

AutoPercep: A Pipeline for Onboard Neighbor Position Estimation Toward Large-Scale Swarm Robotics

Ruiheng Wu^{1,2}, Simay Atasoy Bingöl^{1,2}, Oliver Deussen^{1,2}, Heiko Hamann^{1,2}, Iain D. Couzin^{2,3,4},
Andreagiovanni Reina^{1,2,3}, and Liang Li^{1,2,3,4}

Abstract—Autonomous mobile robots must know each other’s positions to coordinate their actions and motion. Beyond collision avoidance, relative position estimation is essential for spatial coordination tasks such as collective motion, leader–follower dynamics, or formation control. To overcome the scalability and resilience issues of centralized orchestrators that transmit real-time positional information to every robot, we study mechanisms of onboard vision sensing. Conventional localization methods, such as SLAM, are typically too computationally demanding for real-time use on small, resource-constrained mobile robots. Vision-based neural networks offer a promising alternative but often require large, high-quality datasets that are expensive to collect. We present AutoPercep, a pipeline that automatically generates training data and trains a lightweight neural network to estimate neighbor positions. Robots capture camera images that are automatically labeled using ground-truth data from a motion-capture system. In our experiments, AutoPercep collected over 10,000 high-quality images within 10 minutes and trained a neural network in about 1 hour, which could be deployed on Raspberry Pi 4B-based robots for onboard neighbour detection. Moreover, we show that a network trained on five robots generalizes to seven-robot deployments. We finally evaluate the trained model in a sequential leader-follower case study. Our end-to-end pipeline demonstrates the feasibility and low cost of onboard, vision-based neighbor perception, supporting scalability to large robot swarms and opening opportunities for deployment beyond laboratory settings. The code for training and evaluation is available at <https://github.com/preon7/autopercp>

I. INTRODUCTION

Swarm robotics, inspired by the behaviors of biological collectives such as bird flocks and fish schools, has the potential transform multi-robot systems [1]. Reproducing collective animal behavior in robotics can enable efficient large-scale operation while providing robustness, flexibility, and improved coordination in dynamic environments [2]. Systems based on decentralized coordination and adaptive behavior are well suited for applications at scale, such as environmental monitoring, disaster response, and autonomous exploration [3]. Because large-scale experiments are costly and logistically demanding, most studies rely on simulated robots, with only a few exceptions that used simple physical robots such as Kilobots [4].

One of the critical challenges in deploying swarm robotics at scale is enabling onboard sensing and computing. In

particular, achieving collective behavior requires localizing the relative positions of neighboring peers, a capability commonly observed in natural systems [5]. Most existing localization methods rely on external facilities such as Wi-Fi, RFID (Radio Frequency Identification), UWB (Ultra-Wideband), and Zigbee (a low-power wireless communication protocol). While effective for small numbers of robots, these approaches become challenging for large-scale swarms. Onboard sensors, such as distance sensors [6]–[8], ultrasonic sensors [9]–[13], infrared sensors [13]–[16], and LiDAR [13], [17], [18], can measure distances and/or bearing angles, but they struggle with short-range objects, long-range detection, high costs, or strong interference when many robots operate simultaneously. With advancements in computing power and deep learning algorithms, researchers have demonstrated that relative positions of neighbors can be inferred from visual data [19]–[21]. Some Animal species can even extract relative position and movement cues from monocular vision [22], enabling group coordination and collective motion. However, training neural networks usually requires large, high-quality datasets. Most available datasets, such as KITTI [19], MegaDepth [20], and the CMU Vehicle Dataset [21], provide labeled data of high quality but are often limited to specific objects and environments, making them impractical or difficult for custom use cases. Adapting pre-trained models to new scenarios, such as new environments or different robot platforms, is also challenging, as it typically requires additional data collection and fine-tuning. Moreover, several state-of-the-art models, including PoseCNN [23] and DeepIM [24], are highly complex and demand powerful hardware even for inference (after training), making them unsuitable for small onboard processors such as the Raspberry Pi.

In this work, we present AutoPercep, a pipeline for generating large, high-quality, customized training datasets with minimal manual effort to train lightweight neural networks for onboard detection of neighbors’ positions. Using a motion-capture system synchronized with five differential-drive robots equipped with onboard cameras and Raspberry Pi 4B units, we collected over 10,000 images—each labeled with the ground-truth relative positions of up to four neighbours—in under 10 minutes. We trained and compared three lightweight neural networks, ResNet-18, MobileNet-V3-Large, and MobileNet-V3-Small [25], on this autonomously collected dataset and achieved accurate position estimation through vision. ResNet-18 achieved the best performance, running at 15 FPS on a Raspberry Pi 4B for

¹Department of Computer and Information Science, University of Konstanz, Konstanz, Germany

²Centre for the Advanced Study of Collective Behaviour, University of Konstanz, Konstanz, Germany

³Department of Collective Behaviour, Max Planck Institute of Animal Behavior, Konstanz, Germany. lli@ab.mpg.de

⁴Department of Biology, University of Konstanz, Konstanz, Germany

inference alone. Including camera acquisition and image pre-processing, the onboard processing loop runs at 10 Hz, which is sufficient to coordinate our robots moving at up to 20 cm/s. We tested AutoPercep in a sequential following behavior with two and three robots.

II. RELATED WORK

When operating autonomously in large groups, mobile robots inevitably need to know the positions of their neighbors. This ability is essential not only for basic functions such as collision avoidance and motion coordination but also for spatial coordination tasks, including flocking [26], formation control [27], and self-assembly [4]. One solution is to provide each robot with real-time information about its location via an external infrastructure [28], [29]. While such centralized coordination enables highly efficient collective behaviors, it is limited by single points of failure and scalability bottlenecks [30], [31]. In particular, real-time communication among large numbers of robots can encounter bandwidth constraints [32]. An alternative to centralized orchestration is decentralized control, which is the key paradigm of swarm robotics [2]. By sensing the relative positions of their neighbors, robots can rely on onboard computation to coordinate their collective motion [4], [26], [27]. This approach avoids communication bottlenecks, simplifies required infrastructure, and inherently supports scalability.

a) Non-vision-based robot navigation: Robots with limited processing power often use ultrasonic and infrared sensors to navigate. These sensors provide simple information such as light intensity or object distance, offering a low-cost solution for robot navigation. Studies such as [9]–[16] emphasize the design of navigation algorithms based on sensor data that can be efficiently processed by robots.

In contrast, LiDAR sensors [33] offer a much more detailed geometric representation of the environment through point clouds [34]. Compared to ultrasonic and infrared sensors, LiDAR-based systems [13], [17], [18] are typically more expensive and demand more computational resources to process their data.

b) Vision-based robot navigation: Onboard cameras provide a low-cost alternative for robot navigation. As in most biological systems, image data enables simultaneous detection of an object’s presence, position, and orientation, which has led to the increasing adoption of monocular RGB and depth cameras as primary sensory inputs. A common strategy for processing image data is generating a semantic map [35]–[37] or a depth map [20], [38], translating RGB input into depth information to support obstacle avoidance [39]–[41]. Other works reduce information dimensionality, for example, representing the positions of surrounding robots using only angles and size while ignoring overlaps [42], thereby lowering the computational complexity of image processing. Another approach focuses on detecting object positions and bounding boxes [39], enabling more precise control strategies. For instance, MSL-RAPTOR [43] is an image-based pose tracking framework that detects 2D bounding boxes of other drones and updates their spatial positions using an unscented Kalman filter, rather than tracking full 3D positions. However, this method requires the target object to remain continuously visible. Some studies employ stereo cameras to obtain depth information directly, localizing neighboring robots by applying object detection to stereo images [44], [45]. While effective, these methods require greater computational resources and more sophisticated hardware to achieve accurate robot localization.

III. METHODS AND RESULTS

Fig. 1 and the supplementary video provide an overview of our pipeline, which comprises two main steps: automatic collection of a labeled training dataset and deployment of the neural network on low-cost embedded computers. For dataset collection, the system requires a mobile robot equipped with an onboard camera and a global tracking system, such as a motion-capture setup. The tracking system provides high-frequency, accurate ground-truth labels of robot positions. With this setup, large volumes of data can be collected autonomously and labeled automatically through simple post-processing, enabling the training of lightweight neural networks. After training, the network can be deployed on the robot to provide real-time, onboard perception of neighboring robots’ relative positions.

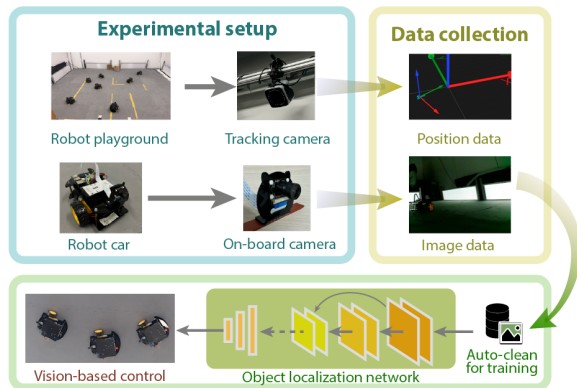


Fig. 1. Overview of our AutoPercep pipeline for data collection and training. We deploy up to five Raspberry Pi 4B-controlled robot cars that move randomly within a dedicated test arena. During the experiment, each robot captures images at 30 FPS with its onboard camera. Simultaneously, each robot’s position and orientation were tracked and recorded by a motion capture system at 100 FPS. Through simple post-processing, we obtain a large dataset of high-quality training data. Finally, we use this dataset to train a deep convolutional neural network to predict the relative positions of visible neighbor robots from onboard camera images. This approach enables vision-based control in swarm-robotics tasks such as flocking or leader–follower behavior.

A. Data collection

For data collection, we used the Osoyoo Pi Robot Car (Pinetree Electronics Ltd.) as the mobile robot, a differential drive robot equipped with a Raspberry Pi for control and an RGB camera for image collection. The differential-drive robot is ~ 25 cm in length/width, with speed 20–35 cm/s and max acceleration 90 cm/s^2 . The onboard camera is a CSI (Camera Serial Interface) web camera with a 70° field

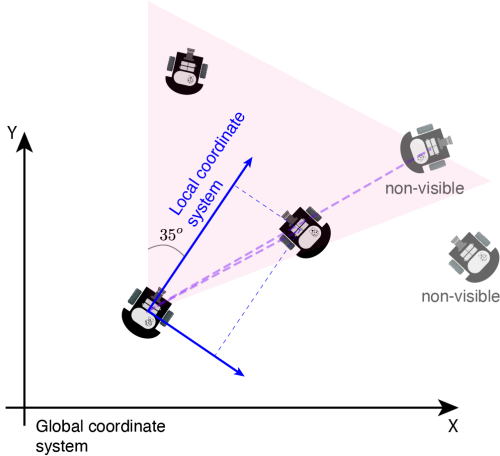


Fig. 2. When generating labeled data automatically from MoCap measurements, neighboring robots are classified as visible or non-visible based on the camera’s field of view and occlusion. An occlusion is detected when two robots lie within the field of view but have an angular separation smaller than the occlusion threshold of 5° (see dashed purple lines). Labels are then sorted by each neighbor’s distance to the focal robot and expressed in the focal robot’s local coordinate frame.

of view, capturing 30 FPS images at a resolution of 640×480 pixels. These onboard images were recorded while the Qualisys motion-capture (MoCap) system simultaneously tracked the three-dimensional coordinates (x, y, z) of infrared-reflective markers on each robot and reconstructed their full six-degree-of-freedom pose $(x, y, z, roll, yaw, pitch)$. The camera frames and MoCap measurements were time-stamped and autonomously synchronized to ensure precise alignment between images captured through onboard cameras and ground-truth positions. Experiments were conducted in a $3 \times 4 \text{ m}^2$ arena covered by six MoCap cameras and surrounded by light panels to compensate for reduced brightness due to short exposure times, thereby reducing motion blur in the onboard images. Five robots moved randomly while a MoCap-based closed-loop controller prevented collisions with other robots and the arena walls.

We applied a lightweight cleaning and labeling pipeline to the raw images and MoCap data. First, we synchronized each onboard image with the temporally closest MoCap record (100 Hz) and retained only pairs with a timestamp mismatch below 5 ms (half the MoCap sampling period). This yielded 11,519 well-aligned image–pose pairs out of 11,872 images (97%) for training. Second, we transformed neighbor positions from the MoCap global frame into each robot’s local coordinate frame [46] using the robot’s tracked position (with a fixed, calibrated offset from the MoCap marker frame to the robot frame) and heading. Third, in the local frame, we classified each neighbour as visible or non-visible based on whether it lay within the camera’s field of view and whether it was occluded. Occlusion was determined by an angular separation threshold of $\theta_{\text{occlusion}} = 5^\circ$: if two neighbors lay within this angle, the farther one was labeled non-visible (Fig. 2). The threshold was chosen based on the typical robot–camera distance ($\sim 1.5 \text{ m}$) and the robot size

(25 cm). Finally, we sorted the remaining visible neighbors by distance to the camera and used the resulting ordered list of relative positions as the training label for each image.

Overall, this process enables efficient collection of a large volume of image data with ground-truth labels, entirely free of manual annotation. By synchronizing the tracked 6D coordinates with the captured images using timestamps, calculating the relative positions of other robots based on each camera’s view direction, and filtering out the occluded robots, we collected a dataset of 10,000 image–position pairs in just 10 minutes. We then used the obtained dataset as input for training an end-to-end object localization network.

B. Model training

Using the collected data, we trained three convolutional network architectures: ResNet-18 (11.6 million parameters) [47], MobileNet-V3-Large (5.5 million parameters), and MobileNet-V3-Small (2.5 million parameters) [25]—to detect neighbors’ relative positions from the onboard camera views. Before training, we performed several preprocessing steps. First, to improve both training efficiency and inference speed, we downscaled the original 640×480 images to two smaller resolutions, 256×256 and 128×128 pixels. Second, to increase robustness under varying lighting conditions, we randomly adjusted image brightness by a factor between 1.2 and 1.5. Finally, we normalized the images using a mean of $[0.485, 0.456, 0.406]$ and a standard deviation of $[0.229, 0.224, 0.225]$, matching the statistics used for pretrained models based on ImageNet [47]. We defined the output tensor with shape $(4, 3)$ because, with five robots in total, each robot can observe at most four neighbors. For each neighbor, the network predicts (x, y, z_{exist}) , where x and y give the relative position and z_{exist} indicates the probability that the neighbor is present.

For training, the total loss is defined as

$$\mathcal{L} = w_{\text{pos}} \mathcal{L}_{\text{pos}} z_j^{\text{appear}} + w_{\text{exist}} \mathcal{L}_{\text{exist}}, \quad (1)$$

where w_{pos} and w_{exist} are scalar weights that balance two components: the *position loss* \mathcal{L}_{pos} and the *existence loss* $\mathcal{L}_{\text{exist}}$. The indicator $z_j^{\text{appear}} \in \{0, 1\}$ ensures that the position loss \mathcal{L}_{pos} contributes only when neighbor j is visible in the image.

The *position loss* \mathcal{L}_{pos} compares the predicted neighbor coordinates (\hat{x}_i, \hat{y}_i) with the ground-truth positions (x_j, y_j) provided by the labeled data converted from the motion-capture system:

$$\mathcal{L}_{\text{pos}} = \text{SL1}_\beta(\hat{x}_i - x_j) + \text{SL1}_\beta(\hat{y}_i - y_j), \quad (2)$$

where $\text{SL1}_\beta(r)$ is the Smooth-L1 (Huber) loss [48] defined as

$$\text{SL1}_\beta(r) = \begin{cases} \frac{r^2}{2\beta} & \text{if } |r| < \beta \\ |r| - \frac{\beta}{2} & \text{otherwise} \end{cases}. \quad (3)$$

This loss is quadratic for small errors ($|r| < \beta$) to encourage fine accuracy and linear for larger errors, thereby reducing the influence of outliers.

The *existence loss* $\mathcal{L}_{\text{exist}}$ evaluates how well the network predicts whether neighbor j is present in the image:

$$\mathcal{L}_{\text{exist}} = \text{BCElogit}(z_i^{\text{exist}}, z_j^{\text{appear}}), \quad (4)$$

where z_i^{exist} is the network’s raw logit output and $z_j^{\text{appear}} \in \{0, 1\}$ is the ground-truth visibility label. The binary cross-entropy with logits [49] is defined as:

$$\text{BCElogit}(z, y) = -\left[y \ln \sigma(z) + (1 - y) \ln (1 - \sigma(z)) \right], \quad (5)$$

with the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \quad (6)$$

The sigmoid $\sigma(z)$ converts the logit into a probability, and the loss penalizes the discrepancy between this probability and the true binary label.

The dataset was split into 90% (10,685 images) for training and 10% (1,187 images) for validation. Training was performed on an NVIDIA 4090 GPU with a batch size of 32, using the AdamW optimiser with a learning rate of 0.0001, for 300 epochs.

To determine w_{pos} and w_{exist} , we performed a two-stage grid search (see Fig. 3). We first trained models for all $(w_{\text{pos}}, w_{\text{exist}})$ combinations with each weight ranging from 0.1 to 4.0 in steps of 0.5, and then refined the search around the best-performing region using a finer step size of 0.2. For each setting, detection performance was measured on the test set based on the interpolated test data, and the final weight pair was selected as the one achieving the highest performance.

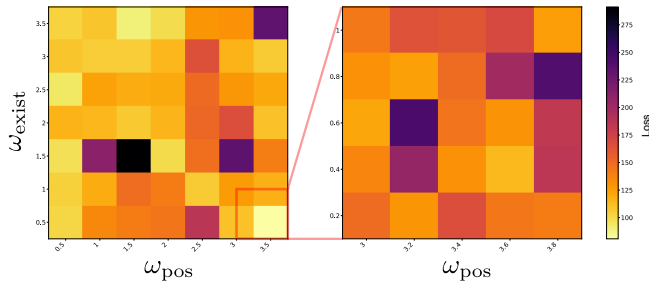


Fig. 3. The test-set loss after training with each $(w_{\text{pos}}, w_{\text{exist}})$ combination.

C. Inference Evaluation

Next, we evaluated the performance of the three convolutional network backends using two test sets: a five-robot set (2,000 images) and a seven-robot set (3,403 images) collected separately. Representative detection results are shown in Fig. 4, demonstrating its generalisability and indicating its potential to scale to larger robot groups. We further conducted a systematic performance evaluation using three metrics: the mean squared error (MSE) of the predicted coordinates (expressed in m^2), the average absolute angular error relative to the camera (expressed in degrees), and the inference time (expressed in ms). A summary of the results is provided in Table II. Among the three networks, ResNet-18

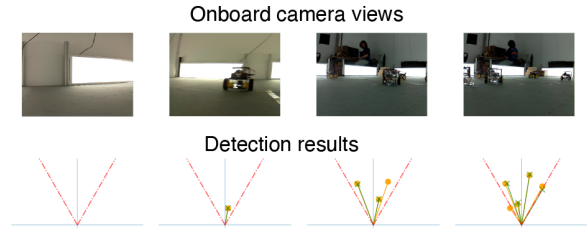


Fig. 4. Four representative examples of neighbor detection from camera images using the neural network Resnet-18 with a group of seven robots. The top row shows the images collected by the robot through its onboard camera. The bottom row shows the detection results (green cross), which are relatively close to the ground-truth positions (yellow circles). The non-visible robots are not shown.

performed best, with an MSE of 0.095 and an angular error of 2.24° . MobileNet-V3-Large followed with an MSE of 0.111 and an angular error of 2.76° , slightly outperforming MobileNet-V3-Small, which achieved an MSE of 0.121 and an angular error of 3.65° .

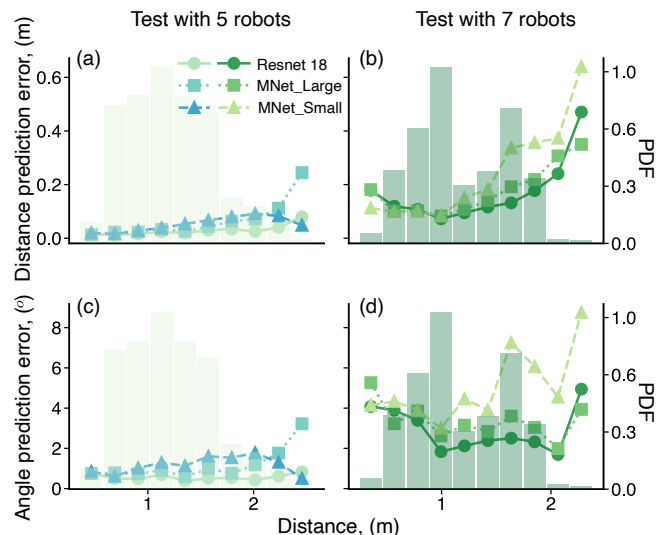


Fig. 5. Distance (L_2) prediction error (a, b) and angle prediction error (c, d) as functions of the distance between neighbor and focal robots in groups of five (a, c) and seven (b, d) robots. The test set was divided into 10 bins, each spanning 0.25 m. The x -axis shows the relative distance between the neighbor and focal robot. The left y -axis shows the average prediction error within each bin, and the right y -axis shows the normalized probability density per bin. All results were evaluated using the network trained on the five-robot dataset only.

In Fig. 5, we show how the evaluation metrics vary with target distance for groups of five and seven robots. The networks performed well on both configurations, although errors were slightly larger in the seven-robot case (up to 50 cm in distance prediction and 8° in angle prediction). We attribute the performance drop in the seven-robot case to a mismatch in the distribution of inter-robot distances between training and evaluation: the five-robot training dataset contains relatively few samples with

neighbors beyond 1.6 m, whereas such distances occur more frequently in the seven-robot configuration (Fig. 5). This limitation could be mitigated by collecting additional training data for robots positioned at greater distances. Overall, these results suggest that the trained network can scale to larger robot groups without retraining, provided it focuses on the four closest neighbors in a large swarm. For both configurations, distance error increased with target distance, while angular error remained small. This trend is likely due to fewer training samples for distant neighbors and the lower pixel resolution at greater distances.

TABLE I
CONFUSION MATRICES FOR THREE MODELS EVALUATED ON 2875 IMAGES

	ResNet-18		MV3-Large		MV3-Small	
	PP	PN	PP	PN	PP	PN
TP	76.5%	0.6%	74.7%	0.5%	73.9%	1.0%
TN	3.2%	19.7%	4.9%	19.9%	5.7%	19.3%

Table abbreviations: TP: True positive; TN: True negative; PP: Predicted positive; PN: Predicted Negative

We also analyzed the accuracy of the predicted existence using confusion matrices (Table I). The results show that all three models achieved similar prediction accuracy, with ResNet-18 providing the highest precision—particularly at larger distances (Fig. 5)—but at a comparatively high computational cost. The low rates of false positives and false negatives indicate that the trained models effectively distinguish between robot presence and absence in the camera view.

To measure the inference time on the Raspberry Pi 4B, we exported each trained PyTorch model to the Open Neural Network Exchange (ONNX) format [50] and executed it with the ONNX Runtime engine (see Table II). Inference time was obtained by averaging the wall-clock duration of 100 consecutive forward passes for two input sizes (256×256 and 128×128 pixels). MobileNet-V3-Small achieved the fastest processing, requiring only 15 ms per 128×128 image, whereas ResNet-18—the most accurate model, required about 61 ms per image. Despite this slower speed, ResNet-18 still delivers position estimates at about 10 FPS, which is sufficient for our real-time control task with robot speeds of about 20 cm/s.

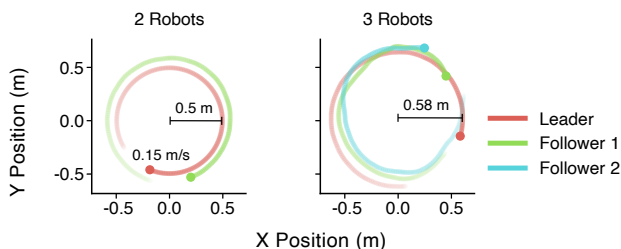


Fig. 6. Trajectories of two- and three-robot groups in the leader-follower experiments using AutoPercep neighbor detection. The leader moved at 0.15 m/s in both cases.

TABLE II
ROBOT CAR LOCALIZATION PERFORMANCE COMPARISON

ONNX (on RPi 4B)	PT 256	PT 128	MSE	angle
ResNet-18	208.4 ms	61.7 ms	0.095 m	2.24°
MobileNet-V3-Large	114.6 ms	36.3 ms	0.111 m	2.76°
MobileNet-V3-Small	45.5 ms	15.7 ms	0.121 m	3.65°
PyTorch (on PC)	-	-	-	-
ResNet-18	0.80 ms	0.84 ms	0.095 m	2.24°
MobileNet-V3-Large	2.11 ms	2.06 ms	0.111 m	2.76°
MobileNet-V3-Small	1.77 ms	1.76 ms	0.121 m	3.65°

Table abbreviations: RPi 4B: Raspberry Pi 4B; PT: prediction time; 256, 128: image size of 256×256 , 128×128 ; MSE: mean squared error; angle: average absolute angle difference; ms: millisecond; m: metre.

D. Leader-Follower Experiments with Two and Three Robots

To evaluate the feasibility of applying AutoPercep to fully decentralized robot control, we conducted a leader-follower case study similar to that described in [5]. In these experiments, a leader robot moved along a circular path. In the two-robot scenario, a single follower robot tracks the leader, whereas in the three-robot scenario, a second follower tracks the first follower, creating a two-step chain.

The robot motion was controlled using a simple proportional (P) controller:

$$v_x = K_p x, \quad (7)$$

$$v_y = K_p y, \quad (8)$$

where x and y are the relative position errors and $K_p = 73$ is the proportional gain tuned empirically via trial and error.

Fig. 6 shows the trajectories of robots moving using onboard detection only, without relying on any external tracking information. These results demonstrate that robots can coordinate their movements using neural networks trained via AutoPercep. In the three-robot case, the second follower correctly identified its nearest neighbor (the first follower) and applied the appropriate control to maintain the leader-follower chain. Overall, these findings indicate that AutoPercep can be readily applied to decentralized control in swarm robotics.

IV. CONCLUSION AND FUTURE STUDIES

In this paper, we presented an automated pipeline for vision-based perception of neighboring robots' relative positions using an onboard camera and a low-cost embedded computer, with the goal of enabling large-scale swarm robotics. This pipeline enables efficient collection of large-scale, high-quality training datasets for lightweight neural network neural networks that can be deployed on low-power embedded computers. To autonomously generate training datasets, we used a motion capture system to accurately label robot positions and synchronize them with onboard camera views. We collected a 10,000-image dataset in 10 minutes, which proved effective for training three different lightweight neural networks to estimate and track the relative positions of neighboring robots from onboard camera images. Testing on a Raspberry Pi showed that the onboard perception loop runs

at a over 10 Hz. Preliminary tests with two- and three-robot groups performing leader-follower tasks using AutoPercep and simple proportional control further demonstrate the proposed pipeline’s potential for applications for swarm robotics applications.

This pipeline provides a simple, practical approach to training-data collection and model training, and it can be readily applied across different environments and robotic platforms, as well as to a wide range of swarm-robotics tasks. For instance, networks trained on data from five robots not only perform well when tested with seven robots but also show potential to scale to even larger swarms. Because the network consistently detects and reports the four closest neighbors with high accuracy, it provides a strong foundation for future applications in large-scale multi-robot systems. In the three-robot leader-follower experiment, the second follower was less accurate than the first. This difference may stem from a mismatch between the training data, collected from robots moving randomly, and the specific following behavior. Accuracy in this task could be improved by collecting more training images from similar scenarios and retraining a network tailored to this behavior, while also refining the occlusion detection method using an adaptive angular threshold. For more general swarm-robotics platforms, one could also replay biological collective-behavior data—such as fish-schooling trajectories—within the robot swarm and collect the corresponding images and labels to train models for biologically inspired swarm behaviors.

Exploiting swarm parallelism and global tracking for efficient data generation, our approach reduces the cost of training data in mobile robotics and provides a foundation for future field deployment. In such scenarios, swarms could autonomously collect diverse, task-specific datasets during real-world missions and self-label them by combining on-board sensors (e.g., GPS/RTK, UWB) to generate training annotations for vision-based models.

ACKNOWLEDGMENT

The authors acknowledge Alexander Bruttel for assembling the robots and funding support from the Max Planck Society, the Office of Naval Research N00014-64019-1-2556 (I.D.C., L.L.), the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement 860949 (I.D.C.), the PathFinder European Innovation Council Work Programme 101098722 (I.D.C.), the Struktur-und Innovationsfonds für die Forschung (SI-BW) of the State of Baden-Württemberg, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy-EXC 2117-422037984 (I.D.C., O.D., A.R.), DFG project number 462886202 (I.D.C.), the Deutsche Forschungsgemeinschaft Gottfried Wilhelm Leibniz Prize 2022 584/22 (I.D.C.), the Sino-German Centre in Beijing for generous funding of the Sino-German mobility grant M-0541 (L.L.), Messmer Foundation Research Award (L.L.). L.L. and R.W. acknowledge Michael Stroh for ideas about image processing

networks and Mohammad Nomaan Husain for helping with the data collection.

REFERENCES

- [1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield *et al.*, “The grand challenges of science robotics,” *Science robotics*, vol. 3, no. 14, p. eaar7650, 2018.
- [2] H. Hamann, *Swarm robotics: A formal approach*. Springer, 2018.
- [3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.
- [4] M. Rubenstein, A. Cornejo, and R. Nagpal, “Programmable self-assembly in a thousand-robot swarm,” *Science*, vol. 345, no. 6198, pp. 795–799, 2014.
- [5] L. Li, M. Nagy, G. Amichay, R. Wu, W. Wang, O. Deussen, D. Rus, and I. D. Couzin, “Reverse engineering the control law for schooling in zebrafish using virtual reality,” *Science Robotics*, vol. 10, no. 101, p. eadq6784, 2025.
- [6] I. Engedy and G. Horváth, “Artificial neural network based local motion planning of a wheeled mobile robot,” in *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, 2010, pp. 213–218.
- [7] D. Hutabarat, M. Rivai, D. Purwanto, and H. Hutomo, “LiDAR-based obstacle avoidance for the autonomous mobile robot,” in *2019 12th International Conference on Information & Communication Technology and System (ICTS)*. IEEE, 2019, pp. 197–202.
- [8] P. T.-T. Nguyen, S.-W. Yan, J.-F. Liao, and C.-H. Kuo, “Autonomous mobile robot navigation in sparse LiDAR feature environments,” *Applied Sciences*, vol. 11, no. 13, 2021.
- [9] J. L. Crowley, “World modeling and position estimation for a mobile robot using ultrasonic ranging,” in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, vol. 89. IEEE, 1989, pp. 674–680.
- [10] R. Li, Z. Du, Y. Zhao, and S. Liu, “Design and implementation of mobile robot ultrasonic localization system,” in *2016 Chinese Control and Decision Conference (CCDC)*. IEEE, 2016, pp. 5347–5352.
- [11] J. Azeta, C. Bolu, D. Hinvi, and A. A. Abioye, “Obstacle detection using ultrasonic sensor for a mobile robot,” *IOP Conference Series: Materials Science and Engineering*, vol. 707, no. 1, p. 012012, 2019.
- [12] L. Moreno, J. M. Armingol, S. Garrido, A. De La Escalera, and M. A. Salichs, “A genetic algorithm for mobile robot localization using ultrasonic sensors,” *Journal of Intelligent and Robotic Systems*, vol. 34, pp. 135–154, 2002.
- [13] G. Csaba, L. Somlyai, and Z. Vámosy, “Mobil robot navigation using 2D LIDAR,” in *2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMII)*. IEEE, 2018, pp. 000 143–000 148.
- [14] E. M. Gorostiza, J. L. Lázaro Galilea, F. J. Meca Meca, D. Salido Monzú, F. Espinosa Zapata, and L. Pallarés Puerto, “Infrared sensor system for mobile-robot positioning in intelligent spaces,” *Sensors*, vol. 11, no. 5, pp. 5416–5438, 2011.
- [15] H. D. Eom and J. W. Jeon, “Environment map building using low-cost IR sensors and a servo motor for mobile robot,” in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*. IEEE, 2014, pp. 1–2.
- [16] G. Benet, F. Blanes, J. Simó, and P. Pérez, “Using infrared sensors for distance measurement in mobile robots,” *Robotics and Autonomous Systems*, vol. 40, no. 4, pp. 255–266, 2002.
- [17] F. B. Malavazi, R. Guyonneau, J.-B. Fasquel, S. Lagrange, and F. Mercier, “LiDAR-only based navigation algorithm for an autonomous agricultural robot,” *Computers and Electronics in Agriculture*, vol. 154, pp. 71–79, 2018.
- [18] S. Jiang, S. Wang, Z. Yi, M. Zhang, and X. Lv, “Autonomous navigation system of greenhouse mobile robot based on 3D LiDAR and 2D LiDAR SLAM,” *Frontiers in Plant Science*, vol. 13, 2022.
- [19] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.
- [20] Z. Li and N. Snavely, “Megadepth: Learning single-view depth prediction from internet photos,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 2041–2050.

- [21] H. Badino, D. Huber, and T. Kanade, "Visual topometric localization," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 794–799.
- [22] D. L. Krongauz, A. Ayali, and G. A. Kaminka, "Vision-based collective motion: A locust-inspired reductionist model," *PLOS Computational Biology*, vol. 20, no. 1, 2024.
- [23] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," *arXiv: 1711.00199*, 2018.
- [24] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6D pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, September 2018.
- [25] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," *arXiv:1905.02244*, 2019.
- [26] T. A. Karagüzel, F. van Diggelen, A. G. Rincon, and E. Ferrante, "Self-organized flocking in three dimensions," in *Swarm Intelligence (ANTS 2024)*, ser. LNCS, vol. 14987. Springer, 2024, pp. 155–167.
- [27] H. G. de Marina, "Maneuvering and robustness issues in undirected displacement-consensus-based formation control," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 3370–3377, 2020.
- [28] S. J. Kim, Y. Jeong, S. Park, K. Ryu, and G. Oh, "A survey of drone use for entertainment and AVR (augmented and virtual reality)," in *Augmented reality and virtual reality: empowering human, place and business*. Springer, 2017, pp. 339–352.
- [29] M. S. Talamali, A. Saha, J. A. R. Marshall, and A. Reina, "When less is more: Robot swarms adapt better to changes with constrained communication," *Science Robotics*, vol. 6, no. 56, p. eabf1416, 2021.
- [30] H. Hamann and A. Reina, "Scalability in computing and robotics," *IEEE Transactions on Computers*, vol. 71, no. 6, pp. 1453–1465, 2022.
- [31] J. Kuckling, R. Luckey, V. Avrutin, A. Vardy, A. Reina, and H. Hamann, "Do we run large-scale multi-robot systems on the edge? More evidence for two-phase performance in system size scaling," in *Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA 2024)*. IEEE, 2024, pp. 4562–4568.
- [32] R. J. Marcotte, X. Wang, D. Mehta, and E. Olson, "Optimizing multi-robot communication under bandwidth constraints," *Autonomous Robots*, vol. 44, no. 1, pp. 43–55, 2020.
- [33] R. T. H. Collis, "LiDAR," *Applied Optics*, vol. 9, no. 8, pp. 1782–1788, Aug 1970.
- [34] T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, and A. Hussain, "A survey on LiDAR scanning mechanisms," *Electronics*, vol. 9, no. 5, 2020.
- [35] T.-V. Dang, D.-M.-C. Tran, and P. X. Tan, "IRDC-Net: Lightweight semantic segmentation network based on monocular camera for mobile robot navigation," *Sensors*, vol. 23, no. 15, 2023.
- [36] T.-V. Dang and N.-T. Bui, "Obstacle avoidance strategy for mobile robot based on monocular camera," *Electronics*, vol. 12, no. 8, 2023.
- [37] S. Jia, K. Wang, and X. Li, "Mobile robot simultaneous localization and mapping based on a monocular camera," *Journal of Robotics*, vol. 2016, no. 1, p. 7630340, 2016.
- [38] K. Yokoyama and K. Morioka, "Autonomous mobile robot with simple navigation system based on deep reinforcement learning and a monocular camera," in *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2020, pp. 525–530.
- [39] F. Wang, C. Zhang, W. Zhang, C. Fang, Y. Xia, Y. Liu, and H. Dong, "Object-based reliable visual navigation for mobile robot," *Sensors*, vol. 22, no. 6, 2022.
- [40] Y. Yang, X. Meng, and M. Gao, "Vision system of mobile robot combining binocular and depth cameras," *Journal of Sensors*, vol. 2017, no. 1, p. 4562934, 2017.
- [41] L. Mu, P. Yao, Y. Zheng, K. Chen, F. Wang, and N. Qi, "Research on SLAM algorithm of mobile robot based on the fusion of 2D LiDAR and depth camera," *IEEE Access*, vol. 8, pp. 157 628–157 642, 2020.
- [42] D. Mezey, R. Bastien, Y. Zheng, N. McKee, D. Stoll, H. Hamann, and P. Romanczuk, "Purely vision-based collective movement of robots," *npj Robotics*, vol. 3, no. 1, p. 11, 2025.
- [43] B. Ramtoula, A. Caccavale, G. Beltrame, and M. Schwager, "MSL-RAPTOR: A 6DoF relative pose tracker for onboard robotic perception," in *Experimental Robotics*, B. Siciliano, C. Laschi, and O. Khatib, Eds. Springer, 2021, pp. 520–532.
- [44] Y. Li, A. Zhao, Y. Wang, Z. Xu, X. Zhou, C. Xu, J. Zhou, and F. Gao, "Fact: Fast and active coordinate initialization for vision-based drone swarms," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 931–938, 2025.
- [45] T. Nguyen, K. Mohta, C. J. Taylor, and V. Kumar, "Vision-based multi-mav localization with anonymous relative measurements using coupled probabilistic data association filter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3349–3355.
- [46] J. E. Gentle, *Matrix algebra: theory, computations, and applications in statistics*. Springer, 2007.
- [47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.
- [48] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 492–518.
- [49] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. Cambridge: MIT Press, 2016, vol. 1, no. 2.
- [50] ONNX Runtime developers, "ONNX runtime," <https://onnxruntime.ai/>, 2021, version: 1.23.2.