2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Direct learning of home vector direction for insect-inspired robot navigation

Michiel V.M. Firlefyn, Jesse J. Hagenaars and Guido C.H.E. de Croon

***Abstract*— Insects have long been recognized for their ability to navigate and return home using visual cues from their nest's environment. However, the precise mechanism underlying this remarkable homing skill remains a subject of ongoing investigation. Drawing inspiration from the learning flights of honey bees and wasps, we propose a robot navigation method that directly learns the home vector direction from visual percepts during a learning flight in the vicinity of the nest. After learning, the robot will travel away from the nest, come back by means of odometry, and eliminate the resultant drift by inferring the home vector orientation from the currently experienced view. Using a compact convolutional neural network, we demonstrate successful learning in both simulated and real forest environments, as well as successful homing control of a simulated quadrotor. The average errors of the inferred home vectors in general stay well below the 90° required for successful homing, and below 24° if all images contain sufficient texture and illumination. Moreover, we show that the trajectory followed during the initial learning flight has a pronounced impact on the network's performance. A higher density of sample points in proximity to the nest results in a more consistent return. Code and data are available at `https://mavlab.tudelft.nl/learning_to_home`.**

I. INTRODUCTION

Imagine yourself in the middle of a forest. How accurately can you point to the direction of your home without external aids? Flying insects like honey bees and wasps have evolved into experts regarding such high-level navigation tasks, with foraging trips covering distances of up to 13.5 km from their nests [1]. Throughout such trips, they retain a sense of the home direction through path integration [2] based on the Sun's polarization pattern and the optical flow across the retinal image [3]. To ensure consistent returns, any odometric drift accumulated over this journey will have to be corrected for. To do so, honey bees emerging from the nest for the first time perform 'learning' flights around the hive, acquiring visual memories that can be interpreted in later flights in order to successfully navigate back to the hive [4], [5].

Two main models of insect-inspired navigation that could accomplish this have been proposed. In the 'snapshot model' [6] a stored snapshot view is compared with the current view, and the robot moves to minimize their difference [7], [8]. The robot can determine a direction home if the views contain sufficient similar elements for matching, restricting homing success to a limited *catchment area* around the snapshot. For navigating larger distances, snapshots can be linked together in a sequence, allowing for route following. The 'familiarity model' is an alternative model for route following that proposes to move into the most familiar direction when returning to the nest [9]. Typically, a neural network is trained to discriminate between views that have been perceived before and views that are novel. Various methods, from rotating physically to using repulsion and attraction, have been proposed to find the most familiar direction on the way back to the nest [9]–[11]. Route following has two main problems: (i) getting off-route, e.g., by one failure in homing to a snapshot or determining the familiar direction, can lead to a complete navigation loss, and (ii) the outbound route may be very tortuous and hence suboptimal for following as an inbound route.

Instead of route following, many insects return to their nest

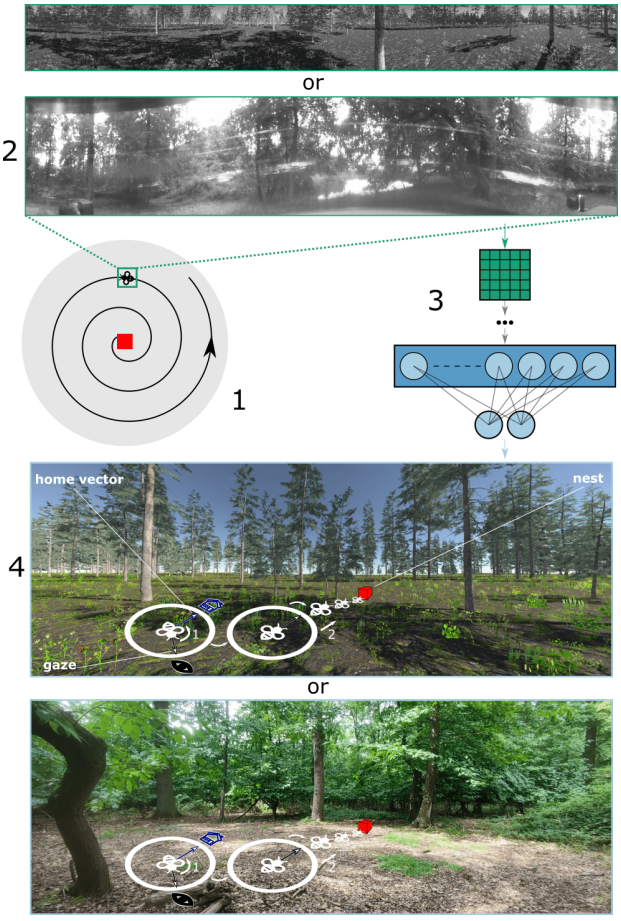

Fig. 1. During a learning trajectory (1), we capture omnidirectional images (2) and train a convolutional neural network (3) to estimate the home vector direction in a simulated or real environment (4).

This publication is part of NWA ACT, which is funded by NWO (NWA.1292.19.298). All authors are with the Micro Air Vehicle Laboratory, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands. Correspondence: `j.j.hagenaars@tudelft.nl`.

in a straight trajectory on a novel path [12], [13]. In [13] it has been argued that insects navigate by means of a nest-centered coordinate frame, always maintaining an internal representation of a vector to the nest. This view is also supported by the fact that honey bees convey this type of information to other bees when communicating about the location of flower fields through waggle dances [14].

Interestingly, honey bees perform 'learning flights' of increasing area when they first leave the hive [15]. After a few learning flights, they are able to navigate to far-away flower fields and return successfully. We hypothesize that honey bees use these learning flights to learn a *direct* mapping from visual percepts to a vector representation of home, consisting of direction and possibly distance. Based on this hypothesis, we introduce a novel insect-inspired navigation algorithm that uses learning flights around the home to learn how to directly map the currently experienced view to the home direction.

This algorithm can be abstractly implemented on a robot, making use of the available insect sensor suite, but not implementing their specific neural architecture. More specifically, we propose to train a compact convolutional neural network (CNN) that takes rectified catadioptric omnidirectional images around the nest as input and returns a unit vector representing the relative home direction. Fig. 1 shows an overview of this. After learning, a robot can travel far outside the range of its learning flights and return based on odometry. Upon return, the trained visual homing network can compensate for any odometric drift as long as this drift falls within the region explored during the learning flights. We demonstrate successful learning in both simulated and real forest environments, as well as successful homing control of a simulated quadrotor. This approach is very promising for navigation in large environments, since the learning flight effectively creates a very large homing catchment area. Depending on the odometry drift, robots can travel much farther than this area's diameter. Moreover, in contrast to previous approaches, our method allows robots to travel back home with odometry on straight, previously unseen paths. Finally, it is a very memory efficient navigation method as the homing knowledge is stored in a relatively small neural network.

II. RELATED WORK

The first known model to consider insect-inspired navigation is the snapshot model by Cartwright and Collett [6]. Based on their own experiments, which involved differing arrays of landmarks, they observed that visual homing in honey bees relies on visual memories acquired at the nest location, with the bees primarily concerned with the apparent size and relative bearing of landmarks. They theorized a framework where the currently perceived horizontal view is compared to the remembered one, resulting in a home vector that is oriented towards the hive.

Lambrinos et al. [7] conducted a robotic experiment of the snapshot hypothesis using a wheeled robot equipped with an omnidirectional camera. Their set-up consisted of black cylinders positioned on a flat 40×40 m desert terrain. Although their tested variants address many limitations of the original snapshot model, they still face challenges when visual noise or distant landmarks are introduced to the testing area [6]. Cues that have a large impact on home vector direction, such as landmark occlusion or mismatches, appear to be the most common obstacles.

Hafner and Möller [8] demonstrated the efficiency and simplicity of learning the home vector by visually guiding a wheeled robot to its target location in a simple simulated environment with black cylinder landmarks against a white background. They employed a snapshot-inspired approach, where a multi-layer perceptron (MLP) was trained to produce a population encoding, representing 8 possible homing directions, based on concatenating the current and home views as inputs to the network. Instead of concatenating multiple views, we propose to learn from only the currently experienced view, in more complex environments.

Baddeley et al. [9] developed a route-following framework that instead learns a familiarity score based on acquired images along a route. By following the images with a relatively high familiarity score, the route can be retraced. While scene familiarity can account for changing landscapes and distant landmarks, it alone cannot provide an accurate indication of the nest direction without a continual stop-scan-go routine. This could be addressed by incorporating a repulsive score, which, when combined with the attractive score, yields an angular error indicating the most familiar viewing direction [10]. Van Dalen et al. [16] presented a validation of scene familiarity using data obtained from a simulated drone, while Gattaux et al. [11] implement familiarity-based control on a small car-like robot. Although the concept of attractive/repulsive scene familiarity appears to resolve previous limitations of homing algorithms, one could question the necessity of inferring two abstract measures from an image and subsequently integrating them into a relative angle that indicates the local home vector bearing. Alternatively, the same outcome could be achieved by directly learning the bearing from the input image.

Consequently, we present the first visual homing algorithm that allows a robot to learn the home directions directly during its learning flight, while only requiring the currently experienced omnidirectional view and odometry to return to its starting location.

III. METHODOLOGY

As shown in Fig. 1, we perform experiments in both simulated and real forest environments, collecting omnidirectional images in regular-grid or spiral patterns around a nest location. Using these images, we train a compact CNN in a supervised manner to estimate the home vector direction relative to the current gaze. In simulation, we investigate the generalization to unseen locations, and use the trained network for homing of a simulated quadrotor. The following subsections explain these parts in more detail.

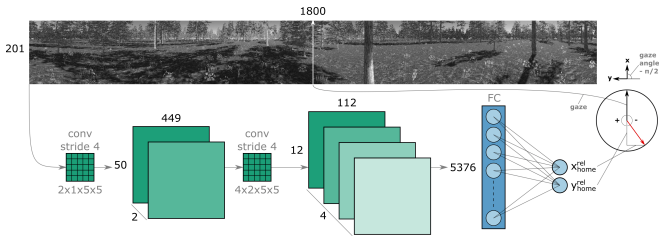

Fig. 2. CNN with kernel and feature map sizes and example input. Convolutional layers are shown in teal; fully connected (FC) layers in blue. The gaze is indicated as a black or white arrow and the home vector as a red arrow. The last nodes have an output value in the range of $[-1, 1]$, encoding the coordinates of the relative home vector. Note that the presented gaze angle $0°$ corresponds to north/upwards.

A. Environment

We use Flightmare [17] and Unity to create a 1000×1000 m photo-realistic forest environment with grass, flowers, different kinds of bushes and trees, and Perlin-noise terrain elevation (Fig. 3). We spawn quadrotors in the desired pattern at a fixed height of 1.89 m above the terrain, and capture six images (front, back, left, right, dorsal, ventral) with a $90°$-FoV camera with a resolution of 1024×1024 pixels. These are then combined into a single image using a catadioptric projection [18] with a latus rectum and distance to the camera image plane of 0.1 m. To get an omnidirectional view, we rectify the image from polar to Cartesian coordinates, setting the gaze angle for this rectification as the angle where the circular image is cut.

The position of the nest is set at $[538, 573]^T$. Images are captured in either a rectangular grid or an Archimedean spiral (representing the honey bee's learning flights). Depending on the experiment, the 10×10 m grid is either spaced at 1 m or 0.25 m intervals. The spiral follows radius $r = 0.25\theta$, with $\theta \in [0, 8\pi]$ in steps of 0.08π.

We assume two-dimensional movement at the fixed height of 1.89 m to simplify rotations and avoid introducing tilt in the acquired images. Furthermore, for labeling and control purposes, we assume a source of heading information is available, such as a magnetic or polarization compass, as has been observed in bees [19], [20]. Lastly, as we focus purely on navigation, we do not consider collisions, and allow simulated quadrotors/cameras to move and spawn in objects, possibly leading to (partial) occlusions of the view.

B. Network architecture and training

We employ the compact three-layer CNN shown in Fig. 2, which consists of two convolutional layers and one fully connected layer. The input image, grayscale with a shape of $1 \times 201 \times 1800$ (channels$\times$height$\times$width), is convolved twice with 5×5 kernels and a stride of 4, doubling the channel count each time. The resulting $4 \times 12 \times 112$ feature map is flattened and passed through a fully connected layer with two outputs, which represent the x- and y-coordinate of the egocentric home vector relative to the subject's gaze on the unit circle. All layers utilize the hyperbolic tangent ($\tanh$) activation function in their respective intermediate outputs. The resulting network consists of 11k parameters (42 kB).

Directly predicting the direction to the nest as a single relative angle did not give good results, likely because similar angles might have very different values (e.g., $-175°$ and $175°$). To resolve this, we decompose the relative angle into its two-dimensional components on the unit circle, as can be seen in Fig. 2. Note that the angle formed by the home vector coordinates is relative to the subject's gaze, i.e., the middle of the currently experienced omnidirectional view.

To train the network, we rectify every omnidirectional image collected during the learning trajectory into 360 distinct gaze directions ($1°$ spacing) and compute the ground truth relative home vectors $\boldsymbol{x}^{\text{rel}}_{\text{home}}$ from geometric relations for all these gazes:

$$\boldsymbol{x}^{\text{rel}}_{\text{home}} = R_z(-\omega_{\text{gaze}})\boldsymbol{x}_{\text{home}} \tag{1}$$

$$R_z(-\omega_{\text{gaze}}) = \begin{bmatrix} \cos(\omega_{\text{gaze}}) & \sin(\omega_{\text{gaze}}) \\ -\sin(\omega_{\text{gaze}}) & \cos(\omega_{\text{gaze}}) \end{bmatrix} \tag{2}$$

where $\boldsymbol{x}_{\text{home}}$ is the home vector relative to north ($[0, 1]^T$) and ω_{gaze} is the gaze angle relative to north. The latter can be computed from the gaze vector $\boldsymbol{x}_{\text{gaze}}$ and north vector using $\arctan2$. For training, we use the mean squared error on the predicted and ground-truth relative home vector as loss function. Furthermore, we use a batch size of 1 and a learning rate of $9\text{e}{-4}$, and train for a single epoch. Note that these settings were chosen as a step towards online learning. All training is done in PyTorch on CPU.[1]

As described, training trajectories are either a 10×10 m grid or an Archimedean spiral. With 100 sample locations and 360 rectified images per location, we get a total of $35,640$ training images (excluding the nest location), which we shuffle before training. For all results in the next section, we evaluate the performance for a single northward gaze angle, as the input to the network has to be rectified, and we cannot show results for all 360 possible gaze angles. Section IV-C is an exception: in that case, we control the quadrotor's gaze/heading based on the network's output.

IV. EXPERIMENTS

A. Learning and generalization in simulation

Training on the 10×10 m grid and Archimedean spiral flight patterns, we get the performance as shown in Fig. 3. The bearing map shows the network's predicted relative home vector $\hat{\boldsymbol{x}}^{\text{rel}}_{\text{home}}$ at every location. If this home vector is not a unit vector, this suggests that the network has not converged to an exact solution, even if the predicted angle is correct. We interpret this as the 'confidence' of the network in its prediction. The prediction error, or angle deviation, is computed as the absolute difference between the predicted and ground-truth relative home direction, $\hat{\omega}$ and ω, which are computed from $\hat{\boldsymbol{x}}^{\text{rel}}_{\text{home}}$ and $\boldsymbol{x}^{\text{rel}}_{\text{home}}$ with $\arctan2$.

Training the network on images captured in a rectangular grid yields a better overall performance compared to the spiral case, with respective average errors[2] of $15.54°$ and

[1] Intel Core i7-8550U running at 1.80 GHz.

[2] We report average errors in degrees instead of the mean squared error in m^2 used during training to allow better interpretation.

TABLE I

AVERAGE ERROR [°] WHEN EVALUATING ON HIGH-RESOLUTION GRID.

	Trained w. label noise	**Trained w.o. label noise**
Trained on grid	16.82 ± 18.98	17.66 ± 18.43
Trained on spiral	23.34 ± 21.77	22.77 ± 21.94

19.72°. However, it is worth noting that, in the case of control, the robot should be able to return to the vicinity of the nest in both scenarios, since the angular errors of all locations are below 45°. Furthermore, looking at the shaded green areas in the bearing maps in Fig. 3, areas near trees that are surrounded by foliage do not show a significant decline in performance. It seems that important landscape features remain interpretable, even when they are partially occluded.

To demonstrate generalization to unseen locations, we capture images in a grid with decreased spacing (0.25 m instead of 1 m), while maintaining the 10×10 m grid size. Fig. 4 shows the performance when evaluating on this higher-resolution grid after training on the lower-resolution grid or the spiral trajectory. Once again, training on the grid exhibits superior performance, as can be seen in Table I. When the home vectors are interpreted as velocity vectors and integrated over the evaluation grid, the stream plots shown in Fig. 4 can be generated. These plots offer insights into the potential trajectory a controlled robot would take. As such, their endpoints are a rendition of the algorithm's catchment area, delineating the boundaries for successful homing. Note that these homing streams converge toward a point that does not precisely coincide with the nest location, as this was excluded in training. Interestingly, in the case of the spiral (Fig. 4B), this convergence point lies closer to the hive in all cases. This difference is a result of the higher training location density near the nest. Moreover, a circular convergence area takes shape instead of a single point. Such an area creates less of an 'artificial' nest location, as can be seen in the singular convergence point of the grid case.

While occlusions due to trees do not seem to affect training performance very much (Fig. 3), they do seem to impact generalization. Looking at the bearing map and error plots in Fig. 4, we see that whenever the network evaluates a particular view occluded by tree foliage (home vectors located in green shading) not encountered during training, the error will be larger. Regions with high error in the plots of Fig. 4 can therefore be attributed to the presence of occluding trees. The same holds for the 'confidence' of the network's prediction, which seems to correlate to some extent with the error and the tree locations.

To examine the robustness of the algorithm, we train another set of networks with labels perturbed by Gaussian noise with $\sigma = 10°$. The results in Table I show that this has a negligible impact on the evaluation performance.

B. Characterizing network activation

Existing insect-inspired visual homing approaches explicitly define which landmarks or features to use. On the other hand, we employ a learning approach without the explicit construction of such features. To get insight into what information the network uses for prediction, we construct a simplified environment consisting of flat terrain and three trees of identical appearance (the 'landmark array'), and evaluate the activations of networks trained in this environment.

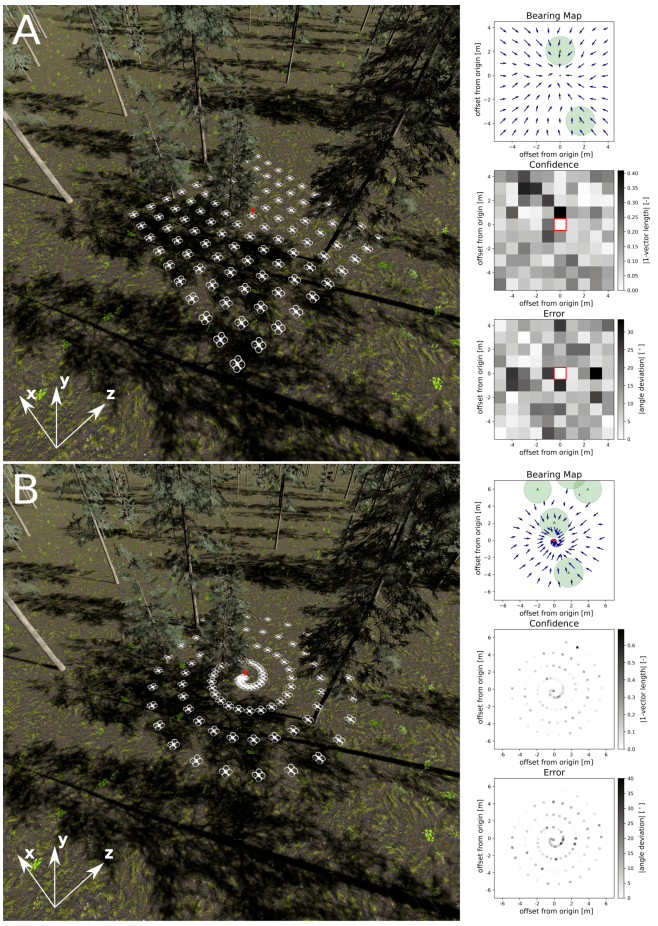

Fig. 3. Simulation environment and training performance for (A) a 10×10 m grid and (B) an Archimedean spiral. The bearing maps show the predicted home vectors for all sampled locations, with the nest location in red and trees shaded in green.

Fig. 5A/B shows the largest-magnitude gradients of the network's output with respect to the activations of the second convolutional layer [21]. Areas in bright yellow and red show that the network mainly looks at a combination of horizon and foliage, as well as some tree shadows. There seems to be little difference in cues for forward/backward (x-coordinate) and left/right (y-coordinate) motion. While using trees as landmarks is not unlike honey bees, it is interesting that the networks trained here seem to consider the contrast of trees against the sky as more useful than the tree stems. This could be a result of the relatively low-resolution grayscale input, lacking diversity in shades of green compared to real-world environments, or the fact that the appearance of the sky is constant in simulation.

Furthermore, we investigate how predictions of a trained network change when the landmark array is rotated. In

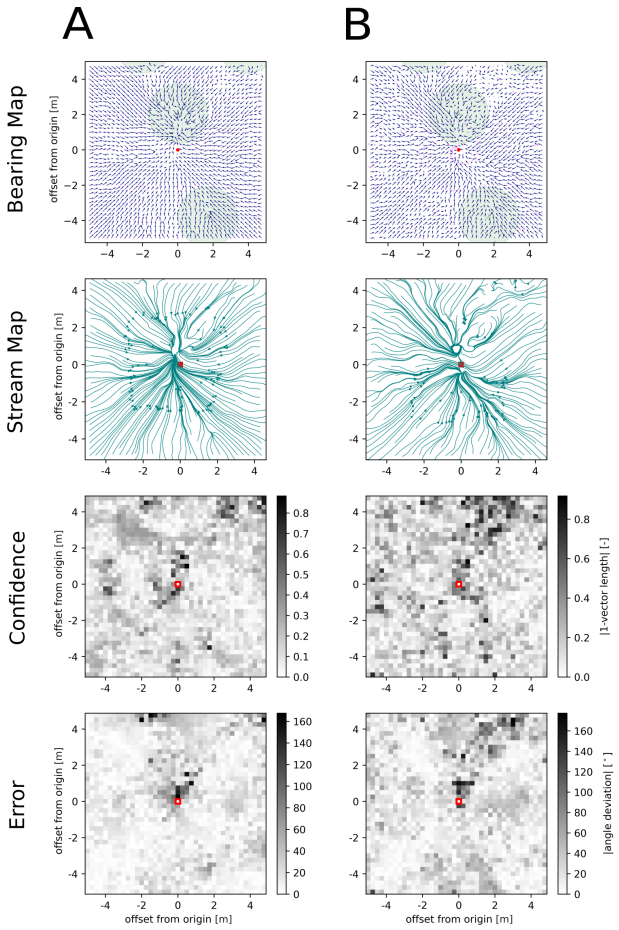

Fig. 4. Generalization of networks trained on (A) a 10×10 m grid and (B) an Archimedean spiral to unseen locations, showing predicted home vectors and resulting stream plots. Locations seen during training are in magenta, the nest is in red and trees are shaded in green.

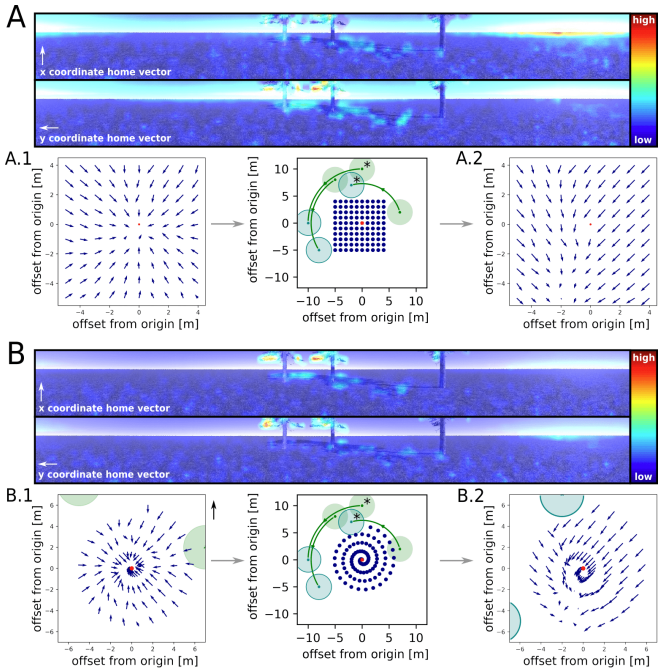

Fig. 5. Analysis (following [21]) of the highest gradients of the network output with respect to the activations of the second convolutional layer when trained on (A) a 10×10 m grid and (B) an Archimedean spiral. Predicted home vectors after training (A.1/B.1) change when evaluated in an environment with a landmark array rotated $90°$ CCW (A.2/B.2). Original landmarks are in green, rotated ones in teal, and dominant landmarks are marked with an asterisk.

Fig. 5, plots A.1/B.1 and A.2/B.2 show predictions before and after rotation, respectively, with an overview of the rotation itself shown in the middle. For both A.2 and B.2, the network predicts a nest location that is shifted downwards by multiple meters. This shift seems to occur due to the network confusing the original center tree and rotated right tree (both marked $*$), with the latter closer to the nest. These results are not unlike those obtained from experiments performed by Collett and Cartwright [6], where honey bees would maintain relative positions when landmark arrays were shifted.

C. Control of a simulated quadrotor

To further demonstrate generalization and usability, we use the output of the network to control a simulated quadrotor. First, we collect training data in either a grid or spiral learning flight and train the network as before. Second, the quadrotor embarks on an outbound flight. Third, we transition to an inbound phase, using path integration to fly in the nest direction. To simulate odometric drift, we stop the inbound flight at the edge of the learning trajectory. Fourth, we transition to visual homing, making use of the output of the network. The insets in Fig. 6 show these steps for a single run per homing starting location, whereas the full figure shows the visual homing part of all runs (transparent trajectories, multiple starting locations). Each run is made with an individually trained network that takes in an omnidirectional view at that location, sets the quadrotor's heading equal to the predicted home direction, and takes a fixed step of 0.25 m in that direction, after which a new omnidirectional view is taken for the next step. If the quadrotor arrives within 0.25 m of the nest location, the run is terminated and counted as a success.

As discussed in Section IV-A, a too low sample density in the vicinity of the nest leads to an offset convergence point, and we see that in Fig. 6A as well. This means that none of the runs successfully reach the nest. On the other hand, when training on a spiral trajectory as in Fig. 6B, 10 out of 12 runs are successful, with on average 33.5 steps taken. So, to ensure consistent returns, the learning trajectory in the vicinity of the nest should be of sufficiently high resolution.

D. Real-world validation

We show the robustness of our approach by validating it on data collected from 10×10 m grids in two real forest environments, shown in Fig. 7. Location A is relatively open and bright, while location B is denser and darker. Omnidirectional images are taken with a Oneplus One smartphone and a Kogeto Dot panoramic lens at a height of ~ 1.44 m above the ground and a spacing of 1 m.

As can be seen, the real-world images contain artifacts not present in their simulated counterparts; lens flare as

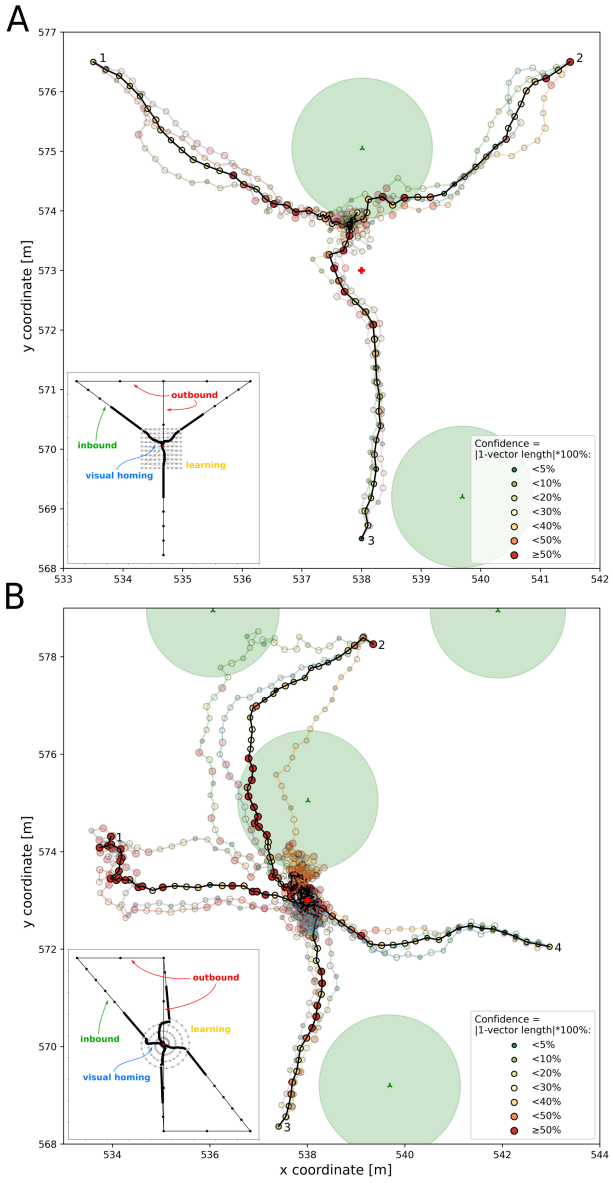

Fig. 6. Homing performance when controlling a simulated quadrotor based on a network trained on (A) a 10×10 m grid and (B) an Archimedean spiral. Individual trajectories are shown transparent, with the mean trajectory per homing starting point in solid. Learning, outbound, inbound and visual homing phases are shown in the inset. Trees are shown in shaded green, and the nest is marked with a red cross at location $[538, 573]^T$.

well as no/inconsistent illumination make the real pictures unclear. Moreover, judging from the wave-like appearance of the images, the employed lens likely does not conform to the assumption of polar-to-Cartesian rectification. Despite all this, it seems that our approach transfers well to real-world data, with the bearing map and training error plot shown in Fig. 7A comparable to the ones in simulation (e.g., Fig. 3A). In the case of location B, performance decreased due to the presence of more vegetation and shadows falling from the dense tree canopy on the forest floor. The respective average errors for locations A and B after training are $15.72°$ and $30.52°$, with the former again comparable to the $15.54°$

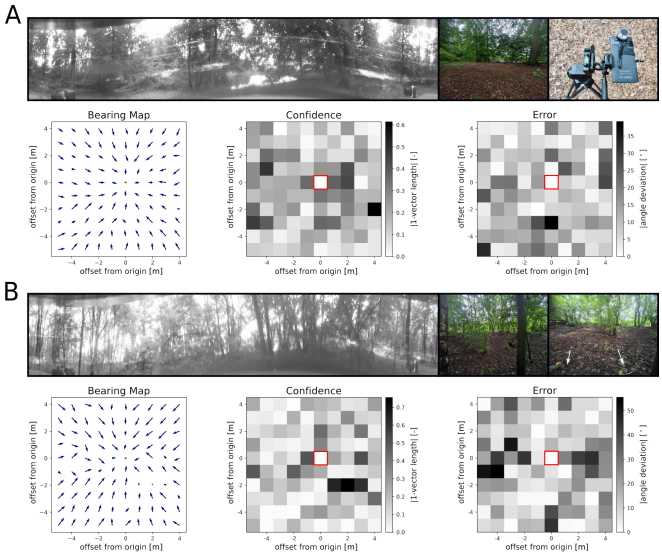

Fig. 7. Training performance in two real forest environments, one (A) open and bright and another (B) covered and dark. The images show grayscale training samples and an overview of the experimental area, along with experimental equipment and sample location markers (white arrows). Nest locations are shown in red.

obtained in simulation.

V. DISCUSSION & CONCLUSION

We proposed a novel insect-inspired visual homing approach that relies exclusively on the currently experienced omnidirectional view. Our main hypothesis posits that bees learn directional cues to their nest during their learning flights, and use these cues to eliminate odometric drift when returning from a long foraging journey. We have subsequently demonstrated successful training of a compact convolutional neural network to predict the home direction in both simulated and real-world forest environments. Furthermore, we have investigated the generalization capability of such networks to locations not seen in training, as well as how they use environmental cues in their predictions. Successful control of a simulated quadrotor shows the promise of using our approach for visual homing on real robots.

This work can be extended in multiple ways, with first and most obvious the implementation of the network on a real robot with simplified 2D velocity control. Second, we observed that not shuffling the obtained images prior to training would lead to catastrophic forgetting. To allow true online during the learning trajectory (and not have to retain all images in memory), this will have to be mitigated [22]. Third, we envision to not only learn direction but also distance during the learning flights. This distance information can for example be used to adapt the flight velocity, allowing to speed up homing without losing homing precision. Lastly, to demonstrate the efficiency of our approach, we should compare it to existing map-building and relocalization techniques for robots like (visual) SLAM [23] and PoseNet [24].

REFERENCES

[1] K. von Frisch, *The Dance Language and Orientation of Bees*. Harvard University Press, 1993.

[2] T. Haferlach, J. Wessnitzer, M. Mangan, and B. Webb, "Evolving a Neural Model of Insect Path Integration," *Adaptive Behavior*, vol. 15, pp. 273–287, 2007.

[3] M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett, "Honeybee Navigation En Route to the Goal: Visual Flight Control and Odometry," *Journal of Experimental Biology*, vol. 199, pp. 237–244, 1996.

[4] E. A. Capaldi, A. D. Smith, J. L. Osborne, S. E. Fahrbach, S. M. Farris, D. R. Reynolds, A. S. Edwards, A. Martin, G. E. Robinson, G. M. Poppy, and J. R. Riley, "Ontogeny of orientation flight in the honeybee revealed by harmonic radar," *Nature*, vol. 403, pp. 537–540, 2000.

[5] J. Zeil, "Visual homing: An insect perspective," *Current Opinion in Neurobiology*, vol. 22, pp. 285–293, 2012.

[6] B. A. Cartwright and T. S. Collett, "Landmark learning in bees," *Journal of comparative physiology*, vol. 151, pp. 521–543, 1983.

[7] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner, "A mobile robot employing insect strategies for navigation," *Robotics and Autonomous Systems*, vol. 30, pp. 39–64, 2000.

[8] V. V. Hafner and R. Möller, "Learning of Visual Navigation Strategies," in *Proceedings of the European Workshop on Learning Robots*, vol. 9, 2001, pp. 47–56.

[9] B. Baddeley, P. Graham, P. Husbands, and A. Philippides, "A Model of Ant Route Navigation Driven by Scene Familiarity," *PLOS Computational Biology*, vol. 8, p. e1002336, 2012.

[10] F. Le Möel and A. Wystrach, "Opponent processes in visual memories: A model of attraction and repulsion in navigating insects' mushroom bodies," *PLOS Computational Biology*, vol. 16, p. e1007631, 2020.

[11] G. Gattaux, R. Vimbert, A. Wystrach, J. R. Serres, and F. Ruffier, "Antcar: Simple Route Following Task with Ants-Inspired Vision and Neural Model," 2023.

[12] R. Wehner and S. Wehner, "Insect navigation: Use of maps or Ariadne's thread?" *Ethology Ecology & Evolution*, vol. 2, pp. 27–48, 1990.

[13] B. Webb, "The internal maps of insects," *Journal of Experimental Biology*, vol. 222, p. jeb188094, 2019.

[14] L. Chittka, *The Mind of a Bee*. Princeton University Press, 2022.

[15] J. Degen, A. Kirbach, L. Reiter, K. Lehmann, P. Norton, M. Storms, M. Koblofsky, S. Winter, P. B. Georgieva, H. Nguyen, H. Chamkhi, U. Greggers, and R. Menzel, "Exploratory behaviour of honeybees during orientation flights," *Animal Behaviour*, vol. 102, pp. 45–57, 2015.

[16] G. J. J. van Dalen, K. N. McGuire, and G. C. H. E. de Croon, "Visual Homing for Micro Aerial Vehicles Using Scene Familiarity," *Unmanned Systems*, vol. 06, pp. 119–130, 2018.

[17] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A Flexible Quadrotor Simulator," in *Proceedings of the 2020 Conference on Robot Learning*. PMLR, 2021, pp. 1147–1157.

[18] B. Berenguel-Baeta, J. Bermudez-Cameo, and J. J. Guerrero, "OmniSCV: An Omnidirectional Synthetic Image Generator for Computer Vision," *Sensors*, vol. 20, p. 2066, 2020.

[19] J. L. Gould, J. L. Kirschvink, and K. S. Deffeyes, "Bees Have Magnetic Remanence," *Science*, vol. 201, pp. 1026–1028, 1978.

[20] S. Rossel and R. Wehner, "Polarization vision in bees," *Nature*, vol. 323, pp. 128–131, 1986.

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, pp. 3521–3526, 2017.

[23] E. Kruzhkov, A. Savinykh, P. Karpyshev, M. Kurenkov, E. Yudin, A. Potapov, and D. Tsetserukou, "MeSLAM: Memory Efficient SLAM based on Neural Fields," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2022, pp. 430–435.

[24] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2938–2946.