AERIAL ROBOTS

Visual route following for tiny autonomous robots

Tom van Dijk*, Christophe De Wagter, Guido C. H. E. de Croon*

Navigation is an essential capability for autonomous robots. In particular, visual navigation has been a major research topic in robotics because cameras are lightweight, power-efficient sensors that provide rich information on the environment. However, the main challenge of visual navigation is that it requires substantial computational power and memory for visual processing and storage of the results. As of yet, this has precluded its use on small, extremely resource-constrained robots such as lightweight drones. Inspired by the parsimony of natural intelligence, we propose an insect-inspired approach toward visual navigation that is specifically aimed at extremely resource-restricted robots. It is a route-following approach in which a robot's outbound trajectory is stored as a collection of highly compressed panoramic images together with their spatial relationships as measured with odometry. During the inbound journey, the robot uses a combination of odometry and visual homing to return to the stored locations, with visual homing preventing the buildup of odometric drift. A main advancement of the proposed strategy is that the number of stored compressed images is minimized by spacing them apart as far as the accuracy of odometry allows. To demonstrate the suitability for small systems, we implemented the strategy on a tiny 56-gram drone. The drone could successfully follow routes up to 100 meters with a trajectory representation that consumed less than 20 bytes per meter. The presented method forms a substantial step toward the autonomous visual navigation of tiny robots, facilitating their more widespread application.

INTRODUCTION

To do useful tasks, mobile autonomous robots need to navigate through their surrounding environment. Unlike their fixed counterparts, mobile robots need to travel toward locations that are relevant to their mission or to return to their base after their mission has been completed. Currently, most autonomous robots rely on external infrastructure for localization and navigation, such as the Global Positioning System (GPS) outdoors (*1*) or ultra-wideband localization systems indoors (*2*). However, for many applications, this dependence on external infrastructure is undesirable. For one, the external infrastructure might not always be available, such as GPS in dense urban environments, extreme environments such as in caves, or when it is jammed. Second, it might be too impractical or time-consuming to set up additional infrastructure, especially in new or unknown environments such as search-and-rescue operations. Even in fully controlled environments, such as greenhouses or warehouses, costs might be another prohibitive factor. Hence, for many applications, it is vital that robots can navigate using only their own sensors, without reliance on external infrastructure.

Although options are abundant for larger robots, this is unfortunately not the case for smaller systems, such as the 56-g drone considered here (Fig. 1), or even tinier systems (*3*). First of all, the sensors might be too large, heavy, or power hungry for use on small platforms. This is, for instance, the case with light detection and ranging sensors (*4*), which otherwise provide a popular and high-precision solution for larger robots. Vision-based navigation could be a solution here because cameras are passive sensors that can be both very lightweight and power efficient (*5*, *6*). However, here, we run into the second issue: the excessive computational demands of the underlying vision algorithms. Mainstream approaches toward visual navigation tend to rely on simultaneous localization and mapping (SLAM) (*7*), a class of algorithms that typically construct and maintain detailed, metrically accurate maps of the environment while taking measurement uncertainties explicitly into account. The calculations that deal with these uncertainties and correct the map globally make SLAM computationally complex and memory intensive, requiring hundreds of megabytes to even several gigabytes to map medium-sized spaces in the order of tens of square meters (*8*). Achieving navigation with the maps built by visual SLAM additionally requires algorithms for path planning and trajectory tracking. The resulting computational and memory demands require powerful embedded processing units that far exceed smaller robots' payload capacity or power budget.

For this reason, there has been an increasing amount of work that focuses on more efficient navigation solutions. For instance, the class of topological SLAM methods saves on computation and memory by forgoing the construction of a global metric map (*9*, *10*). A major concern for these methods is the visual distinctiveness of different places in the environment because place recognition is necessary to globally correct the map. This typically still requires quite complex visual processing. Reducing the computational complexity of visual place recognition is hence an active research area (*11*, *12*). Other approaches to achieve more efficient navigation leverage the design of smaller, more powerful processors. For example, in (*13*), judicious software-hardware co-design led to a 2.4-mW–consuming custom-designed chip that was able to perform visual-inertial odometry (VIO). VIO algorithms form a subset of SLAM in which there is no recognition of already visited places (known as "loop closure"), and hence, they cannot eliminate odometric drift. Moreover, there is still some progress in scaling down existing processors despite the slowing down of processor miniaturization because of impending physical limits (*14*). Examples of lightweight computing units include Google Coral's tensor processing unit (TPU) (*15*), Intel's Neural Compute Stick (*16*), and the JeVois smart camera, for example, used in (*17*). Until now, these processors have not yet brought vision-based navigation to very small robots, like lightweight nanocopters. Furthermore, even if processing power increases in the future, it is

Control and Operations Department, Faculty of Aerospace Engineering, Delft University of Technology, Delft, Netherlands.
*Corresponding author. Email: j.c.vandijk-1@tudelft.nl (T.v.D.); g.c.h.e.decroon@tudelft.nl (G.C.H.E.d.C.)

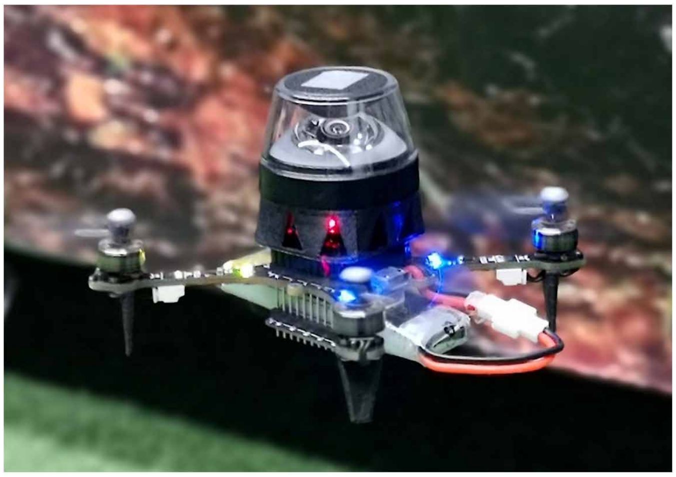

Fig. 1. Tiny, 56-g drone that performs autonomous visual route following with its panoramic camera. We propose an insect-inspired approach to visual route following that allows tiny, resource-restricted robots to follow long routes using only a microcontroller and exceptionally little memory.

questionable whether one will have the luxury to spend all that processing power on navigation. Real-world applications involve many other tasks, such as perception for obstacle avoidance or for recognizing mission-relevant objects. Hence, a parsimonious solution to navigation is and will remain highly relevant for small, autonomous robots.

Luckily, nature is a great source of inspiration for parsimonious solutions to navigation. Insects such as ants and bees can navigate over remarkable distances despite their tiny brains. For example, the desert ant *Cataglyphis* (*18*) can forage over long distances and then walk straight back to its nest, with journey lengths of up to 1 km. To bring a comparable algorithm to our robots, we must first understand how insect navigation works.

Biologists have studied insect navigation for more than a century and have revealed its two core ingredients (*19*). The first ingredient is path integration, which has a counterpart in robotics called "odometry," that is, the integration of traveled distance and direction to estimate one's position. For instance, ants determine the distance they have traveled by counting the number of steps they have taken. Moreover, they integrate ventral optical flow, tracking how fast the ground is seen moving past them (*20*). The direction is measured with respect to the sun and the corresponding polarization of the sky (*18*). Using these measurements, ants can maintain an estimate of their position relative to their nest (*19*). In recent work, the mechanisms behind path integration have been traced back all the way to the neural level, where ring attractors in the central complex play an important role (*21*–*23*).

Although path integration can provide an estimate of position, it has one major downside: It is susceptible to drift because it integrates its measurement errors. To solve this, nature uses a second mechanism: view memory. Here, the environment itself forms an additional cue for localization or navigation. There is less consensus on how this view of memory is used by insects (*24*–*26*). The most relevant model for our work is the snapshot model, proposed by Cartwright and Collet (*24*) to describe the homing behavior of bees. In this model, the authors posit that bees remember the presence and location of landmarks in their visual field, as seen at their goal location. Then, to return, they try to maneuver such that the landmarks in their field of view move back to their remembered positions.

Of course, the highly efficient nature of insect navigation has not gone unnoticed by roboticists (*27*, *28*). An early study focused on visual homing with the help of artificial landmarks and showed that visual homing can enable robots to return to a known spot in the environment (*29*). However, visual homing only works within a small region surrounding the target, called the catchment area. To navigate longer distances, a straightforward approach would be to home toward nearby targets in sequence. As long as the robot is inside the catchment area of the next snapshot, this will succeed. However, catchment areas tend to be limited in size. As a result, snapshots need to be spaced close together, which means that a large number of snapshots need to be stored to remember longer routes. Depending on the representation of the snapshot, which can range from full-resolution images (*30*) to more compressed representations (*31*), this can still require more memory than is available on tiny robotic platforms, like the 56-g drone used in this study.

To reduce memory requirements, further proposals have been made in two directions. The first is a reduction in the memory consumption of the snapshots. Many papers already reduced the images to a single row of pixels, where the lateral flow of features is sufficient for visual homing. However, Stürzl and Mallot (*31*) took this one step further by transforming this line into the frequency domain and remembering only the lowest-frequency components, thereby markedly reducing the size of the snapshot even further. The second direction is to increase the spacing between snapshots. In a study by Denuelle and Srinivasan (*32*), an improvement was proposed in which the homing vector was used as a position estimate relative to the snapshot. This allowed the drone to navigate some distance toward the next catchment area, provided that the vector was accurate enough. As a result, the overlap between snapshots was reduced, though not eliminated. A simulation study by Vardy (*33*) combined odometry with visual homing. New snapshots were taken when the odometry and visual homing estimates of the direction toward the snapshot started to diverge. However, because this happens at the edge of the catchment area, this method still resulted in considerable overlap between the catchment areas of subsequent snapshots. In this work, we propose an approach to substantially increase the distance between snapshots and combine it with a memory-efficient homing algorithm.

Minimal-memory approach to visual homing–based trajectory following

To bring visual navigation to tiny robots, we present a highly memory-efficient strategy for visual route following. We propose to traverse longer distances by better exploiting the combination of visual homing and odometry (Fig. 2). In this framework, we assume that the robot first performs an outbound flight toward a mission goal, followed by an inbound flight back to the starting location. The outbound flight can, in principle, be performed under any control law, including manual control. Because our focus here lies on route following during the inbound flight, we assume that the outbound flight is performed without collision and that the environment is static such that the route remains free of obstacles.

During the inbound flight, most of the distance is covered using odometry, but without any correction, the odometric drift would eventually become too large. To correct this drift, we let the robot use visual homing to periodically return to known locations in the environment (the snapshot locations). During homing, it compares its omnidirectional image only with the current active snapshot. After homing, the robot is at a known position and can reset its pose estimate, thereby eliminating any incurred odometric drift. After this reset, the robot can start a new leg of odometry and visual homing, where the initial error only consists of the homing inaccuracies at the last snapshot.

The proposed strategy is extremely memory efficient because the snapshots are spaced as far apart as possible. Specifically, we propose that the distance between snapshots is ultimately limited by the accuracy of the odometry, in that the strategy will succeed as long as the drone can reliably end up inside the next catchment area. Given a reasonable odometry accuracy, this distance can be far greater than when overlapping catchment areas for consecutive snapshots are required.

In the remainder of this article, we implement and evaluate this strategy step by step. First, we evaluate and compare different visual homing algorithms by their memory efficiency, where we take both the size of the snapshot and the resulting catchment area into

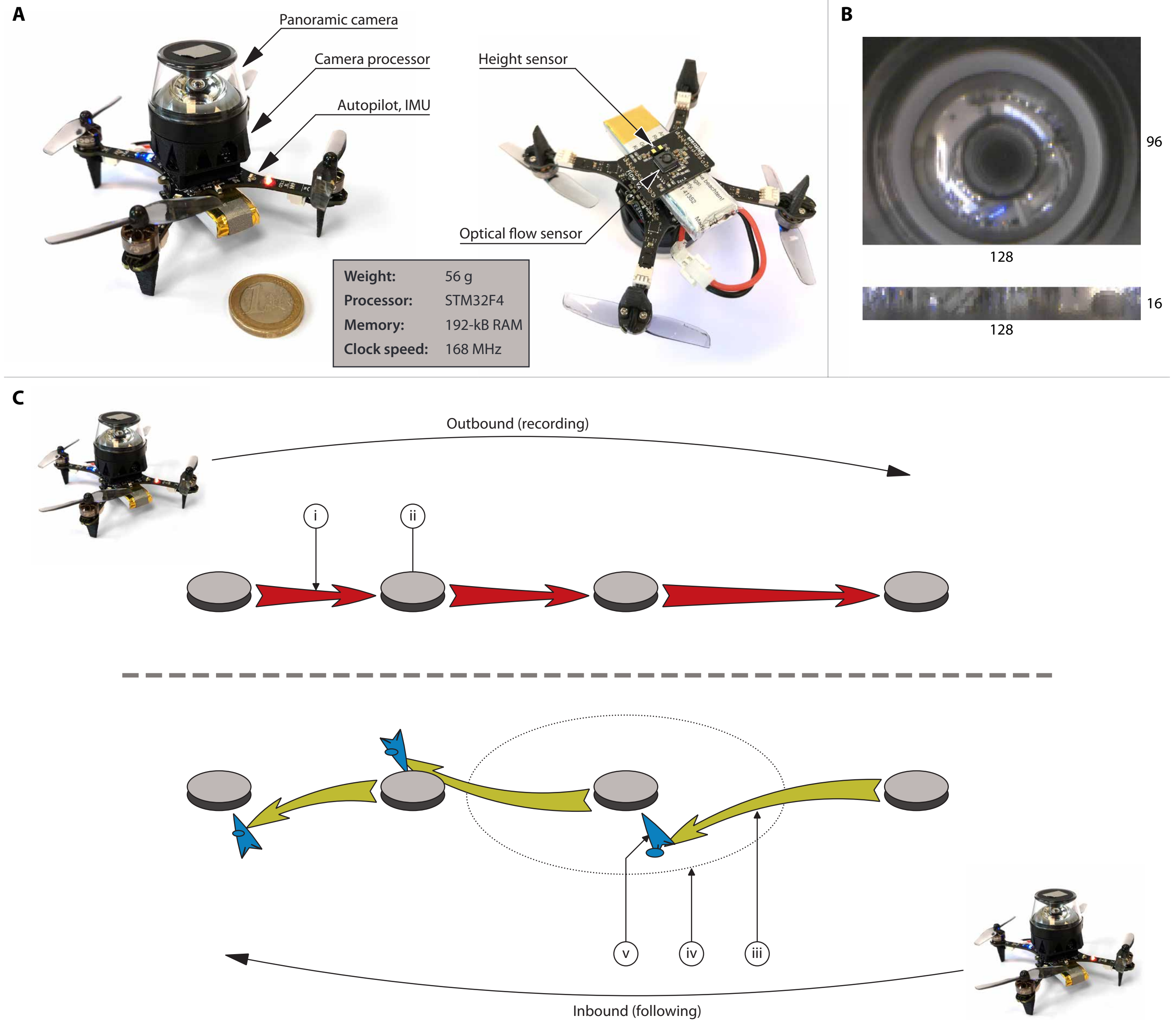

Fig. 2. Proposed navigation strategy. (**A**) The experimental platform considered here: a tiny, 56-g Crazyflie Brushless drone. (**B**) Raw and unwrapped omnidirectional camera image. (**C**) The route-following strategy. During the outbound trajectory, which could be performed under an arbitrary control law (i), the robot periodically takes snapshots (ii) of its surroundings. To follow the same route in reverse, the robot first uses odometry (iii) to move toward the location of the next snapshot. For success, it is vital that the drone ends up inside the catchment area (iv) of this snapshot. Hence, the distance between snapshots has to be proportional to the expected odometry drift and catchment area size. After the odometry movement has been completed, the robot uses visual homing (v) to converge to the snapshot location and thereby cancel the incurred odometric drift. These steps are repeated until the robot is back at its intended location.

account. After selecting a suitable algorithm for efficient visual homing, we continue with the key experiment of this article: We implemented our strategy on a tiny 56-g drone and let it follow trajectories of up to ~100 m. We show that periodically homing toward a snapshot eliminates the buildup of odometric error over time. Last, we demonstrate the same strategy on more complex trajectories and environments.

By implementing our strategy on a tiny, 56-g drone equipped with a microcontroller featuring a mere 192 kilobytes of memory, we show that the proposed algorithm is especially suitable for tiny robots. With our contribution, we bring autonomous visual navigation to a class of robots for which this was previously unavailable.

RESULTS

Selected homing algorithms for comparison

As a first step toward memory-efficient visual route following, we compared homing algorithms in terms of their memory efficiency. To aid the search for an efficient algorithm, first, we broadly categorized visual homing algorithms along two axes. We made a distinction between "steering methods" or "visual compass–based methods" and "vector methods." Steering methods are characterized by their calculation of a steering angle or rate for position control. Typically, these methods depend only on features in the forward field of view of the robot. Steering methods are often used in visual teach-and-repeat navigation tasks and have been successfully demonstrated over long distances, such as in (*34*–*37*). In addition, convergence has been proven for straight-line segments (*38*) and more complex paths (*36*, *39*).

On the other hand, vector-based methods typically do not produce a steering angle but a vector toward the snapshot. In contrast to steering-based methods, vector methods tend to not have an obvious "forward" direction and typically rely on a panoramic field of view. Vector methods are often used to home toward a single point, and they have also been used in visual route following such as in (*40*, *41*), although research in this direction has been far less extensive than for steering methods.

Although steering-based methods might have been an obvious choice, for this article, we chose to focus on vector-based approaches. Our reasoning was as follows: First, steering-based methods require their reference images to be spaced close together. Following the proof in (*38*), the distance between images must be smaller than the distance to the dominant feature in the environment; a distance of 35 cm was used in said article. We expected to achieve a far greater spacing using odometry. Second, we aimed to bring the robot as close to the snapshot as possible, ideally on top of it. As a result, we had no guarantee that the snapshot would be in front of the robot after traveling with odometry; it could just as likely have been behind it or to its side. Last, we thought that a vector is a more natural way to express movement for holonomic systems like our drone.

The second axis by which we categorized homing algorithms was the way in which snapshots are represented. We considered two broad categories of snapshot representations for use in visual homing: landmark-based and holistic representations. Landmark-based methods represent the snapshot as a collection of point landmarks that each have their own bearing. Consequently, in landmark-based visual homing, points in the environment are tracked from the current to the target image. From the point correspondences between these two images, a homing vector directed toward the target can be derived, for instance, using visual servoing (*42*–*44*). To describe and track landmarks, keypoint detectors and descriptors from computer vision are used. Examples include the computationally expensive scale-invariant feature transform (*45*) features and the much more efficient binary robust invariant scalable keypoints (*46*). In addition, the bearing toward each landmark needs to be stored. All in all, this still leads to a sizable memory consumption, especially when a large number of snapshots need to be remembered. To reduce memory consumption, it is possible to share descriptors between multiple snapshots as demonstrated by Stelzer *et al.* (*47*). However, the size of a snapshot remains in the order of hundreds of bytes or more.

On the other end of the spectrum are the holistic methods. Unlike landmark-based algorithms, these operate on the image as a whole. Instead of matching the bearings of landmarks, the entire current and target images are matched, for instance, with the sum of square differences. This leads to an image difference function (IDF) (see Fig. 3), which should be zero when the current and target view coincide and increase smoothly with distance near the target location. By finding the direction in which the IDF decreases, homing can be performed. One option to detect and follow the gradient down the IDF is to make physical movements, as demonstrated by Zeil *et al.* (*48*). Finding the gradient in this way may be time-consuming. Therefore, Franz *et al.* (*30*) proposed an alternative method in which small movements were simulated by warping the image. Using this method, the authors performed a brute-force search over multiple potential movements and selected the best match. Hence, we term the method "Search" here. As a computationally more efficient alternative, Möller *et al.* (*49*) suggested only predicting two perpendicular movements and using these to estimate the gradient of the IDF. By following the gradient, the robot will end up in the (local) minimum of the IDF. In later work, they included the second-order gradient as well. They termed their approach "MFDID." Storing entire images is not ideal in terms of memory efficiency; it is worse than most landmark-based approaches. A first improvement is that the snapshot images can be vertically averaged because just the lateral flow should already be sufficient to find the homing vector. On top of that, Stürzl and Mallot (*31*) showed that these one-dimensional snapshots could be substantially compressed while maintaining homing performance. This compression was performed by first transforming the snapshots to the frequency domain and then only keeping the lowest frequency components, where most of the power is found in natural images. The authors showed that homing was still possible using only the lower five components. With appropriate rounding, such a snapshot could be stored in as little as 10 bytes per snapshot. Figure 3A shows a raw panoramic image and its reconstruction using the highly compressed Fourier representation below. It can be observed that this method, which we term "Fourier," captures the coarse vertical structures in the environment. Because these images can ultimately be compressed further than the bearing-descriptor pairs of landmark-based homing, we focus on holistic algorithms for the remainder of this article.

Comparison of holistic visual homing algorithms

The selection of a specific visual homing algorithm meant exploring a trade-off between the catchment area size, memory consumption of snapshots, and computational demands. We evaluated the catchment areas from the abovementioned holistic vector-based methods (*30*, *31*, *49*) with the help of the publicly available panoramic image dataset by Gaffin and Brayfield (*50*). The dataset contains

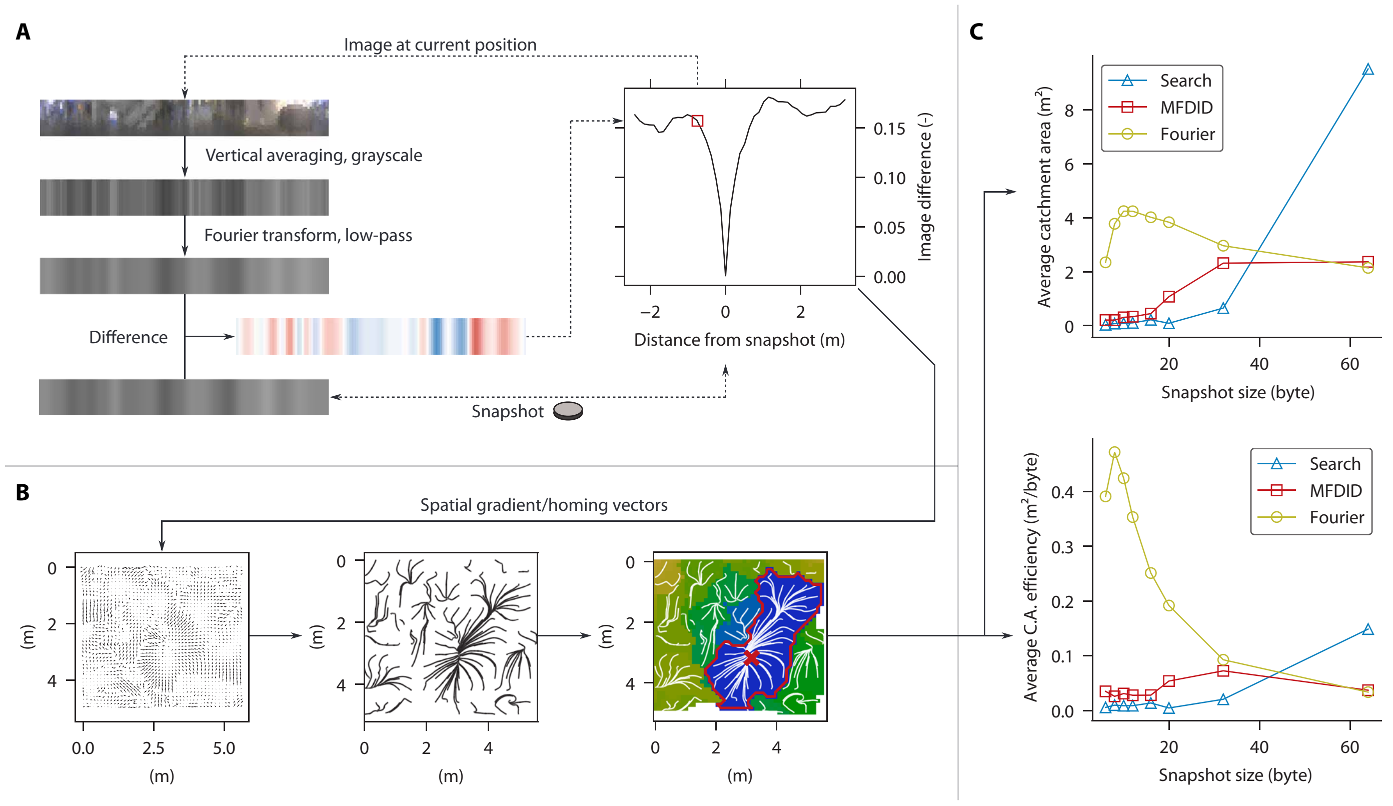

Fig. 3. Comparison of visual homing algorithms. (**A**) Illustration of the IDF with the Fourier method (*31*). Images were compressed using vertical averaging and a low-passed Fourier transform. The difference between the compressed image at the current and target location increases smoothly with distance, as shown in the plot. Finding the minimum in this IDF brought the robot back to the target position. (**B**) To evaluate the size of the catchment area belonging to a snapshot, we generated all homing vectors (left), predicted the robot trajectories (center), and collected all starting points for which the end-point error was small (right; error indicated by color). (**C**) Homing algorithms were compared by the size of their catchment area, given a snapshot size in bytes. At small snapshot sizes, Fourier-based homing performed best, whereas at larger sizes, search-based homing was more effective. In terms of efficiency in square meters of catchment area per byte, Fourier-based homing performed best.

100 pixel–by–100 pixel grayscale images taken at 12.7-cm intervals in a 7.3 m–by–6.9 m room and part of the adjacent corridor. To evaluate the catchment area, we generated snapshots throughout the environment, and for each snapshot, we calculated the homing vectors at all panoramic image locations in the room (Fig. 3B). Then, from each starting position, homing trajectories were generated by integrating the bilinearly interpolated homing vectors. These trajectories let us determine the final position of the robot after homing and thereby find the set of starting locations from which homing would be successful, that is, the catchment area. Because the catchment area can have a highly irregular shape (see the dark blue area in Fig. 3B), we took the number of successful starting cells as a measure of the size of the catchment area. The top plot in Fig. 3C shows the relationship between the snapshot size in bytes (indicated with the letter B) and the average catchment area in square meters. The Fourier method led to the largest catchment areas for snapshot sizes below 32 bytes, whereas the Search method gave the largest catchment areas above. The lower plot in Fig. 3C shows the average ratio of the catchment area with respect to the snapshot size. It shows that the Fourier method is highly efficient for small snapshots. Because this algorithm is both memory-efficient and computationally cheap, it was used in our further experiments. Please note that the choice for this representation entailed a dependence on contrasts along the horizon line and mainly large vertical features, such as walls, doors, windows, and trees. Furthermore, the idea of snapshot matching relies on these contrasts being static.

Visual homing and odometry

Before combining visual homing and odometry into a single navigation strategy, we examined these elements in isolation. We first validated the homing performance of the selected visual snapshot representation on our robotic platform, a 56-g, Bitcraze Crazyflie Brushless (Figs. 1 and 2A). This tiny 12.5-cm drone carried a 10-g panoramic camera assembly (included in the 56-g takeoff weight). The assembly included an STM32F4 chip for processing the omnidirectional images onboard in real time. Furthermore, the drone was equipped with a "flow deck" with a downward-looking camera and a tiny laser ranger to measure optical flow and height, respectively. Combining these measurements results in velocity estimates, which were used for odometry.

In the homing experiment, the drone was first directed to the center of our testing environment, a 10 m–by–10 m–by–7 m flight arena termed the "Cyberzoo" (see Fig. 4C for an impression). Subsequently, we commanded the drone to go to a small number of locations away from the target, to a maximum distance of approximately 2 m. Then, the drone performed visual homing with the Fourier method. Figure 4 shows an overhead view of the drone's trajectories (Fig. 4A), the distance to the target location over time during homing (Fig. 4B), and an overview image of the flight arena (Fig. 4C). We observed that eight of nine runs brought the drone close to the target location, within ~0.5 m, at which point the run was ended. The failed run started at one of the outer positions. By definition, the failed homing attempt means that the starting position was outside of the (unknown) catchment area of the snapshot in the center. The experiment also showed that the drone did not always fly straight to the target location, an indication that the homing vector can point in a different direction than the target vector. As long as the homing vector is within ±90°, however, the distance will decrease, and the drone will eventually arrive at the target location.

As explained earlier, our strategy relies on spacing the snapshots as far apart as odometry allows. Specifically, the drone should end up just inside the next catchment area. The drone's positioning accuracy depends on two main factors: the accuracy of its starting pose after homing and the drift incurred while moving toward the new position. This was demonstrated by a simulation of multiple trajectories in Fig. 5A, in which the SDs of the initial position and heading, yaw rates, and velocity errors were exaggerated to demonstrate their effect. The model showed that at short distances, the position error was primarily caused by errors in the initial pose because it was almost constant. For longer distances, the error begins to grow because of integrated odometry errors. At the end of a route leg traveled by means of odometry, the cross-track error was larger than the along-track error. This can be seen in Fig. 5A, where there is a larger spread orthogonal to than along the route, and in Fig. 5C, in which this is also the case for real-world odometry experiments. This effect is caused by the heading error, consisting of an initial offset and subsequent drift. Figure 5B shows the absolute cross-track errors of the drone experiment. Overall, the accuracy was quite good, with a cross-track root mean square error of 13 cm after 5 m of travel.

The plots in Fig. 5 (C and D) show a similar experiment but over longer distances. The drone traversed a line of approximately 5 m back and forth 10 times. In Fig. 5C, the drone only used odometry for this procedure. In Fig. 5D, the drone recorded a snapshot in the top left corner and used this to realign itself on each arrival there. The results show that the odometry did drift and that the drift became substantial for longer distances. They also show that our periodic realignment scheme, while introducing some error because of homing inaccuracies, prevented the buildup of odometric drift over time and, as a result, kept the error bounded when traveling longer distances.

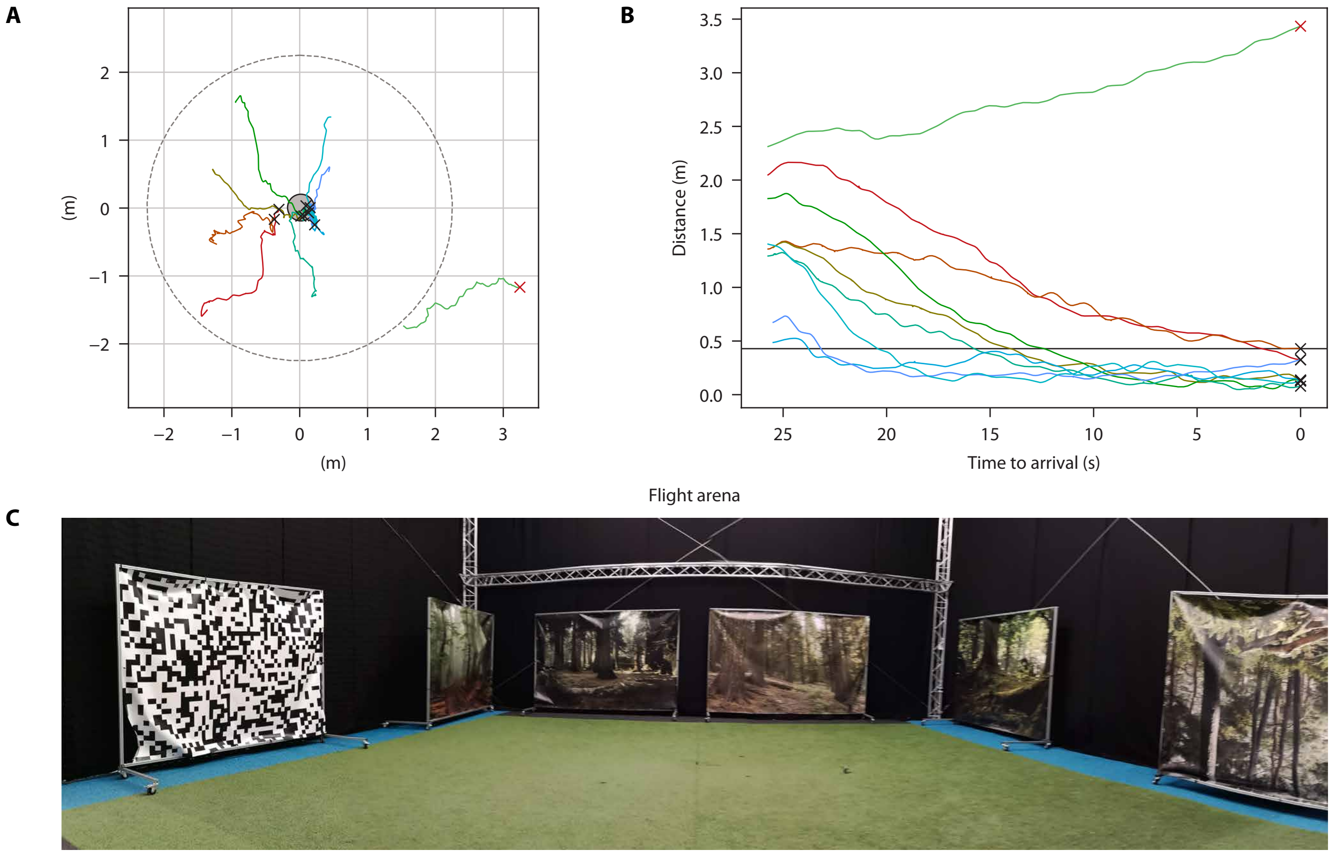

Fig. 4. Visual homing toward a single point. The drone was commanded to home over longer distances toward a snapshot at the center of the flight arena (gray circle). (**A**) Homing trajectories shown as colored lines, with crosses for the end positions. (**B**) The decreasing distance toward the snapshot over time, shown with colored lines corresponding to the trajectories in (A). The black line indicates the worst homing performance attained in this experiment for trajectories converging to the snapshot. For illustration purposes, in (A), a possible catchment area is drawn with a dashed line. One of the starting positions lay outside of the catchment area and led the drone to diverge (red cross as the end position). Homing was successful for distances well more than 1 m, whereas the odometric drift between snapshots during route following was expected to be substantially smaller. (**C**) Picture of the environment setup for this experiment.

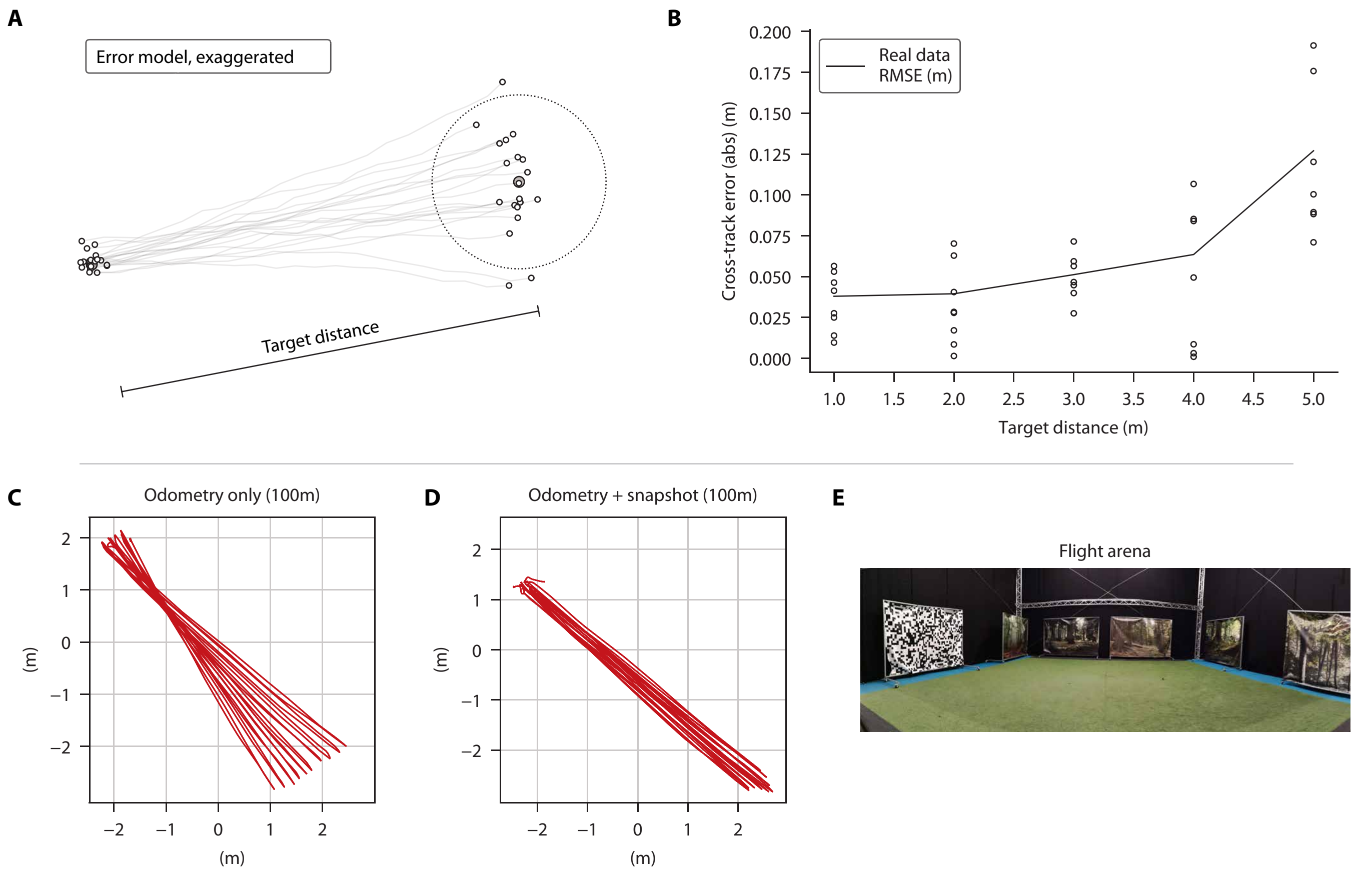

Fig. 5. Combining visual homing and odometry. Each leg of the route-following strategy consists of a homing operation toward a snapshot, followed by an inbound maneuver toward the next snapshot using odometry. (**A**) The error model with exaggerated noise parameters showing both the initial pose error after homing and the increased spread because of odometry. (**B**) Experimental data for real-world odometry experiments in which the traveled distance was varied. (**C** and **D**) Proof-of-concept demonstration. The drone was commanded to fly a 5-m trajectory back and forth for a total distance of 100 m. In (C), the drone only used odometry, whereas in (D), it periodically homed to a snapshot. Repeatedly homing toward snapshots was shown to prevent drift over longer trajectories. (**E**) Overview photo of the test environment.

Route following with minimal memory

With the core principles proven, we then demonstrated the complete strategy on more complex trajectories and environments. We created different types of trajectories, of which the outbound portion was traversed using odometry (without any global position feedback). After the outbound journey was completed, the drone started its inbound journey with the help of the proposed insect-inspired navigation strategy. We qualitatively compared the route-following accuracy with respect to the outbound trajectory. A motion capture system was used to record the absolute position of the drone during its outbound and inbound flights. These measurements were never communicated to the drone; they were only used for evaluation after the experiment. The trajectories consisted of multiple traversals of an S shape (Fig. 6A) or a U shape (Fig. 6B). The trajectories were repeated to maximize the travel distance within the limited testing area. The resulting path lengths (of the outbound route) were 40 m for the S-shaped trajectory and 56 m for the U-shaped one. Ultimately, the length of the experiment was limited by the battery capacity of the drone.

When introducing our minimal-memory approach to visual homing–based trajectory following, we mentioned that snapshots are spaced as far as odometry allows. Given the varying unknown shapes and sizes of catchment areas and the variable nature of drift, choosing a spacing between snapshots has implications for the trade-off between navigation robustness and memory expenditure (see the "Theoretical model for spacing snapshots" section in the Supplementary Materials). In our experiments, we used a fixed 1- or 2-m spacing between snapshots during the experiments. First, these were conservative values where the position error was primarily dominated by homing inaccuracies, whereas the odometric drift between snapshots remained small. Second, this gave us a larger number of visual homing attempts, which allowed a better indication of its use and robustness during route following.

Figure 6 shows the resulting trajectories for the proposed method (Fig. 6, A and B). The robot successfully and reliably followed the outbound trajectory back to the start. The route-following memory for the U trajectory consisted of 31 16-byte snapshots and 2 to 3 2-byte odometry vectors between snapshots, leading to a total memory size of 0.65 kilobytes for a distance of 56 m. We also compared our method with sequential visual homing on one of the trajectories (Fig. 6C). The experiment showed that homing between successive snapshots is a viable method of navigation. However, the snapshots had to be spaced at a distance of 25 cm; earlier attempts with 1-m spacing consistently failed. Because the catchment areas scale

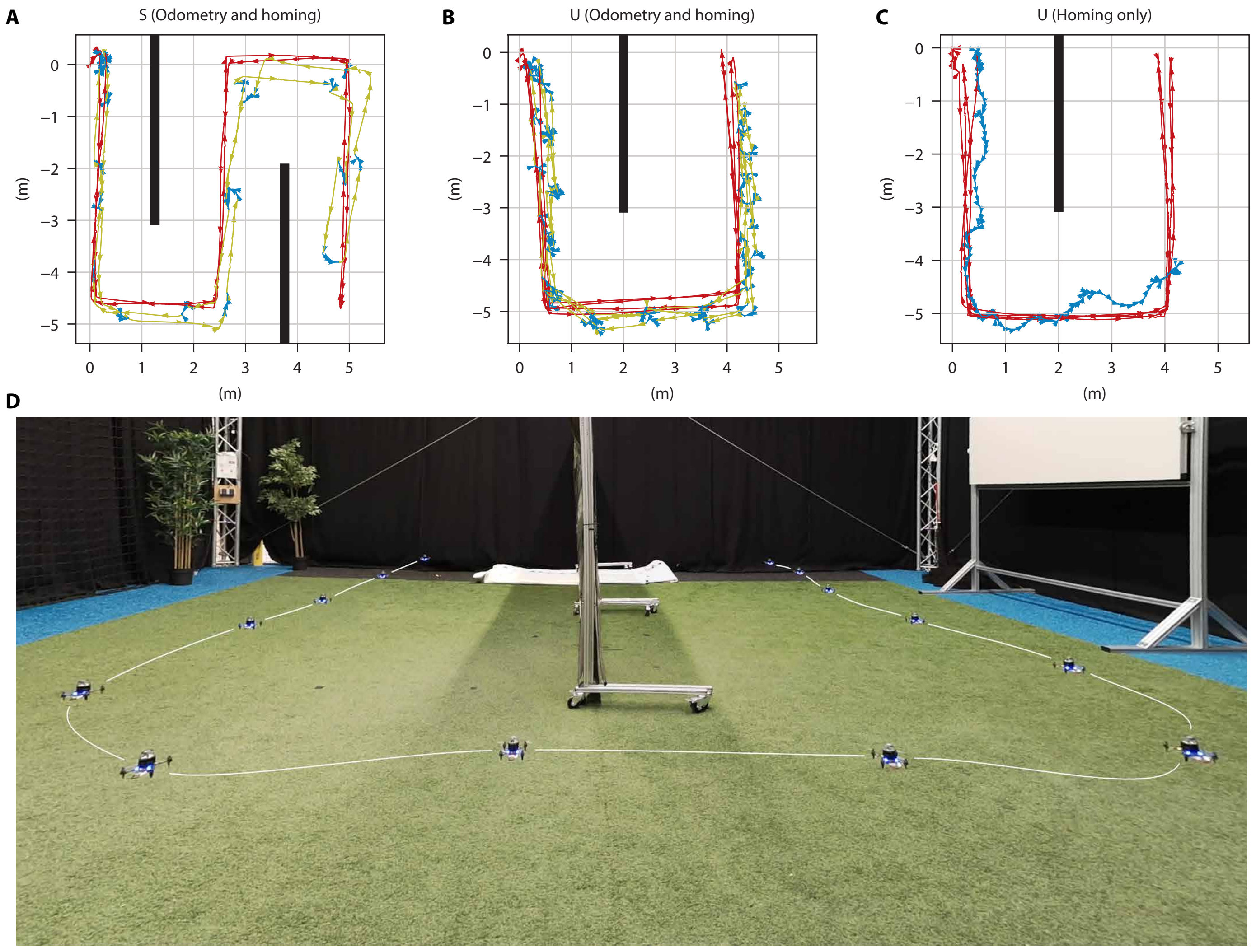

Fig. 6. Validation of the complete route-following strategy. The drone successfully followed long routes in complex environments where it was impossible to see the entire route from a single point of view. The outbound route is shown in red, route following using odometry in yellow, and homing in blue (**A** to **C**). The drone traversed the route multiple times during the outbound and inbound segments to maximize the length of the trajectory (all on the order of ~50 m). For comparison, a route-following attempt in which the drone only used sequential visual homing as a strategy is shown (C). The drone successfully followed the route but stopped early because the increased travel time caused the battery to run out before completing one stretch of the trajectory. A time-lapse photo of the experiment in (B) is shown in (**D**).

proportionally to the size of the environment (distance to the dominant features, the walls in this case), they were considerably smaller than in Fig. 4. Besides the increased memory consumption, the homing procedure was also relatively slow (to prevent overshoot or large pitch/roll angles), and as a result, the drone traveled an appreciably shorter distance before the battery ran out. In comparison, the proposed strategy had a substantially higher average speed, whereas the tracking error had the same order of magnitude.

We also performed a number of experiments to evaluate the robustness of the proposed approach. One important characteristic of the chosen snapshot representation (with vertical averaging and Fourier compression) is that it depends on prominent vertical features (contrasts along the horizon line). Such features are commonly present in both indoor and outdoor environments. To illustrate this, we made a set of snapshots in various places in the building of the Faculty of Aerospace Engineering at TU Delft (see the "Presence of texture in different environments" section in the Supplementary Materials). The low resolution of the snapshots may initially seem purely disadvantageous for accurate homing but actually brings some robustness against small, dynamic objects. This is illustrated with an experiment in which we monitored the resulting home vector, whereas we moved objects around the robot (tables S1 to S4). Sometimes in indoor environments, there are corridors with purely uniform walls. In that case, the current approach will not be able to correct the drift in the direction of the corridor (lateral drift can be canceled because of the different appearance of the floor and walls). Furthermore, to show that the drone is also able to follow routes in different indoor environments, we performed additional experiments in three different places at the Faculty of Aerospace Engineering: close to an airplane simulator SIMONA (*51*), in an office hallway, and in our laboratory space (Fig. 7). Videos of these flights can be found in the Supplementary Materials. Last, because the flight time of the real drone is limited, we performed simulation experiments to show that also longer distances can be covered with the proposed strategy. Specifically, in the AirSim simulator (*52*), a simulated Parrot augmented reality (AR) drone was able to use the strategy to successfully track a 300-m trajectory in a forest environment (fig. S2).

DISCUSSION

We have proposed an insect-inspired navigation strategy for route following, which extensively depends on odometry to reduce memory usage. The strategy was demonstrated on the lightest robot to date to perform vision-based navigation, a 56-g Crazyflie drone, leading to successful navigation for as long as the battery lasted.

The experiments show that even tiny robots can navigate autonomously. Of course, the strategy studied here is a route-following strategy. This sets it apart from SLAM-based navigation approaches and has two important implications. First, the robot will perform an inbound route that is identical to the outbound route. This is similar to other teach-and-repeat methods in robotics (*37*, *53*). By not going straight back to the starting location, as insects are known to do (*18*), the robot follows a trajectory that is suboptimal in terms of path length. In contrast, robots performing metric SLAM can plan and execute optimal paths at the expense of considerable processing. How to perform straight returns to the initial location while coping with odometric drift without metric maps is an important topic for future study. Second, the current algorithm is unable to combine multiple paths to move between arbitrary locations in the environment. It will not be able to fly to another previously visited place that is relevant to its task without first returning to the home location. Also, insects can do this, and it is hypothesized that insects may store different places in a nest-centered reference frame (*19*). Such a representation is reminiscent of topological mapping approaches, which may provide fertile ground for integration with the method proposed here. For example, in (*54*), new nodes (places) were created in a topological map by estimating online whether the robot was about to leave the catchment area of the previous node. This led to overlapping catchment areas, an idea very similar to that of Vardy (*33*) in the insect route-following literature. The proposed idea of further spacing snapshots (or nodes in the topological graph) apart is directly relevant to such a topological mapping approach.

In terms of biological plausibility, we do not believe that the proposed strategy is an accurate model for explaining insect navigation behavior. Although some switching between path integration and visual homing occurs when insects move from unknown to known environments, behavioral experiments show that they mostly use these cues simultaneously (*55*, *56*), in contrast to our strategy. However, our results do support the general idea that path integration and visual homing are best used in combination. Even more, they suggest that path integration has a marked effect on the efficiency and parsimony of navigation even when other (visual) cues are present and encourage further research into the integration of these cues (*19*, *56*, *57*).

In the field of robotics, the performed experiments are highly encouraging because they show that tiny robots are also able to perform vision-based autonomous navigation. Future work could focus on improved robustness by introducing obstacle-avoidance capabilities. For instance, the omnidirectional image could

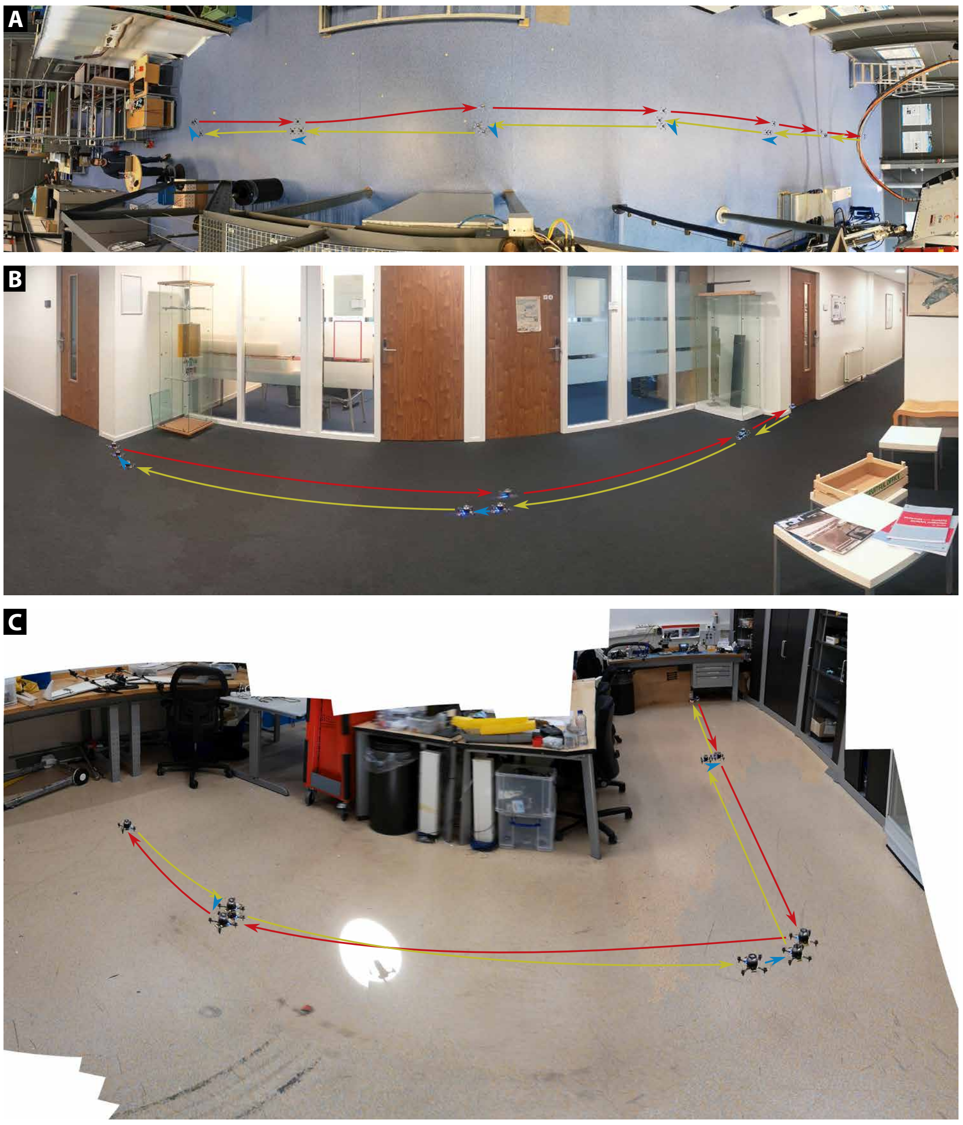

Fig. 7. Flights in different indoor environments. To test the robustness of the proposed visual navigation method, we performed tests in various indoor environments. A time-lapse image of each test is shown for three environments: (**A**) a large indoor test facility for airplane simulation (SIMONA), (**B**) an indoor office hallway, and (**C**) the Micro Air Vehicle Laboratory space (the bright spot on the floor is due to direct, bright sunlight). The outbound trajectory is indicated by red arrows, odometry-based indoor trajectories by yellow arrows, and homing maneuvers by blue arrows.

be used to determine optical flow for collision avoidance (*58*, *59*). Moreover, the robot could estimate the catchment area size online and be endowed with a search procedure when losing the route. Additional experiments in varying environments could identify which elements of the method need the most improvement. Still, as already hinted above, the choice for route following implies a cost in terms of tracking accuracy and flexibility in navigation targets. Even if we further approach the impressive navigation capabilities of insects, it may be that for these characteristics, a traditional SLAM-based navigation approach remains superior. However, for autonomously navigating robots, optimality in terms of mass and energy expenditure is also important. This is definitely the case for applications such as greenhouse monitoring by flying robots. The driving factors for that application are safety and navigation in narrow, cluttered environments. Tiny, lightweight flying robots are hence ideal for such an application, in which it is most important that the robots fly out, gather data, and come back to a fixed charging station. The data can then be uploaded to a server for mission-specific processing, such as evaluating crop growth or disease detection. We expect that the best navigation solution will eventually be task-dependent and venture that even for much bigger robots on the order of kilograms, insect-inspired navigation may be the best option when efficiency is more important than high positioning accuracy along the trajectory. If orders of magnitude in computation and memory can be saved from the task of navigation, this computation can be used for onboard mission-relevant tasks such as the recognition of diseases or pests in a greenhouse application or the counting of products in a warehouse monitoring application. Hence, the current work will not only benefit tiny robots, such as the 56-g drone used here or even the insect-sized Harvard RoboBee (*60*), but much larger robots as well.

MATERIALS AND METHODS

Hardware

The experiments here were performed on a prototype of the Crazyflie Brushless drone provided by Bitcraze. The Flowdeck V2 (PMW3901 optical flow sensor, VL53L1x laser ranger) was used for velocity and altitude control and odometry. For navigation, the drone was equipped with a TCM8230MD camera with a Kogeto Dot 360 panoramic lens. Processing was performed using two STM32F4 microcontrollers, one on the autopilot and one on the camera assembly. Visual processing was performed on the camera microcontroller; state estimation and control were performed on the autopilot. Logging was performed offboard using the radio link.

The default Crazyflie firmware was used as autopilot. State estimation was performed with the default extended Kalman filter. A custom onboard app communicates with the camera over a UART link and sends position set points and measurement updates to the autopilot's controller and estimator.

Image processing

Image processing began with the preprocessing of the camera frames. Raw images were captured at a 128 pixel–by–96 pixel resolution (Fig. 2B). A custom autoexposure routine adjusted the shutter time to keep the horizon's mean luma at a fixed value (80 out of 255) while ignoring the rest of the image (including the lens fixture).

The image was then reprojected to cylindrical coordinates at a 128 pixel–by–16 pixel resolution. We used a look-up table and nearest-neighbor sampling for computational efficiency. The cylindrical images were aligned with the drone's north estimate by offsetting the sampling angle. Derotation of pitch and roll angles was implemented but not used because the angles during the experiment remained sufficiently small. The images were then converted to grayscale and vertically averaged to produce a one-dimensional periodic signal. We used the fast Fourier transform from the ARM CMSIS DSP library to transform this signal to the frequency domain. For memory efficiency, the direct current (DC)– and higher-frequency components were dropped. The remaining complex coefficients were quantized to pairs of 8-bit signed integers, with a fixed per-frequency scaling to cover most of the 8-bit range.

Homing implementation

For the comparison of homing methods, we implemented the algorithms by Franz *et al.* (*30*) (Search), Möller *et al.* (*49*) (MFDID), and Stürzl and Mallot (*31*) (Fourier) as described in the respective papers. For each choice of snapshot size, the parameters were retuned using a grid search to maximize the size of the catchment area. For the Search algorithm, we used a bearing-distance search grid, because the exact grid was not described in the article. For MFDID, we included the use of the Newton-based correction (*61*) as part of the parameter search but found little difference in this dataset, which mainly consists of a square, open room.

The Fourier-based homing algorithm by Stürzl and Mallot (*31*) was also implemented on the experimental hardware. For efficient implementation, we wish to highlight an important property of this algorithm. To find the homing vector, Stürzl and Mallot (*31*) derived and minimized an "approximate IDF" $\mathscr{E}_2$ in terms of hypothetical movement $h \in R^3$ in the form of a quadratic surface:

$$\mathscr{E}_2(h) = \frac{1}{2}h^\top A h + b^\top h + c$$

Here, A, b, and c are fully defined by the complex coefficients of the Fourier-transformed, derotated images. As a result, the homing vector can be found using only the fast Fourier transform and a three-by-three matrix inversion, which makes this algorithm highly efficient in terms of run time. For the full definition, we refer to (*31*).

Although the images are already coarsely aligned with respect to the north estimate, we did include the coarse rotation alignment step of the algorithm. However, we replaced the phase-based algorithm with a brute-force search over all possible rotations, because we found that this provided more robust results in practice. The phase-based algorithm appeared to lack robustness when symmetries were present in the environment or when lower frequencies were absent in the panoramic images, although we did not fully investigate this further.

For the sequencing of snapshots, it is important to detect arrival after homing. This was initially determined by observing the difference between the currently relevant snapshot and the current observation. If this difference did not reach a new minimum during the last 10 frames (~1 s), the drone was considered to have arrived. Although this worked, it resulted in long hover times near the snapshots. In the final experiments, this detection was replaced by a simple timeout. This considerably reduced hovering times and thereby allowed longer travel distances, at the cost of a slightly higher homing position error.

Route memory

The "route memory," containing the snapshots and odometry vectors, was implemented in memory using two stacks (Fig. 8). The odometric trajectory was recorded as a sequence of translation vectors. While recording, a new vector was pushed onto the stack every 0.3 m. These vectors were stored as a pair of 8-bit signed integers, with a resolution of 10 cm. Each new vector was calculated by comparing the current position estimate with the sum of all previous vectors; this prevented the buildup of rounding errors. To reduce memory consumption, the recorded trajectory was simplified when a new snapshot was taken. At that time, the complete trajectory since the last snapshot was decimated using the Ramer-Douglas-Peucker algorithm (*62*, *63*) ($\epsilon = 0.1$ m). This strongly reduced the number of odometry vectors while keeping the resulting deviation within strict bounds. Additional vectors may be stored to keep the lengths within 8-bit integer bounds. After simplifying the trajectory, the snapshot was pushed onto its own stack together with the size of the odometry stack at that time so that its position in the odometry frame could be retrieved during route following.

During route following, after each odometry segment, the drone should be close to the true position of the snapshot on top of the stack (Fig. 8). The drone then started a homing maneuver toward this snapshot, removing the need for along-route localization. Once the homing maneuver was completed, the drone realigned its position and heading estimate to those of the snapshot as recorded by the odometry vectors and the north alignment of the snapshot. In practice, this realignment was implemented as an absolute position and heading measurement with a small covariance for easier integration with the Kalman filter.

Experimental setup

During the flight experiments in Figs. 4 to 6, the true position of the drone was measured using an OptiTrack motion capture system. These position data were not communicated to the drone at any time; they were only recorded for analysis after the flight. The time stamps were aligned by maximizing the cross-correlation between the north or east positions in both log files. The outbound trajectories during the flight experiments did not use the true position either; these relied entirely on the onboard position estimate.

The experimental flights shown in Figs. 4 to 6 were performed over artificial grass. Judging by the performance of odometry-only navigation, this provided sufficient texture for the Crazyflie's optical flow sensor. The drone used its downward-facing laser range sensor to maintain a constant height during the experiment. On the sides of the flight area, canvas panels with mostly natural scenes provided additional texture for navigation. For the U- and S-shaped trajectories, these panels were also placed in the center of the flight area to block the line of sight between the extreme points.

The experimental flights shown in Fig. 7 were performed at different locations in the Faculty of Aerospace Engineering: in a large open space of the SIMONA airplane simulator, a narrow corridor, and the Micro Air Vehicle Laboratory. Each experiment started with a manually designed, preprogrammed outbound flight that was executed on the basis of odometry. The outbound flight was followed by an autonomous inbound flight. Figure 7 contains stitched time-lapse images, in which we show the drone when making the snapshot during the outbound flight (red arrows), when the drone estimates arriving at the snapshot location with odometry during the inbound flight (yellow arrows), and after homing to the snapshot (blue arrows). The videos can also be found in the Supplementary Materials.

Simulation setup

Figure 5A shows the results of elementary simulation experiments to illustrate the effects of odometry drift. The simulation included both an initial position and heading offset because of imperfect homing and odometry drift along the route leg. Hence, we initialized the pose as $x, y, \psi \leftarrow x_0 + \mathcal{N}(\sigma_{x0}), y_0 + \mathcal{N}(\sigma_{y0}), \psi_0 + \mathcal{N}(\sigma_{\psi 0})$, where (x_0, y_0, ψ_0) is the actual snapshot position and $\mathcal{N}(\sigma)$ is a normally distributed random variable with zero mean and SD σ. Then, for each time step, the state was updated as follows:

$$\psi \leftarrow \psi + \mathcal{N}(\sigma_\psi)$$

$$x \leftarrow x + \Delta x \cos\psi + \mathcal{N}(\sigma_x)$$

$$y \leftarrow y + \Delta x \sin\psi + \mathcal{N}(\sigma_y)$$

For the simulation in Fig. 5A, the following values were used: $\Delta x = 0.25$ m, $\sigma_{x0} = \sigma_{y0} = 0.10$ m, $\sigma_{\psi 0} = 5^\circ$, $\sigma_x = \sigma_y = 0.025$ m, and $\sigma_\psi = 2^\circ$. Please note that these values are large compared with the real homing and drift errors so that the figure shows the effects they have on the position error at the end of the route leg.

Statistical tests

No statistical tests were performed for this article.

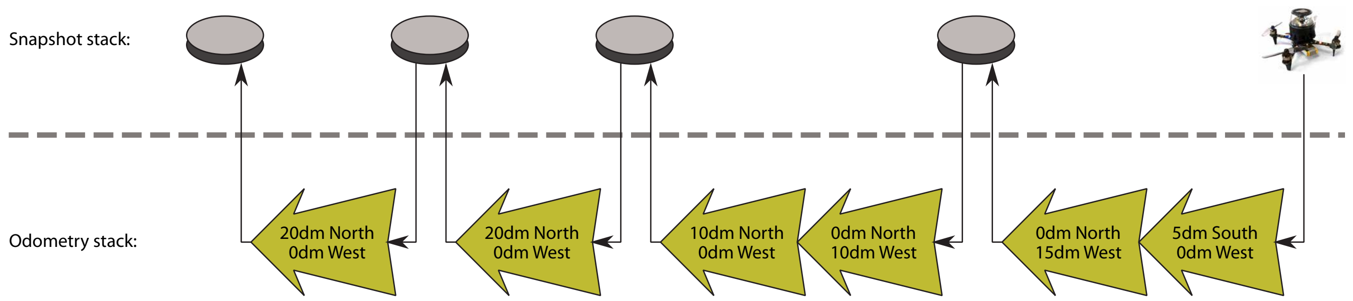

Fig. 8. Route representation in memory. The trajectory was represented using two stacks: one holding snapshots and one holding odometry vectors. When a new snapshot was pushed, it was stored together with the number of odometry vectors that were present at that time. This allowed the snapshot position to be found by adding all odometry vectors before it and aided in the sequencing of odometry and homing maneuvers. A single odometry vector consisted of two int8 numbers at decimeter resolution (2 bytes). A snapshot consisted of eight complex coefficients (2× int8, so 16 bytes in total) plus a uint16 odometry count (2 bytes).

Supplementary Materials

The PDF file includes:
Methods
Figs. S1 to S8
Tables S1 to S4

Other Supplementary Material for this manuscript includes the following:
Data file S1
Movies S1 to S13

REFERENCES AND NOTES

1. K. P. Valavanis, G. J. Vachtsevanos, Eds., *Handbook of Unmanned Aerial Vehicles* (Springer Netherlands, 2015).
2. M. W. Mueller, M. Hamer, R. D'Andrea, Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation, in *2015 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2015), pp. 1730–1736.
3. S. B. Fuller, Four wings: An insect-sized aerial robot with steering ability and payload capacity for autonomy. *IEEE Robot. Autom. Lett.* **4**, 570–577 (2019).
4. T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, A. Hussain, A survey on LiDAR scanning mechanisms. *Electronics* **9**, 741 (2020).
5. S. Balasubramanian, Y. M. Chukewad, J. M. James, G. L. Barrows, S. B. Fuller, An insect-sized robot that uses a custom-built onboard camera and a neural network to classify and respond to visual input, in *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)* (IEEE, 2018), pp. 1297–1302.
6. S. Viollet, S. Godiot, R. Leitel, W. Buss, P. Breugnon, M. Menouni, R. Juston, F. Expert, F. Colonnier, G. L'Eplattenier, A. Brückner, F. Kraze, H. Mallot, N. Franceschini, R. Pericet-Camara, F. Ruffier, D. Floreano, Hardware architecture and cutting-edge assembly process of a tiny curved compound eye. *Sensors* **14**, 21702–21721 (2014).
7. H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **13**, 99–110 (2006).
8. B. Bodin, H. Wagstaff, S. Saecdi, L. Nardi, E. Vespa, J. Mawer, A. Nisbet, M. Lujan, S. Furber, A. J. Davison, P. H. J. Kelly, M. F. P. O'Boyle, SLAMBench2: Multi-objective head-to-head benchmarking for visual SLAM, in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018), pp. 3637–3644.
9. J. Boal, Á. Sánchez-Miralles, Á. Arranz, Topological simultaneous localization and mapping: A survey. *Robotica* **32**, 803–821 (2014).
10. E. Garcia-Fidalgo, A. Ortiz, Vision-based topological mapping and localization methods: A survey. *Robot. Auton. Syst.* **64**, 1–20 (2015).
11. S. Hussaini, M. Milford, T. Fischer, Spiking neural networks for visual place recognition via weighted neuronal assignments. *IEEE Robot. Autom. Lett.* **7**, 4094–4101 (2022).
12. B. Ferrarini, M. Milford, K. D. McDonald-Maier, S. Ehsan, Highly-efficient binary neural networks for visual place recognition, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2022), pp. 5493–5500.
13. A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, V. Sze, Navion: A 2-mW fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. *IEEE J. Solid-State Circuits* **54**, 1106–1119 (2019).
14. J. Shalf, The future of computing beyond Moore's law. *Philos. Trans. R. Soc. A* **378**, 20190061 (2020).
15. Y. Sun, A. M. Kist, Deep learning on edge TPUs. arXiv:2108.13732 [cs.CV] (2021).
16. N. Tiwari, K. Mondal, NCS based ultra low power optimized machine learning techniques for image classification, in *2019 IEEE Region 10 Symposium (TENSYMP)* (IEEE, 2019), pp. 750–753.
17. S. Li, E. van der Horst, P. Duernay, C. De Wagter, G. C. H. E. de Croon, Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone. *J. Field Robot.* **37**, 667–692 (2020).
18. R. Wehner, M. V. Srinivasan, Path integration in insects, in *The Neurobiology of Spatial Behaviour* (Oxford Univ. Press, 2003), pp. 9–30.
19. B. Webb, The internal maps of insects. *J. Exp. Biol.* **222**, jeb188094 (2019).
20. B. Ronacher, R. Wehner, Desert ants *Cataglyphis fortis* use self-induced optic flow to measure distances travelled. *J. Comp. Physiol. A* **177**, 21–27 (1995).
21. S. S. Kim, H. Rouault, S. Druckmann, V. Jayaraman, Ring attractor dynamics in the *Drosophila* central brain. *Science* **356**, 849–853 (2017).
22. T. Stone, B. Webb, A. Adden, N. Ben Weddig, A. Honkanen, R. Templin, W. Wcislo, L. Scimeca, E. Warrant, S. Heinze, An anatomically constrained model for path integration in the bee brain. *Curr. Biol.* **27**, 3069–3085.e11 (2017).
23. J. Green, A. Adachi, K. K. Shah, J. D. Hirokawa, P. S. Magani, G. Maimon, A neural circuit architecture for angular integration in *Drosophila*. *Nature* **546**, 101–106 (2017).
24. B. A. Cartwright, T. S. Collett, How honey bees use landmarks to guide their return to a food source. *Nature* **295**, 560–564 (1982).
25. B. Baddeley, P. Graham, P. Husbands, A. Philippides, A model of ant route navigation driven by scene familiarity. *PLoS Comput. Biol.* **8**, e1002336 (2012).
26. F. Le Möel, A. Wystrach, Opponent processes in visual memories: A model of attraction and repulsion in navigating insects' mushroom bodies. *PLoS Comput. Biol.* **16**, e1007631 (2020).
27. A. Denuelle, M. V. Srinivasan, Bio-inspired visual guidance: From insect homing to UAS navigation, in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2015), pp. 326–332.
28. J. Dupeyroux, J. R. Serres, S. Viollet, AntBot: A six-legged walking robot able to home like desert ants in outdoor environments. *Sci. Robot.* **4**, eaau0307 (2019).
29. D. Lambrinos, H. Kobayashi, R. Pfeifer, M. Maris, T. Labhart, R. Wehner, An autonomous agent navigating with a polarized light compass. *Adapt. Behav.* **6**, 131–161 (1997).
30. M. O. Franz, B. Schölkopf, H. A. Mallot, H. H. Bülthoff, Where did I take that snapshot? Scene-based homing by image matching. *Biol. Cybern.* **79**, 191–202 (1998).
31. W. Stürzl, H. A. Mallot, Efficient visual homing based on Fourier transformed panoramic images. *Rob. Auton. Syst.* **54**, 300–313 (2006).
32. A. Denuelle, M. V. Srinivasan, A sparse snapshot-based navigation strategy for UAS guidance in natural environments, in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2016), pp. 3455–3462.
33. A. Vardy, Long-range visual homing, in *2006 IEEE International Conference on Robotics and Biomimetics* (IEEE, 2006), pp. 220–226.
34. S. Raj, P. R. Giordano, F. Chaumette, Appearance-based indoor navigation by IBVS using mutual information, in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (IEEE, 2016), pp. 1–6.
35. J. Courbon, G. Blanc, Y. Mezouar, P. Martinet, Navigation of a non-holonomic mobile robot with a memory of omnidirectional images, in *2007 ICRA 2007 Workshop: Planning, Perception and Navigation for Intelligent Vehicles* (IEEE, 2007).
36. T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, L. Přeučil, Simple yet stable bearing-only navigation. *J. Field Robot.* **27**, 511–533 (2010).
37. D. Dall'Osto, T. Fischer, M. Milford, Fast and robust bio-inspired teach and repeat navigation, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021), pp. 500–507.
38. A. M. Zhang, L. Kleeman, Robust appearance based visual route following for navigation in large-scale outdoor environments. *Int. J. Robot. Res.* **28**, 331–356 (2009).
39. T. Krajnik, F. Majer, L. Halodova, T. Vintr, Navigation without localisation: Reliable teach and repeat based on the convergence theorem, in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018), pp. 1657–1664.
40. F. Labrosse, Short and long-range visual navigation using warped panoramic images. *Robot. Auton. Syst.* **55**, 675–684 (2007).
41. M. O. Franz, B. Schölkopf, H. A. Mallot, H. H. Bülthoff, Learning view graphs for robot navigation. *Auton. Robots* **5**, 111–125 (1998).
42. F. Chaumette, S. Hutchinson, Visual servo control. I. Basic approaches. *IEEE Robot. Autom. Mag.* **13**, 82–90 (2006).
43. M. Liu, C. Pradalier, Qijun Chen, R. Siegwart, A bearing-only 2D/3D-homing method under a visual servoing framework, in *2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010), pp. 4062–4067.
44. F. Chaumette, S. Hutchinson, P. Corke, "Visual servoing" in *Springer Handbook of Robotics*, B. Siciliano, O. Khatib, Eds. (Springer, 2016), pp. 841–866.
45. D. G. Lowe, Object recognition from local scale-invariant features, in *Proceedings of the Seventh IEEE International Conference on Computer Vision* (IEEE, 1999), vol. 2, pp. 1150–1157.
46. S. Leutenegger, M. Chli, R. Y. Siegwart, BRISK: Binary robust invariant scalable keypoints, in *2011 International Conference on Computer Vision* (IEEE, 2011), pp. 2548–2555.
47. A. Stelzer, M. Vayugundla, E. Mair, M. Suppa, W. Burgard, Towards efficient and scalable visual homing. *Int. J. Rob. Res.* **37**, 225–248 (2018).
48. J. Zeil, N. Boeddeker, W. Stürzl, "Visual homing in insects and robots" in *Flying Insects and Robots* (Springer, 2009), pp. 87–100.
49. R. Möller, A. Vardy, Local visual homing by matched-filter descent in image distances. *Biol. Cybern.* **95**, 413–430 (2006).
50. D. D. Gaffin, B. P. Brayfield, Autonomous visual navigation of an indoor environment using a parsimonious, insect inspired familiarity algorithm. *PLOS ONE* **11**, e0153706 (2016).
51. S. Edvani, E. Va, S. Bettendorf, S. Edvani, E. Va, S. Bettendorf, SIMONA–A reconfigurable and versatile research facility, in *Modeling and Simulation Technologies Conference* (American Institute of Aeronautics and Astronautics, 1997), p. 3809.
52. S. Shah, D. Dey, C. Lovett, A. Kapoor, AirSim: High-fidelity visual and physical simulation for autonomous vehicles, in *Field and Service Robotics: Results of the 11th International Conference*, M. Hutter, R. Siegwart, Eds., vol. 5 of *Springer Proceedings in Advanced Robotics* (Springer, 2018), pp. 621–635.
53. P. Furgale, T. D. Barfoot, Visual teach and repeat for long-range rover autonomy. *J. Field Robot.* **27**, 534–560 (2010).
54. M. A. Khan, F. Labrosse, Visual topological mapping using an appearance-based location selection method, in *Towards Autonomous Robotic Systems: 21st Annual Conference, TAROS 2020, Nottingham, UK, September 16, 2020, Proceedings*, A. Mohammad, X. Dong, M. Russo, Eds., vol. 12228 of *Lecture Notes in Computer Science* (Springer, 2020), pp. 90–102.

55. C. Buehlmann, M. Mangan, P. Graham, Multimodal interactions in insect navigation. *Anim. Cogn.* **23**, 1129–1141 (2020).
56. X. Sun, S. Yue, M. Mangan, A decentralised neural model explaining optimal integration of navigational strategies in insects. *eLife* **9**, e54026 (2020).
57. X. Sun, S. Yue, M. Mangan, How the insect central complex could coordinate multimodal navigation. *eLife* **10**, e73077 (2021).
58. S. Zingg, D. Scaramuzza, S. Weiss, R. Siegwart, MAV navigation through indoor corridors using optical flow, in *2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010), pp. 3361–3368.
59. J. R. Serres, F. Ruffier, Optic flow-based collision-free strategies: From insects to robots. *Arthropod Struct. Dev.* **46**, 703–717 (2017).
60. 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).
61. R. Möller, A. Vardy, S. Kreft, S. Ruwisch, Visual homing in environments with anisotropic landmark distribution. *Auton. Robots* **23**, 231–245 (2007).
62. U. Ramer, An iterative procedure for the polygonal approximation of plane curves. *Comp. Graph. Image Proc.* **1**, 244–256 (1972).
63. D. H. Douglas, T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartogr. Int. J. Geogr. Inf. Geovisualization* **10**, 112–122 (1973).

Acknowledgments: We thank T. Antonsson of Bitcraze AB for letting us use their Crazyflie Brushless prototype drone, which made the experiments possible because of the increased payload capabilities. We thank K. McGuire, P. Campoy, and P. Jonker for guidance and input during the preparatory study, which was performed in the context of T.v.D.'s master's thesis. **Funding:** No external funding was used for this research project. **Author contributions:** Conceptualization: T.v.D. and G.C.H.E.d.C. Methodology: T.v.D. and G.C.H.E.d.C. Software: T.v.D. and C.D.W. Investigation: T.v.D. Resources: T.v.D., C.D.W., and G.C.H.E.d.C. Visualization: T.v.D. Project administration: T.v.D., C.D.W., and G.C.H.E.d.C. Supervision: C.D.W. and G.C.H.E.d.C. Writing—original draft: T.v.D., C.D.W., and G.C.H.E.d.C. Writing—review and editing: T.v.D., C.D.W., and G.C.H.E.d.C. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data of the flight experiments are available within the Supplementary Materials. The comparison between homing algorithms was based on a publicly available dataset (*50*), and the algorithms are documented in their respective articles (*30*, *31*, *49*, *61*).

Submitted 4 August 2023
Accepted 14 June 2024
Published 17 July 2024
10.1126/scirobotics.adk0310