of time. The trend of the bars indicates that the coverage is subject to a law of diminishing returns; adding two more robots has more effect when going from two to four robots than when going from 8 to 10 robots. The results suggest that this effect is mainly due to covering the same areas in the environment and not due to robots interfering with each other. Namely, the coverage per robot (Fig. 3D, yellow bars) and the return rate (Fig. 3E, light blue bars) did not decrease for the studied number of robots. The return rate is also of interest for the envisaged proof-of-concept search-and-rescue mission, in which the robots would store images onboard and only robots that return to the base station provide information on the task. The return rate was lower than 100%, mainly because SGBA can lead to suboptimal paths back to the base station (too slow for the total mission time). The coverage by the group of returned robots is shown in turquoise in Fig. 3D. Last, the variances of all performance characteristics became smaller for higher number of drones, showing that adding more agents increased the certainty of the outcome. This seems mainly due to the reduced effect of variations in individual performance on total coverage. For instance, with two robots, an early failing robot could halve the coverage, whereas with 10 robots, the effect would be much less noticeable, underlining swarm robustness. The logging data can be found in (*42*), and statistical tests of the simulation results can be found in text S4.1, which show that the trend of the total coverage and coverage returned are significantly related to the total number of robots. This is not the case for the coverage per robot and the return rate.

Last, we have studied the contributions of the different swarming mechanisms in SGBA: (i) sending them off in different directions, (ii) performing an avoidance maneuver when close to another robot, and (iii) changing preferential direction. All three mechanisms contributed to reducing collisions and increasing coverage. For example, for the "full" SGBA with six robots, there were, on average, 0.2 collisions per exploration trial. When sending off all robots in the same direction, this rose to 1.7 collisions. When only switching off the avoidance maneuvers, there were, on average, 1.2 collisions per trial. Text S6 contains these test results.

Real-world experiment results

Subsequently, we performed real-world experiments. The goal of these experiments was to show that SGBA works in the real world and to investigate whether the results align with the findings from

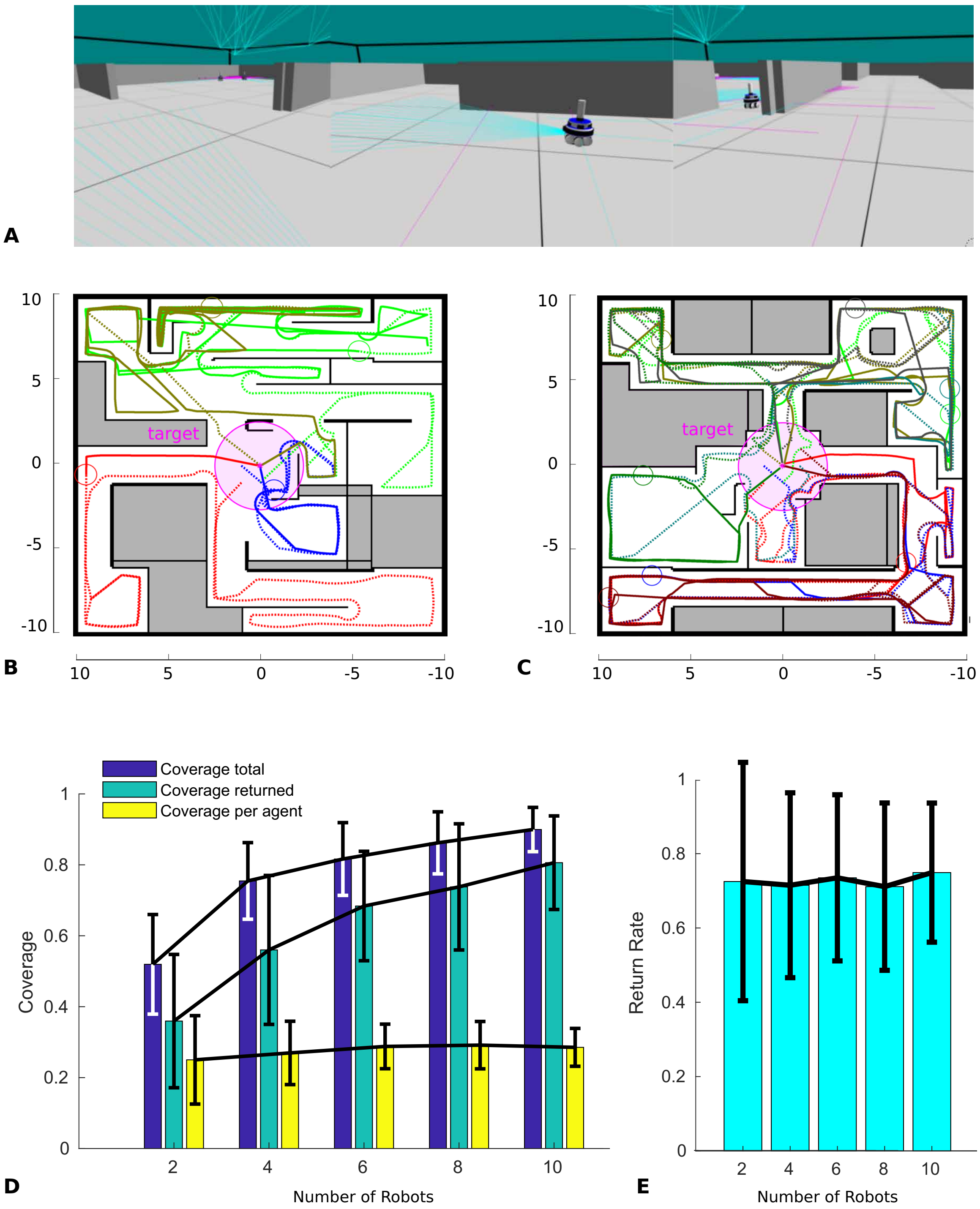


Fig. 3. Simulation results. The results of the simulation environments with (**A**) a representation of the ARGoS simulator and the modified simulated foot-bot. Two example environments and trajectories are shown for (**B**) six robots and (**C**) for four robots. (**D** and **E**) The results of 2, 4, 6, 8, or 10 robots in 100 procedurally generated environments for each configuration, in the coverage (not including nonaccessible areas), and the return rate. Three types of coverage are shown in (D): coverage total (area covered by all robots), coverage returned (area covered only by the robots that have returned), and coverage per robot (area that a single robot has covered). The exact computation of the covered area can be found in text S4.1. Last, in (E), the return rate is shown, i.e., the portion of robots that successfully returns to the base station after exploration. Both bar graphs of (D) and (E) show the mean as the SD, of which the specific values can be found in text S4.1.

simulation. Particularly, we implemented SGBA on the small commercial off-the-shelf (COTS) Crazyflie 2.0 drone developed by Bitcraze AB (*43*). The hardware package of the drones consisted of the following modules. The multi-ranger deck (*44*) was used for obstacle detection and wall following. It has four tiny laser rangers that point to the front, left, right, and back. The flow deck (*45*) was used for coarse visual odometry. It consists of a downward looking camera that determines translational optical flow and a downward pointing laser ranger that scales the flow to obtain height and translational velocity. Bitcraze's own communication hardware, "Crazyradio," with the 2.4-GHz Wi-Fi band (*46*) was used for ranging to other drones and to the wireless beacon. The firmware running on the Crazyflies can be found in (*40*). It also served as a communication channel between the drones for exchanging desired headings. These three light-weight and low-power hardware modules were sufficient for our navigation solution. We performed the experiments in an empty hallway of the faculty of Aerospace Engineering at Delft University of Technology because it allowed us to perform extensive testing (Fig. 4A; see text S1 for a more detailed description). We conducted real-world tests with two, four, and six Crazyflies at the same time. For each number of drones, five different flights were performed.

As in the simulation experiments, the robots started in the middle of the environment. From here, they flew with SGBA to their own preferred outbound flight direction for about one third of their battery life, which was about 2 min. Afterward, they needed to return again using the RSSI of the home beacon. Along the entire path, they avoided each other by using the RSSI of the inter-robot connection, which was handled by the communication scheme presented in Materials and Methods. Because we did not have access to ground-truth global coordinates, we determined the coverage performance in terms of the number of visited rooms, excluding the hallway.

SGBA also allowed tiny robots in the real world to explore the environment, with six tiny drones, on average, flying into 83% of the open rooms in the 40 m by 12 m environment within 7.5 min (Fig. 4D, dark blue bar). Figure 4 (B and C) shows two example trajectories with four and six Crazyflies, respectively. The trajectories show that when a drone entered a room, it typically flew along its complete boundaries. Hence, upon entry, we considered the room "covered." Note that the rather accurate trajectories in Fig. 4 (B and C) have only been reconstructed for visualization purposes and did not play any role in the navigation. The trajectories were plotted on the basis of the coarse onboard odometry, adjusted with the video footage of the external cameras on the scene using post-processing (text S3 explains the procedure and shows the difference between the original and adjusted odometry). The trajectories show how, generally, the drones explored different parts of the environment, thanks to the different preferential directions. In Fig. 4C, an example can be seen where drone 5 lost connection with the beacon in the far top-left room of the environment, so the external camera had to provide the additional trajectory information. Losing the connection was no problem for autonomous navigation because the FSM ran on board the Crazyflie. The visual odometry and wall-following behaviors allowed the drone to escape the room and to reconnect with the home beacon. Video compilations of the flight from Fig. 4C can be found in movie S1.

Figure 4D shows that, as in simulation, a clear trend can be seen that the total coverage increases with the number of drones. However, the increase of the coverage by returned drones seems less steep than in simulation. The reason for this is that both the coverage per robot and the return rate (Fig. 4E) slightly decreased when adding more Crazyflies. This is due to many issues, such as hardware malfunctions, sensing failures, and (even for six drones) a collision between two drones (see the pie chart of Fig. 4E). Having a limited battery capacity is a real-world problem as well but was taken into account in the simulation in the form of a time limit for the results in Fig. 3. Concerning the collision avoidance, in all 15 real-world experiments, there were 54 encounters between drones and only one collision. This corresponds to a 98% success rate of the implemented avoidance maneuver. The logged data can be found in (*42*), and statistical tests of the real-world results can be found together with a table with the numbers of encounters and collisions in text S4.2. In addition, the real-world results (Fig. 4D) show that the trend of the total coverage is significantly related to the total number of drones.

Proof-of-concept search-and-rescue mission

Last, to illustrate the potential application of SGBA, we applied it to a search-and-rescue exploration mission. The light-weight Crazyflies carried a mission-relevant payload: A forward-looking camera and an SD card (see Materials and Methods for the exact setup), which resulted in a flight time of 5.5 min in total. This extra camera allowed storage of images captured during flight for inspection by a human. Although the proposed navigation solution worked with COTS Crazyflie modules, for the proof-of-concept search-and-rescue mission, we had to make a custom, lighter weight, and lower power laser-ranging module to accommodate the extra camera and SD card. This custom ranging module replaced the Crazyflie's multi-ranger module.

The experiment simulated a search-and-rescue scenario, in which two human-size wooden figures were placed in two different rooms in the hallway. The same starting position was used for the four-camera–equipped Crazyflies as in the previous test. To cope with the limited flight time of the prototypes, we chose to perform only the outbound flight. The trajectories in Fig. 5C were inferred from the onboard camera footage in combination with the external cameras.

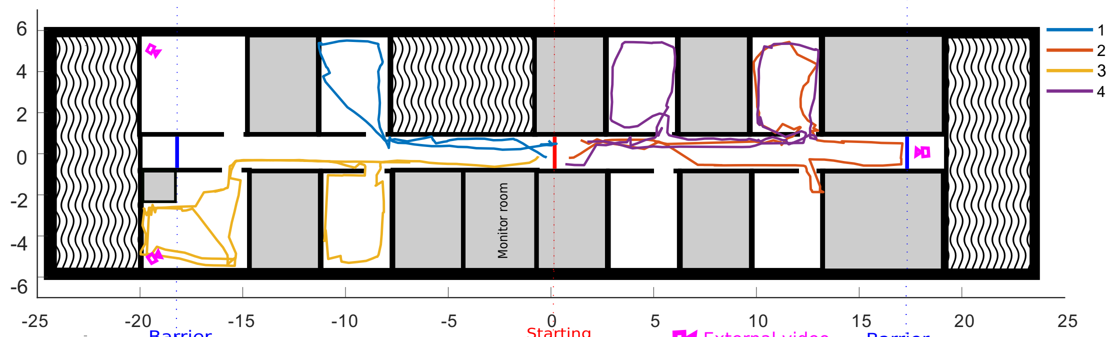
Both victims were found by the drones. In Fig. 5 (A and B), we can see that Crazyflies 1 and 4 were flying in the rooms where the victims were located. Drone 4 was able to capture the victim on its onboard camera (Fig. 5E). However, Crazyflie 1 stopped recording right before it flew into the room with the victim. Luckily, the victim was spotted by drone 3 from another angle (Fig. 5D). This example shows the advantage of using swarming, which can yield redundant observations in an exploration task. A video compilation can be found in movie S2.

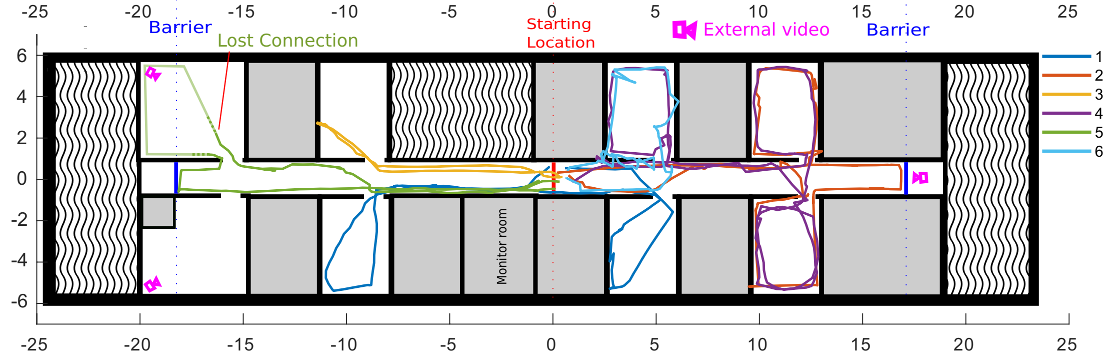
As explained, the drones did not make a map of their environment for navigation. After the drones came back, their collected images could be downloaded to the base station. A human end user can then go through the images, and, when finding a victim, look at the video of the robot in fast-forward to find the location. If a map is desired by the end user, the base station computer could also generate a map based on the onboard images with state-of-the-art SLAM methods [e.g., (*16*)]. This map generation may be challenging due to real-world factors such as quick motions, motion blur from vibrations, lack of texture, etc.

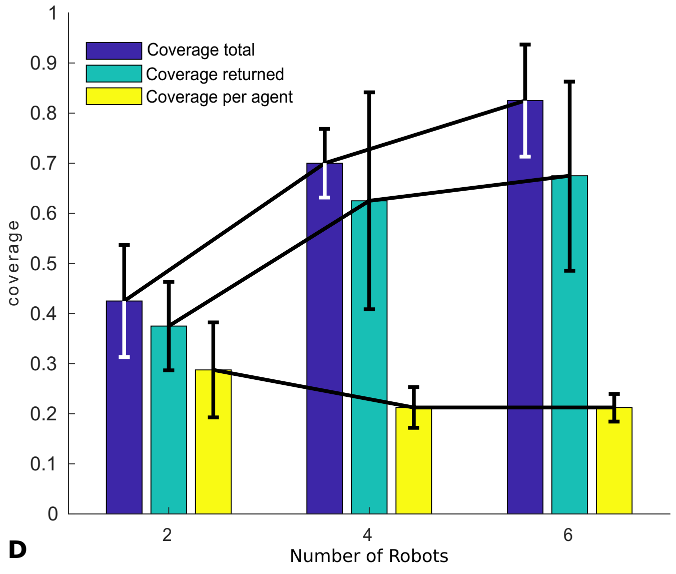
DISCUSSION

The purpose of this work was to develop an alternative for navigating a group of tiny, resource-limited flying robots in an unknown, GPS-denied environment. We presented the SGBA, which fit onboard the Crazyflie robot, weighing a mere 33 g. Owing to the STM32F4 microprocessor and added sensing capabilities (multi-ranger and flow-deck expansion decks), it navigated through a real office environment. Moreover, while communicating with its peers, it avoided other Crazyflies and increased coverage in the overall exploration task. The algorithm enabled a group of small and limited flying robots to fully autonomously navigate in a real environment by using their

A

B


C


D


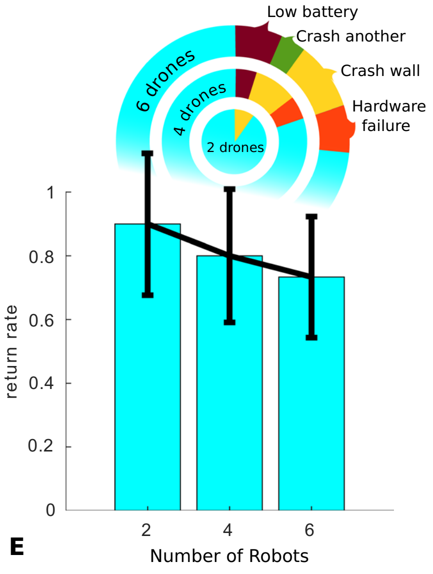
E


Fig. 4. Real-world results. The results of the real-world experiments with (**A**) a representation of the environment used and the Crazyflie 2.0's with the necessary expansion decks. Several example trajectories are shown for (**B**) four robots and (**C**) for six robots from their onboard odometry (adjusted by means of the external cameras). The results of two, four, or six robots for five flights in each configuration are shown in (**D**) for the coverage (not including the nonaccessible areas in gray) and in (**E**) for the return rate (cyan bars), with a pie chart additionally indicating the percentages of real-world–related issues, which prevented a successful return to the base station. Both bar graphs of (D) and (E) show the mean as the SD, of which the specific values can be found in text S4.2.

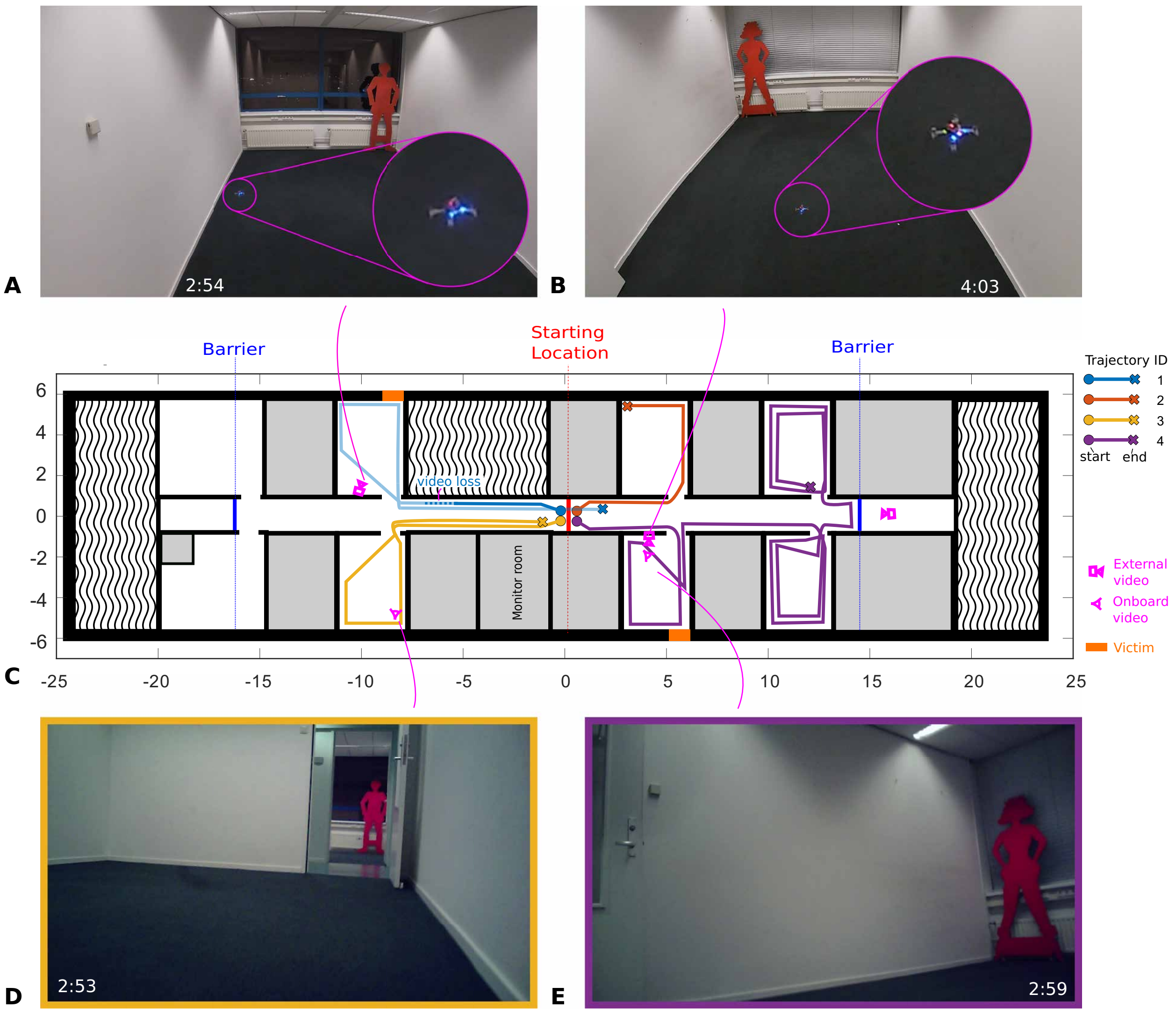


Fig. 5. Proof-of-concept search-and-rescue mission. The results of the experiment in which the Crazyflies carry a camera to detect victims in the environment. (**A** and **B**) Screenshots of the external cameras capturing the Crazyflies during their flight. (**C**) Trajectory of the four Crazyflies (inferred from the onboard and external camera). (**D** and **E**) Screenshots of the onboard Hubsan camera, with the two human-shaped silhouettes captured during the exploration flight.

onboard sensors and processing capabilities. Still, there are several elements to consider for extending this work to bigger and more complex environments as seen, for instance, in real-world search-and-rescue scenarios.

There are several options that would improve navigation performance. First, we expect that using Ultra Wide Band (UWB) instead of the Crazyradio PA would substantially improve both localization with respect to the beacon and sensing of other drones [see e.g., (*47*)]. The Crazyradio is heavily influenced by other 2.4-GHz sources (Wi-Fi), and the inter-Crazyflie chatter and the RSSI-based distance measurements are particularly noisy and heading dependent. The good overall results show the robustness of SGBA. Still, communication could be considerably improved by using UWB communication with time-of-flight ranging. For instance, a single DecaWave DWM1000 module can provide ranging to another such module for a distance up to 290 m, through walls and obstacles, with ~10-cm accuracy (*48*). Because SGBA only needs ranging to a single beacon, it can use the full extent of the UWB range to notably improve the inbound flight.

In addition, the collision avoidance between drones may benefit from using UWB. The experimental results showed an increase in coverage when adding more drones to the swarm. However, the increase follows a law of diminishing returns. We expect this phenomenon to be fundamental because having more robots necessarily means covering more of the same area when they start from the same point, and it also means that robots will spend more time avoiding each other. Still, in our current implementation, using too many drones also led to communication overload during the flight. In our current communication scheme (see Materials and Methods), the more drones there were, the slower they communicated with each other. Increasing the number of drones may cause problems of miscoordination or, even worse, a higher likelihood of interdrone crashes. We saw in the real-world results (Fig. 4E) that the latter did not occur with two and four drones, but it did once with six drones.

The optimal number of drones will depend on the size of the environment and, with the current implementation of interswarm avoidance, the communication hardware and protocol. Because of UWB's greater robustness with respect to interference and its higher throughput, we expect it to also improve the scalability of the current proposed scheme for drone collision avoidance.

During the real-world experiments, there was almost always a connection between the home beacon and all drones. In Fig. 4C, a disconnected drone kept executing its tasks autonomously because the FSM runs fully on board. It was therefore able to get out of a communication dead zone eventually. However, the question still arises: How will the robots be able to get home if the beacon is lost completely? Even with the earlier mentioned UWB improvement, there are situations where the environment is larger than the range of the beacon. A useful addition to cope with the home beacon loss problem in bigger environments is to make more use of the swarm. As the Crazyflies are communicating with each other, they can also be used as a beacon themselves. As soon as a drone loses connection with the homing beacon, it could try to find another Crazyflie that is still connected to the beacon and navigate toward that position first, reconnect with the original home beacon, and resume its navigation to the starting position [a strategy that reminds of "chains" of robots as used in, e.g., (*49*), but that would be more economical in terms of the number of used robots]. Yet, this requires that at least several Crazyflies always need to stay connected to the home beacon and therefore are limited in their own missions.

Improving the robots' sensing capabilities would also improve the results. Specifically, the multi-ranger deck proved to be sufficient for our test environment, yet there are limitations. For instance, it cannot see very thin objects and relies on the flow deck to work properly. Although the flow deck and the existing sensor fusion provided stable velocity-driven flight, the dark floor in the office environment turned out to be challenging. Therefore, a Crazyflie would occasionally drift and move into a direction where obstacles were present in the blind spots of the multi-ranger. A higher robustness to collisions from a protective cage, as proposed in (*50*), would help. The wall following and obstacle detection can also be made more robust. A possible solution is to add a light-weight vision system, as in (*51*, *52*). Vision can provide distance estimates in an entire field of view and aid the velocity estimation and odometry by means of frontal optical flow [see (*53*)]. This would reduce problems with textureless floors. Even so, a fundamental limitation of using the multi-rangers, cameras, or optical flow sensors is that they will be ineffective if the surroundings are filled with smoke. In that case, different sensors such as sonar or radar can be used, whereas the navigation can remain identical.

The high efficiency of SGBA in terms of sensing, computation, and memory comes at the cost of navigation efficiency. Not building a global map and not performing computationally expensive optimal path planning results in suboptimal paths. Drones can revisit rooms multiple times or can visit rooms that were already visited by other drones. This could perhaps be solved in a relatively efficient manner, e.g., involving visual landmark recognition. Still, the experimental results have shown that multiple measurements from the same area can be beneficial. Camera footage can get temporarily occluded or even lost, as happened with drone 1 in Fig. 5C. Moreover, the fact that drones' views overlap with each other can make a substantial difference in the data collection if not all robots are able to return.

We illustrated the potential of SGBA by implementing it on the smallest possible commercially available quadrotor. However, the discussion above suggests that the method would perhaps be even more successful on a custom-designed drone. One option is to implement SGBA on a smaller platform while keeping a similar performance. Making SGBA work on a smaller drone is possible because a custom design would not have to be as modular and easy to use as the Crazyflie decks. That this is possible is already shown by the lighter and more energy-efficient custom laser ranger deck that was made for the proof-of-concept search-and-rescue mission. Another option is to implement SGBA on a slightly bigger drone for better performance. We expect that using a slightly bigger drone with better sensing, communication devices, and more battery capacity would notably improve the return rate of the drones because it would reduce collisions both with obstacles and with other drones, make the inbound flight more efficient, and extend the flight time available for returning. Even if such a drone could have a bit more processing available, the current proposed navigation solution remains of high interest because it will leave much room for other types of functionalities. This may be used by vision algorithms to enhance the navigation or by other algorithms performing mission-specific tasks.

In the future, more processing power will become available to small robots [see, e.g., (*54*)]. In comparison with 3D SLAM, SGBA will always be available to smaller robots. For instance, it is not unthinkable that SGBA may be applied to the 80-mg RoboBee (*55*). Furthermore, small robots will have to use their onboard computing power for all tasks that they need to perform autonomously. It is essential for small robots to have computationally efficient algorithms for all tasks they perform. Using SGBA implies that there is more computational power and memory available for other mission-relevant tasks. Hence, we expect SGBA to remain relevant even with the further progress in the miniaturization of computing devices.

Last, we suggest application scenarios for which the developed swarm exploration seems suitable. We performed a preliminary investigation into SGBA's use for search and rescue. However, the proposed method is also suitable for other tasks because it allows a swarm of small robots to quickly explore a potentially unknown environment. Hence, we believe that it is also suitable for exploring an unknown cave or inspecting the inside of a building that is about to collapse. Apart from unknown environments, SGBA can also be applied to known environments, for instance, in surveillance applications. In the case of surveillance for security, one may typically think of robots performing regular trajectories, e.g., along the property's perimeter, but a less predictable swarm-based surveillance may be better in countering unwanted intruders. In many applications, robots will need to operate for longer times. For example, in an inventory-tracking scenario, a swarm of safe, tiny drones may buzz around the warehouse, continuously flying out to scan products and then returning to base to recharge. Similar setups may also serve blue algae monitoring by little robot boats or floor cleaning by small garbage collection robots. A swarm of tiny robots has benefits because the robots are safe, cheap, navigate in narrow spaces, and, as a group, can quickly cover relatively large areas.

CONCLUSION

To conclude, we presented a minimal navigation solution, the SGBA, that allows tiny flying robots to successfully explore a real-world environment. In our experiments, the Crazyflie robots only used their inertial measurement unit, four tiny 1D laser range finders, an optical flow deck, and a very light 2.4-GHz radio chip. The processing fit easily in the single 32-bit, 168-MHz, 196-kB RAM microcontroller of the Crazyflie in addition to all flight control code. Instead of building a map of the environment like conventional SLAM techniques, our navigation solution consists of a combination of simple behaviors and behavioral transitions to accomplish the complex tasks of autonomous exploration and homing. The guiding principle here is to trade off properties such as path optimality and accuracy with resource efficiency, allowing for autonomous swarm navigation. We believe that this principle can offer inspiration for solving other complex robotic tasks as well with swarms of cheap and safe tiny robots.

MATERIALS AND METHODS

Here, we explain the exploration and homing strategy of SGBA, starting with the navigation of a single robot and then expanding to larger numbers of robots. Afterward, we explain the hardware used for the real-world experiments.

Outbound travel

We start our explanation of the FSM with the outbound travel of a single robot. Figure 6 (A and B) illustrates the entire FSM, where the robot starts at "Init." For the outbound travel, it is important to realize that the robot just needs to explore the available space and does not need to go to a specific location. Therefore, it will only be assigned a preferred heading. After it encounters, follows, and then leaves an obstacle, it will follow that same heading again (Fig. 6C). Of course, there will be heading drift over time. In the case of the Crazyflie robots used in the real-world experiments, the drift was ~0.10°/s (48° over the 8-min flight time). Still, because the main goal of the heading estimate was to send multiple robots into roughly different directions, the drift did not significantly affect SGBA's performance.

After the robot detects an obstacle with its front laser range sensor, it will start the wall-following behavior. First, it chooses an initial "local direction," which decides whether to follow the wall on the right- or left-hand side. We chose a local direction policy based on the strategies of DistBug (*56*) and FuzzyBug (*57*), namely, by adopting the "angle of attack" as the robot approaches the wall. With the current hardware of the four laser range sensors in all four directions of the horizontal plane, the robot could easily determine the angle of the wall by evaluating whether the side range sensors are triggered in combination with the front one. The main assumption here is that the wall needs to be straight. However, if this is not the case, this does not mean that the strategy will fail. If the local direction ends up being a less optimal one, this will be corrected for at a later time. From here on, the robot starts following the boundary of the obstacle and the wall.

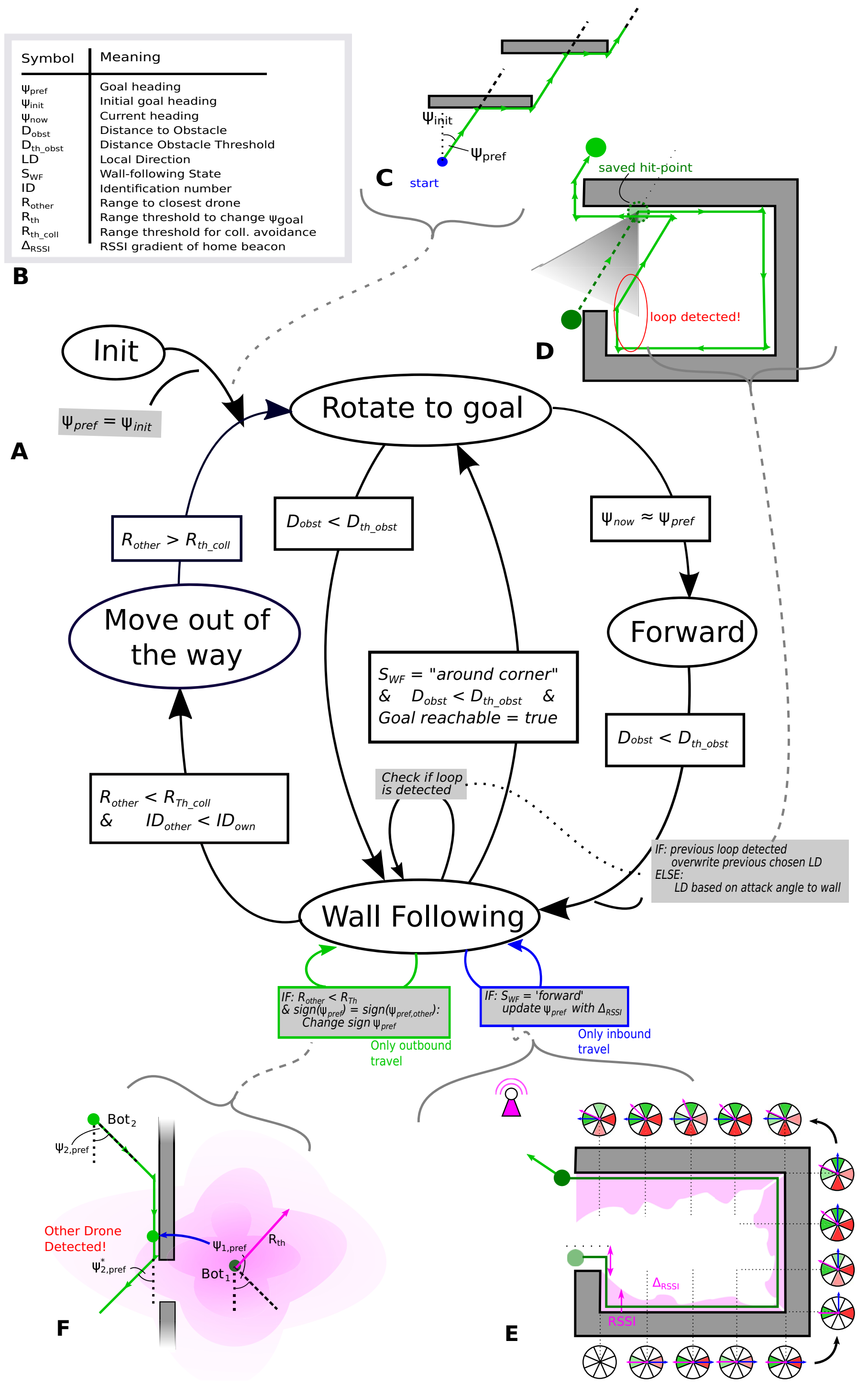
SGBA uses memory for loop detection. Memoryless bug algorithms are prone to getting stuck in loops because they may encounter an obstacle, perform wall following, and then leave the obstacle in a direction, which will lead them back to exactly the same obstacle. This will lead to an endless loop, devastating the navigation performance. An example of this can happen in a room, where a robot's preferred direction is away from the only door in the room. It may then enter the room and travel through the room until it detects the wall on the opposite side of the room. Subsequently, it will follow the walls of the room until it is following the wall with the door entry. However, because its preferred direction is away from the door, it may leave that wall again before reaching the door, travelling to the opposite wall again. The occurrence of this type of loop is why during wall following SGBA keeps track of its position relative to the location where the robot first detected the obstacle (termed the "hit point"). If the robot tracks back, due to the environment characteristics, and crosses the area behind the hit point, it will detect that as a loop. This means that once the robot leaves the obstacle and encounters another, which is usually the same hit point as last time, it will not base its local direction on the current wall angle but on the reverse of the direction chosen at the previous saved hit point. This position tracking is illustrated in Fig. 6D and is done completely with relative position estimations of the onboard odometry. Because this procedure is only used for local decision-making within small rooms, this was a sufficient tactic to handle loops within our experiment environment. However, this probably will not prevent a potential loop in large areas because the drift will be too severe. We have studied the effect that SGBA's loop detection has on the return rate (text S6). The results show that for one robot, the return rate dropped substantially when there was no loop detection, but for six robots, the effect was less evident. Upon detailed inspection, we noticed that inter-robot encounters are responsible for getting stuck robots out of a loop. With this, they can cope with the lack of a proper loop detection, which is an interesting feature of the swarming element of SGBA.

Wireless communication-based inbound travel

After a few minutes, either after a time threshold has passed or based on the remaining voltage of the battery, the robot needs to return to its base station. This is extremely important for robots that store their measurements onboard and do not stream their results to the operator. To achieve this, SGBA keeps track of the gradient of the filtered RSSI (see text S2 for raw measurements) while it is performing the wall following, as seen in Fig. 6E. During the straight parts of the procedure, it has a circular buffer, corresponding to the heading of the robot, where the values in the buffer track the directions in which the RSSI has increased over time. In the direction of the RSSI increase, the buffer value is incremented, whereas the value in the opposite (-180°) direction is decremented to give it a lower influence. For an RSSI decrease, the exact opposite procedure is done, and for no RSSI change, the buffer values stay the same. Both incrementation and decrementation, based on the RSSI's derivative, are done for every N meter, where N is a decimal number defined by the user. Every $N \times k$ meter, where k is a scalar value, a vanishing function is applied to decrease the influence of older RSSI measurements. This RSSI change in function of the heading allows the robots to estimate the direction to the home beacon, which they will use for the return travel any time they are not forced to follow an obstacle or wall. Because the RSSI increase is noisy and irregular, this will usually not be an exact angle but a coarse indication of where the beacon is. This proved to be enough for the robot to return to its home base. Any drift in the robot's heading estimate is not problematic for the inbound travel because the direction to the home beacon is determined with respect to this internal heading representation.

Coordination among drones

A single robot could use the SGBA-FSM by itself to navigate. However, it only has a limited battery capacity and therefore will not be able to explore the entire environment. For this reason, it is more advantageous to use a swarm of robots. However, using multiple



Symbol	Meaning
ψ_{pref}	Goal heading
ψ_{init}	Initial goal heading
ψ_{now}	Current heading
D_{obst}	Distance to Obstacle
D_{th_obst}	Distance Obstacle Threshold
LD	Local Direction
S_{WF}	Wall-following State
ID	Identification number
R_{other}	Range to closest drone
R_{th}	Range threshold to change ψ_{goal}
R_{th_coll}	Range threshold for coll. avoidance
Δ_{RSSI}	RSSI gradient of home beacon

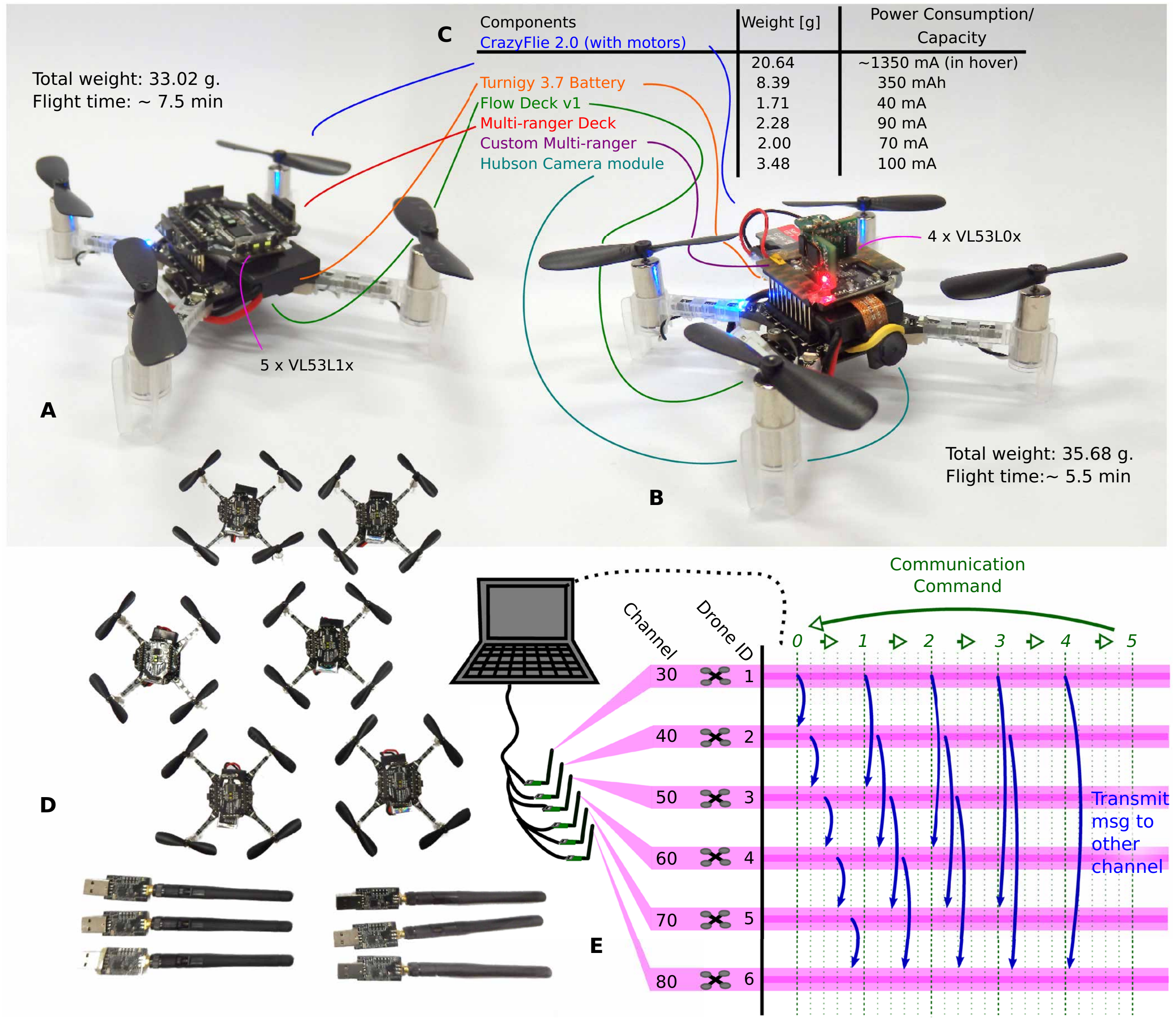
B
C
start
ψinit
ψpref
D
saved hit-point
loop detected!
A
Init
ψpref = ψinit
Rotate to goal
Dobst < Dth_obst
ψnow ≈ ψpref
Rother > Rth_coll
Move out of the way
Forward
SWF = "around corner" & Dobst < Dth_obst & Goal reachable = true
Dobst < Dth_obst
Rother < RTh_coll & IDother < IDown
Check if loop is detected
IF: previous loop detected overwrite previous chosen LD ELSE: LD based on attack angle to wall
Wall Following
IF: Rother < RTh & sign(ψpref) = sign(ψpref,other): Change sign ψpref
Only outbound travel
IF: SWF = 'forward' update ψpref with ΔRSSI
Only inbound travel
Bot2
ψ2,pref
Other Drone Detected!
ψ1,pref
Rth
ψ*2,pref
Bot1
F
ΔRSSI
RSSI
E

Fig. 6. SGBA FSM. (**A**) FSM of the SGBA with (**B**) a legend of symbols. Its individual subsystems are illustrated as follows: (**C**) Outbound navigation in which the robot attempts to follow its goal heading, following obstacle contours on the way, and (**D**) local direction preferences based on angle of attack, i.e., the angle that the robot's trajectory makes with the wall and the principle of the loop detection. The addition to the state machine of the gradient search toward the beacon at the home location for the inbound travel is given in blue, with (**E**) the gradient search method during the straight parts of the wall following. Here, the robot tries to estimate the direction toward the beacon by integrating information on the received signal strength based on its heading along the way. Swarm coordination addition to the state machine for the outbound flight (green), where (**F**) shows that the robot will change its goal heading if another drone (with higher priority) has its preferred heading in the same direction. In case the drones are even closer, the one with the lowest priority will move out of the way completely for both inbound and outbound travel.

robots poses a new set of problems. First, the robots need to avoid each other, and second, they should coordinate the search with each other to achieve maximum dispersion and avoid conflicts. This was done with their communicated information and range measurements (range or RSSI) and implemented by the "move out of way" state in Fig. 6A. For collision avoidance, if two robots come really close to each other ($R_{other} < R_{th\_coll}$), the robot with the high-priority (and in our case lower ID, ID_{low}) will have the right of way. The low-priority robot (ID_{high}) will perform an action enabling ID_{low} to smoothly move past it. After staying out of ID_{low}'s way until $R_{other} > R_{th\_coll}$, ID_{high} resumes navigation. For the coordination of the search, the robots dynamically adapt their preferred heading. Initially, each robot is assigned a preferred heading, chosen out of *K* different directions. If during outbound travel a robot comes nearby another one ($R_{other} < R_{th}$) and these robots have a similar preferred heading, then the low-priority robot ID_{high} will change the sign of its preferred heading and carry on (Fig. 6F). The next time it leaves the obstacle, it will therefore move away from the search area of the robot with the higher priority, ID_{low}. A change in preferred heading is triggered earlier than a collision avoidance action ($R_{th\_coll} < R_{th}$), and it is only performed during outbound travel. During inbound travel, the preferred heading is going toward to the home beacon. Hence, during inbound travel, the range between robots is only used for collision avoidance. The different implementations for simulated and real-world robotic platforms can be found in text S5.

Hardware

For the experiments, we used Bitcraze's platform Crazyflie 2.0 (*43*), augmented with the flow deck v2.0 (*45*) and the multiranger (*44*) expansion decks, which can be seen in Fig. 7A. An alternative battery with more capacity was added for a longer flight time, namely, the Turnigy nanotech 300 mAh (1S 45-90C) LiPo battery, providing an average power supply of 3.7 V. To make sure the entire path in front of the drone is free of obstacles using the 20° field-of-view laser ranger, the minimum required detection range is 50 cm, for which the VL53L1xs (*58*) laser ranger on the multi-ranger is sufficient.



Components	Weight [g]	Power Consumption/ Capacity
CrazyFlie 2.0 (with motors)	20.64	~1350 mA (in hover)
Turnigy 3.7 Battery	8.39	350 mAh
Flow Deck v1	1.71	40 mA
Multi-ranger Deck	2.28	90 mA
Custom Multi-ranger	2.00	70 mA
Hubson Camera module	3.48	100 mA

Fig. 7. Hardware and communication specifics. (**A**) Crazyflie used for outbound and inbound travel and (**B**) the assembly used for the video recording of the environment. (**C**) Components on the Crazyflie, including weight and battery consumption. (**D**) Total of six Crazyflies used including six Crazyradio PAs and (**E**) the communication scheme shown for the six-drone experiment. Here, a counter is regulating when the drone will transmit a message (msg) to another drone (for counter 1: drone 1 to 2, drone 2 to 3, etc.). Between the regulated counter, the drone transmits its message to another drone with a time offset based on its ID. Six PAs were used for the six communication channels to receive logging of the Crazyflies for statistics; however, these can be replaced by one if no telemetry is required.

The flow deck contains a PMW3901MB optical flow sensor (*59*) to detect motion, with an additional VL53L1x for height detection and control. Within the existing state estimation (*60*), the Crazyflie achieved excellent hover and velocity control, and optical flow was detected on most surfaces. Nevertheless, dark colors should be avoided. The dark low-texture floors in our real-world environment were challenging (Fig. 4A), and the flyable height where motion detection was still reliable was only 0.5 m.

For the onboard video recording experiments, we designed a custom expansion board, which included configuration of the lower power VL53L0x time-of-flight (*61*) sensors (the predecessor of the VL53L1xs on Bitcraze's multi-ranger deck) and a camera module, meant as a spare part of the Hubsan X4 H107C RC Quadcopter (*62*). This camera module carries an SD card to record the videos captured during the SGBA navigation of the Crazyflies. This configuration is displayed in Fig. 7B. The weight of the platforms and the average power consumption per expansion board are shown in Fig. 7C, which resulted in approximate flight times of 7.5 min for the left-hand Crazyflie configuration and 5.5 min for the right-hand Crazyflie configuration in Fig. 7.

To fully execute the SGBA, a communication protocol (Fig. 7E) has been flashed into the NRF51 microprocessor, which handles the Crazyradio communication (2.4-GHz Wi-Fi protocol and Bluetooth), and the power distribution. Each drone has its own unique ID number; We consider numbers one to six in this explanation. This ID also indicates in which channel (ID × 10 + 20) the Crazyflie communicates with the computer for logging the onboard variables, as can be seen in Fig. 7E. This separation of channels was done to reduce interference between the Crazyflies. The variable logging of each Crazyflie to the computer was done at 0.5 Hz, which includes the position estimation, the RSSI of the beacon and other robots, the SGBA status of the state machine, etc. In our experiments, because we needed to receive the onboard data of each Crazyflie separately for the statistics in this paper, each drone had its own individual Crazyradio PA (Fig. 7D). This reduced the possibility of package loss of the telemetry data; however, technically, only one beacon is necessary. If the Crazyradio PA quickly switches and transmits empty packages on all the available channels, the SGBA does not require any additional knowledge except for the RSSI value.

The communication between the Crazyflies was done with a counter to prevent package loss due to message collisions (Fig. 7E). The counter regulates when one drone will send a package to another drone, which will be incremented every 0.5 s. For this, it switches to the primary transmitter (PT) mode, changes its communication channel to the other drone's channel, sends the message within a short timeframe, and switches back immediately to its own channel in primary receiver mode to receive messages from the other Crazyflies and to receive an RSSI of the home beacon. Between the regulating counter increments, the moment to switch to PT depends on the drone's ID. This should prevent the Crazyflies from simultaneously sending messages, therefore reducing the possibility of interdrone package loss. The information that the Crazyflies send to each other is their ID and preferred heading. This is necessary for changing direction on the outbound travel. At the same time, the receiving drone also knows the signal strength and hereby has an indication of the proximity of the other robot. Not all Crazyflies send a similar number of messages. The highest priority robot (lowest ID = highest priority) transmits to every channel because all others would need to avoid it, and the lowest priority robot does not send a message at all because it needs to avoid everybody else.

In the experiments, the earlier-mentioned counter was regulated by the computer; however, each Crazyflie would be able to do this by itself after clock synchronization of the autopilots. The separation of channels on the Crazyradio modules was necessary to enable stable communication between Crazyflies. It should be possible to put multiple Crazyflies on one channel; however, the number of usable channels and the number of robots per channel are limited. This poses a restriction for the total number of robots that the 2.4-GHz Crazyradio protocol is useful for; however, this could be further scaled by using UWB instead and a more sophisticated scheduling protocol such as self-organized time-division multiple access (STDMA) as in (*63*).

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/4/35/eaaw9710/DC1

REFERENCES AND NOTES

1. D. Floreano, R. J. Wood, Science, technology and the future of small autonomous drones. *Nature* **521**, 460–466 (2015).
2. K. Y. Ma, P. Chirarattananon, S. B. Fuller, R. J. Wood, Controlled flight of a biologically inspired, insect-scale robot. *Science* **340**, 603–607 (2013).
3. C. R. Reid, D. J. T. Sumpter, M. Beekman, Optimisation in a natural system: Argentine ants solve the Towers of Hanoi. *J. Exp. Biol.* **214**, 50–58 (2011).
4. R. Menzel, J. Fuchs, A. Kirbach, K. Lehmann, U. Greggers, Navigation and communication in honey bees, in *Honeybee Neurobiology and Behavior* (Springer, 2012), pp. 103–116.
5. F. Mondada, L. M. Gambardella, D. Floreano, S. Nolfi, J. Deneubourg, M. Dorigo, The cooperation of swarm-bots: Physical interactions in collective robotics. *IEEE Robot. Autom. Mag.* **12**, 21–28 (2005).
6. K. H. Petersen, R. Nagpal, J. K. Werfel, *Termes: An autonomous robotic system for three-dimensional collective construction* (Robotics: Science and Systems VII, 2011).
7. M. Rubenstein, A. Cornejo, R. Nagpal, Robotics. Programmable self-assembly in a thousand-robot swarm. *Science* **345**, 795–799 (2014).
8. A. Kushleyev, D. Mellinger, C. Powers, V. Kumar, Towards a swarm of agile micro quadrotors. *Auton. Robots* **35**, 287–300 (2013).
9. M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, A. L. Christensen, Evolution of collective behaviors for a real swarm of aquatic surface robots. *PLOS ONE* **11**, e0151834 (2016).
10. Intel, Intel Drone Light Show Breaks Guinness World Records Title At Olympic Winter Games Pyeongchang 2018 (2018); https://newsroom.intel.com/news-releases/intel-drone-light-show-breaks-guinness-world-records-title-olympic-winter-games-pyeongchang-2018 .
11. G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, T. Vicsek, Outdoor flocking and formation flight with autonomous aerial robots, paper presented at the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Chicago, IL, 14 to 18 September 2014.
12. S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, D. Floreano, Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate, paper presented at the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, 25 to 30 September 2011.
13. G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, T. Vicsek, Optimized flocking of autonomous drones in confined environments. *Sci. Robot.* **3**, eaat3536 (2018).
14. J. Fuentes-Pacheco, J. Ruiz-Ascencio, J. M. Rendón-Mancha, Visual simultaneous localization and mapping: A survey. *Artificial Intelligence Review* **43**, 55–81 (2012).
15. M. T. Lazaro, L. M. Paz, P. Pinies, J. A. Castellanos, G. Grisetti, Multi-robot SLAM using condensed measurements, paper presented at the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3 to 7 November 2013.
16. J. Engel, T. Schöps, D. Cremers, LSD-SLAM: Large-scale direct monocular SLAM, paper presented at the 2014 European Conference on Computer Vision (ECCV) Zurich, Switzerland, 6 to 12 September 2014.
17. E. López, S. García, R. Barea, L. Bergasa, E. Molinos, R. Arroyo, E. Romera, S. Pardo, A multi-sensorial simultaneous localization and mapping (SLAM) system for low-cost micro aerial vehicles in GPS-denied environments. *Sensors* **17**, E802 (2017).
18. C. Forster, S. Lynen, L. Kneip, D. Scaramuzza, Collaborative monocular slam with multiple micro aerial vehicles, paper presented at the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3 to 7 November 2013.
19. A. Weinstein, A. Cho, G. Loianno, V. Kumar, Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors. *IEEE Robot. Autom. Lett.* **3**, 1801–1807 (2018).

20. S. Jung, S. Hwang, H. Shin, D. H. Shim, Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robot. Autom. Lett.* **3**, 2539–2544 (2018).
21. N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermüller, Y. Aloimonos, Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robot. Autom. Lett.* **3**, 2799–2806 (2018).
22. M. V. Srinivasan, Honeybees as a model for the study of visually guided flight, navigation, and biologically inspired robotics. *Physiol. Rev.* **91**, 413–460 (2011).
23. J. Dupeyroux, J. R. Serres, S. Viollet, AntBot: A six-legged walking robot able to home like desert ants in outdoor environments. *Science Robotics* **4**, eaau0307 (2019).
24. M. Srinivasan, S. Zhang, N. Bidwell, Visually mediated odometry in honeybees. *J. Exp. Biol.* **200**, 2513–2522 (1997).
25. B. A. Cartwright, T. S. Collett, Landmark learning in bees-experiments and models. *J. Comp. Physiol.* **151**, 521–543 (1983).
26. A. Denuelle, M. V. Srinivasan, A sparse snapshot-based navigation strategy for UAS guidance in natural environments, paper presented at the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16 to 21 May 2016.
27. E. Garcia-Fidalgo, A. Ortiz, Vision-based topological mapping and localization methods: A survey. *Robot. Autonom. Syst.* **64**, 1–20 (2015).
28. V. Lumelsky, A. Stepanov, Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Trans. Autom. Control* **31**, 1058–1063 (1986).
29. I. Kamon, E. Rivlin, Sensory-based motion planning with global proofs. *IEEE Trans. Robot. Autom.* **13**, 814–822 (1997).
30. A. Sankaranarayanan, M. Vidyasagar, A new path planning algorithm for moving a point object amidst unknown obstacles in a plane, paper presented at the IEEE International Conference on Robotics and Automation (ICRA), Cincinnati, OH, 13 to 18 May 1990.
31. F. Mastrogiovanni, A. Sgorbissa, R. Zaccaria, Robust navigation in an unknown environment with minimal sensing and representation. *IEEE Trans. Syst. Man Cybern. Part B* **39**, 212–229 (2009).
32. Q. M. Nguyen, L. N. M. Tran, T. C. Phung, A study on building optimal path planning algorithms for mobile robot, paper presented at the 2018 4th International Conference on Green Technology and Sustainable Development (GTSD), Ho Chi Minh City, Vietnam, 23 to 24 November 2018.
33. K. McGuire, G. de Croon, K. Tuyls, A comparative study of bug algorithms for robot navigation. *Robot. Autonom. Syst.* **121**, 103261 (2019).
34. K. Taylor, S. M. LaValle, I-Bug: An intensity-based bug algorithm, paper presented at the 2009 IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 12 to 17 May 2009.
35. K. Taylor, S. M. LaValle, Intensity-based navigation with global guarantees. *Auton. Robots* **36**, 349–364 (2014).
36. J. N. Twigg, J. R. Fink, L. Y. Paul, B. M. Sadler, RSS gradient-assisted frontier exploration and radio source localization, paper presented at the 2012 IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, MN, 14 to 18 May 2012.
37. M. Chinnaaiah, K. Anusha, B. Bharat, M. Divya, P. S. Raju, S. Dubey, Deliberation of curvature type Obstacles: A new approach using FPGA based robot, paper presented at the 2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT), Kannur, India, 23 to 24 March 2018.
38. C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, M. Dorigo, ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence* **6**, 271–295 (2012).
39. A. Vardy, ARGos Bridge (2016, 2017); https://github.com/BOTSlab/argos_bridge.
40. K. McGuire, SGBA_code_SR_2019 (2019); https://github.com/tudelft/SGBA_code_SR_2019.
41. M. Bonani, V. Longchamp, S. Magnenat, P. Rétornaz, D. Burnier, G. Roulet, F. Vaussard, H. Bleuler, F. Mondada, The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research, paper presented at the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18 to 22 October 2010.
42. K. McGuire, Dataset for "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment" (DataverseNL, 2019); https://hdl.handle.net/10411/YVP873.
43. Bitcraze AB, Crazyflie 2.0 (2019); www.bitcraze.io/crazyflie-2/.
44. Bitcraze AB, Multi-ranger deck (2019); www.bitcraze.io/multi-ranger-deck/.
45. Bitcraze AB, Flow deck v2 (2019); www.bitcraze.io/flow-deck-v2/.
46. Bitcraze AB, Crazyradio PA (2019); www.bitcraze.io/crazyradio-pa/.
47. S. van der Helm, M. Coppola, K. N. McGuire, G. C. H. E. de Croon, On-board range-based relative localization for micro air vehicles in indoor leader–follower flight. *Auton. Robots* **2019**, 1–27 (2019).
48. Decawave, DWM1000 Productbrief p. 1, V1.2, 2018; www.decawave.com/product/dwm1000-module/.
49. M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen, A. Decugniere, G. Di Caro, F. Ducatelle, E. Ferrante, A. Forster, J. M. Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. M. de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stutzle, V. Trianni, E. Tuci, A. E. Turgut, F. Vaussard, Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robot. Autom. Mag.* **20**, 60–71 (2013).
50. Y. Mulgaonkar, A. Makineni, L. Guerrero-Bonilla, V. Kumar, Robust aerial robot swarms without collision avoidance. *IEEE Robot. Autom. Lett.* **3**, 596–603 (2018).
51. C. De Wagter, S. Tijmons, B. D. Remes, G. C. de Croon, Autonomous flight of a 20-gram flapping wing MAV with a 4-gram onboard stereo vision system, paper presented at the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May to 7 June 2014.
52. D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, L. Benini, A 64mW DNN-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal* **2019**, 1–1 (2019).
53. K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, H. Kappen, Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robot. Autom. Letters* **2**, 1070–1076 (2017).
54. S. Jones, M. Studley, S. Hauert, A. F. T. Winfield, A two teraflop swarm. *Front. Robot. AI* **5**, 11 (2018).
55. N. T. Jafferis, E. F. Helbling, M. Karpelson, R. J. J. N. Wood, Untethered flight of an insect-sized flapping-wing microscale aerial vehicle. 570, 491 (2019).
56. I. Kamon, E. Rimon, E. Rivlin, Range-sensor based navigation in three dimensions, paper presented at the Proceedings 1999 IEEE International Conference on Robotics and Automation (ICRA), Detroit, MI, 10 to 15 May 1999.
57. S. Lee, T. M. Adams, B.-y. Ryoo, A fuzzy navigation system for mobile construction robots. *Automation in construction* **6**, 97–107 (1997).
58. STMicroelectronics, VL53L1x (2019); www.st.com/en/imaging-and-photonics-solutions/vl53l1x.html.
59. P. I. Inc., PMW3901MB-TXQT (2019); www.pixart.com/products-detail/44/PMW3901MB-TXQT.
60. M. W. Mueller, M. Hehn, R. D'Andrea, Covariance correction step for kalman filtering with an attitude. *J. Guid. Control Dynam.* **40**, 2301–2306 (2016).
61. STMicroelectronics, VL53L0X (2019); www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html.
62. Hubsan, HD Camera module 720P (2019); www.hubsan.com/eur/index.php?main_page=product_info&products_id=228.
63. M. Coppola, K. N. McGuire, K. Y. W. Scheper, G. C. H. E. de Croon, On-board communication-based relative localization for collision avoidance in Micro Air Vehicle teams. *Auton. Robots* **42**, 1787–1805 (2018).

Acknowledgments: We thank V. Trianni for helpful comments on the early manuscript. We also thank L. Schrover and J. Siebring for allowing us to use a portion of the building of the TU Delft faculty of Aerospace Engineering for our experiments. We also acknowledge the great help of E. van der Horst for his advice on setting up the communication protocol. Moreover, we are grateful to Bitcraze AB's for help with multiple hardware-related issues with the Crazyflie. **Funding:** This work was financed by the Dutch Research Council (NWO) under grant no. 656.000.006 within the Natural Artificial Intelligence Programme. **Author contributions:** All authors contributed to the conception of the project and were involved in the analysis of the results and revising and editing the manuscript. G.C.H.E.d.C. and K.N.M. have forwarded the idea of a swarm bug algorithm with a beacon for homing. K.N.M. has designed and implemented the SGBA specifics for the experiments with support of C.D.W. and G.C.H.E.d.C. The manuscript was primarily written by K.N.M., C.D.W., G.C.H.E.d.C., and K.T., and H.J.K. revised the intermediate versions. K.M. contributed to this work while at Delft University of Technology and has moved to Bitcraze AB. K.T. contributed to this work while at the University of Liverpool and now has moved to DeepMind. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the article are present in the paper or the Supplementary Materials.

Submitted 9 April 2019
Accepted 30 September 2019
Published 23 October 2019
10.1126/scirobotics.aaw9710

Citation: K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, G. de Croon, Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Sci. Robot.* **4**, eaaw9710 (2019).

Science Robotics

Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment

K. N. McGuireC. De WagterK. TuylsH. J. KappenG. C. H. E. de Croon

Sci. Robot., 4 (35), eaaw9710. • DOI: 10.1126/scirobotics.aaw9710