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

High-throughput Visual Nano-drone to Nano-drone Relative Localization using Onboard Fully Convolutional Networks

Luca Crupi[1], Alessandro Giusti[1], and Daniele Palossi[12]

***Abstract*— Relative drone-to-drone localization is a fundamental building block for any swarm operations. We address this task in the context of miniaturized nano-drones, i.e., ∼10 cm in diameter, which show an ever-growing interest due to novel use cases enabled by their reduced form factor. The price for their versatility comes with limited onboard resources, i.e., sensors, processing units, and memory, which limits the complexity of the onboard algorithms. A traditional solution to overcome these limitations is represented by lightweight deep learning models directly deployed aboard nano-drones. This work tackles the challenging relative pose estimation between nano-drones using only a gray-scale low-resolution camera and an ultra-low-power System-on-Chip (SoC) hosted onboard. We present a vertically integrated system based on a novel vision-based fully convolutional neural network (FCNN), which runs at 39 Hz within 101 mW onboard a Crazyflie nano-drone extended with the GWT GAP8 SoC. We compare our FCNN against three State-of-the-Art (SoA) systems. Considering the best-performing SoA approach, our model results in a R^2 improvement from 32 to 47% on the horizontal image coordinate and from 18 to 55% on the vertical image coordinate, on a real-world dataset of ∼30 k images. Finally, our in-field tests show a reduction of the average tracking error of 37% compared to a previous SoA work and an endurance performance up to the entire battery lifetime of 4 min.**

SUPPLEMENTARY VIDEO MATERIAL

In-field tests: https://youtu.be/wMFYnv8UE80.

I. INTRODUCTION

Precise drone-to-drone relative pose estimation is a fundamental skill for many swarm operations [1], [2]. Drones capable of precisely localizing peers in the flock can adjust their attitude to maintain desired formations [1] or to optimize their trajectories to maximize their effectiveness [2], e.g., in search-and-rescue or inspection missions. Palm-sized quadrotors, also called nano-drones, represent an uprising class of flying robotic platforms with a weight of less than 50 g and a sub-10 cm diameter [3], [4], [5], see Figure 1-A. Thanks to their small form factor, these miniaturized flying robots enable novel application scenarios in cluttered and narrow indoor environments [6], [7], e.g., industrial plants, collapsed buildings, etc., as well as in human surroundings, being harmless even in case of a crash [8]. Additionally, nano-drones are extremely cheap platforms compared to traditional kg-scale multi-rotors due to their simplified design and electronics. Conversely, their size severely constrains onboard resources, such as sensors, memory capacity, and computational power. For this reason, many autonomous nano-drones run onboard convolutional neural networks (CNNs) for their perception [3], [4], [5] instead of complex geometrical computer vision pipelines [9].

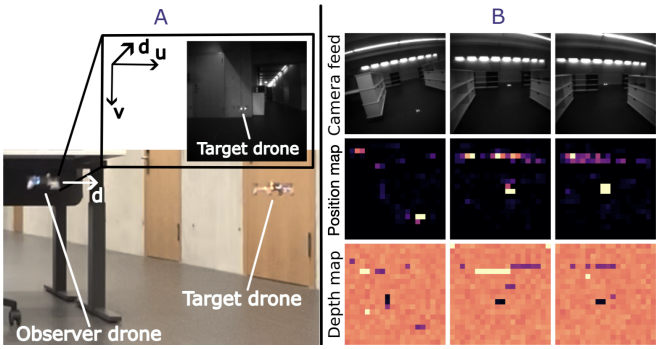

Fig. 1. A) The observer nano-drone tracks a target one. B) Three image samples from the onboard camera associated with the position and the depth map computed by the fully convolutional neural network.

This work addresses the relative drone-to-drone pose estimation with nano-drones relying only on their onboard hardware. Among many successful technologies employed in drone pose estimation, some are prevented aboard nano-drones due to power consumption, weight, and form factor (e.g., LIDAR [10]). In contrast, others require power-hungry radio [11] (up to 100s of mW) and additional ad-hoc infrastructure [12], such as ultra-wideband (UWB) anchors and WiFi routers. For these reasons, many State-of-the-Art (SoA) nano-drone systems [3], [4], [5], including ours, address this problem only by employing cheap vision sensors, e.g., gray-scale low-resolution cameras, and CNNs running aboard nano-drones. Therefore, **our contribution** results in a novel vision-based fully convolutional neural network (FCNN), tailored on the ultra-limited resources aboard a commercially-available Crazyflie nano-drone, extended with the Greenwaves Technologies (GWT) GAP8 multi-core System-on-Chip (SoC). Our FCNN predicts three 20×20 pixel output maps, starting from a gray-scale input image of 160×160 pixels. Two of the three maps are used to solve the relative pose estimation task, i.e., the 2D position of the drone in the u, v image space and the depth map (d). The third map predicts the per-pixel probability of the target drone's LED state (i.e., on/off), which is a strongly correlated task w.r.t. the pose estimation but outside the scope of this work.

This work was partially supported by the Secure Systems Research Center (SSRC) of the UAE Technology Innovation Institute (TII) and the Swiss National Science Foundation (SNSF) through the NCCR Robotics.

[1]L. Crupi, A. Giusti, and D. Palossi are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, Lugano, 6962, Switzerland name.surname@idsia.ch

[2]D. Palossi is also with the Integrated Systems Laboratory (IIS), ETH Zürich, Zürich, 8092, Switzerland

In addition to the FCNN design, we contribute with *i*) full vertical deployment of our FCNN, i.e., from the Python design/training down to the C code execution on the nano-drone's SoC; *ii*) a thorough comparison with three different SoA systems; *iii*) a detailed assessment of our FCNN running onboard the nano-drone, i.e., inference rate and power consumption; *iv*) an in-field evaluation of our closed-loop system regarding endurance and generalization capability. On a ∼30 k images real-world testing dataset, our FCNN outperforms three SoA models [3], [4], [5] designed for the same nano-drone and the same pose estimation task (i.e., output u, v, and d).

On average, on our testing set, over the three outputs, our FCNN achieves an R^2 score of 0.48 while [3] scores 0.3, [4] obtains -0.57, [5] achieves -0.05. In terms of onboard inference rate, on the GAP8 SoC, we achieve a real-time performance of 39 frame/s, while [3] achieves 48 frame/s, [4] achieves ∼5 frame/s, and [5] achieves ∼5 frame/s. Finally, when our system is deployed in the field, it shows remarkable performances: *i*) continuous tracking of the target nano-drone for the entire duration of its battery (∼4 min); *ii*) generalization capabilities with ∼1 min uninterrupted flights in three different *never-seen-before* environments; *iii*) and a reduction of the tracking error of 37%, 52%, and 23% on the x, y, and z coordinates respectively, compared to [3].

II. Related work

Relative pose estimation between drones can leverage various types of sensors. Although, given the strict constraints of nano-drones, i.e., payload, power envelope, and size, not all available technologies are affordable, e.g., GPS [13], LIDAR [10], etc. GPS-based solutions give their best performance outdoors, while in indoor environments, they estimate the position with a 6 m-10 m error [14]. Another technology for 3D localization relies on infrared-based systems. As an example, the solution proposed in [15] comes with the crucial disadvantage of adding more than 120 g onboard and thus is unusable on our nano-drone.

Radio-based solutions employing UWB [16], [17], [11] and WiFi [12] can provide accurate pose estimation (sub-10 cm) and they can be deployed onboard nano-drones [16], [11]. However, they come with two big disadvantages. On the one side, they require ad-hoc infrastructure such as UWB anchors [16], [17], WiFi routers [12], etc., that is not always affordable/deployable. Conversely, UWB-based localization systems need power-hungry devices mounted onboard. In [16], the power consumption for the UWB module aboard the nano-drone peaks at 342 mW, i.e., 5-10× the power envelope for the onboard computation. Vision-based systems, instead, use cameras that are available in lightweight and reduced form factors fitting the payload and size of our nano-drone [3]. Furthermore, they can operate in ultra-low-power budgets, e.g., sub-10 mW without any need for ad-hoc infrastructure, e.g., UWB anchors.

CNNs offer a viable solution for drone pose estimation tasks [3] since they can meet the real-time constraint required for a drone tracking application. In [3], the authors propose

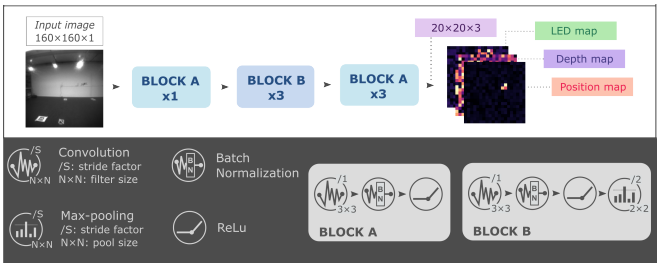

Fig. 2. Our fully convolutional neural network feed with grayscale 160×160 images and producing three 20×20 output maps.

a vision-based neural network that solves position estimation as a regression problem. This network uses as input a 160×96 gray-scale camera frame and produces four scalar variables representing the relative position of the peer drone expressed in x, y, z, and ϕ, i.e., 3D position in space and the relative yaw angle. This approach can only work with one nano-drone as a target since the network estimates only one position, therefore unsuitable for swarm missions. In contrast, our FCNN can tackle multi-drone pose estimation since the output is not bounded a priori to estimate the position of only one drone. Furthermore, the output of the CNN proposed in [3] is produced having as a receptive field the entire image possibly failing to capture local details [18] that are crucial for this application. The nano-drone, in fact, can be as small as 0.06% of the entire image [3].

Works in [4], [5] propose a network based on YOLOv3 [19] that, given an input image produces two maps of 28×40 pixels each. The first map represents the position in the image space, and the second map estimates the distance. These approaches are trained with 900 and 50 k simulated images. The fine-tune is then performed with 192 and ∼8 k real-world images for [4] and [5], respectively. The networks are then tested with 48 and 250 images, respectively, of the same domain used to perform the fine-tuning. The approaches proposed by [4], [5] need 78.7 M multiply-and-accumulate (MAC) operations, which is more than 8.3× the MAC required by our FCNN. Finally, assuming the same efficiency achieved with our FCNN, i.e., 2.2 MAC/cycle, these networks would run on the GAP8 SoC at a maximum of 4.6 frame/s, insufficient for effective tracking. Our work proposes a solution based on FCNN suitable for the deployment on computationally constrained devices such as the GAP8 SoC, still achieving SoA performance in terms of regression performance and throughput, i.e., 39 frame/s.

III. System design

Robotic platform: The nano-drone employed in this work is a 27 g Bitcraze Crazyflie 2.1, featuring an STM32 microcontroller unit that runs the low-level flight controller and the state estimation task. The nano-drone is extended with a 4.4 g AI-deck board, which provides a monocular QVGA grayscale camera (HIMAX HM01B0), a GWT GAP8 SoC, and 8/64 MB off-chip DRAM/Flash. The STM32 and the GAP8 can communicate via a UART bidirectional interface. To increase the drone's stability, we employ a second

extension board called Flow-deck, which provides height measurement with a Time-of-Flight (ToF) sensor and Δx and Δy displacement with a down-looking optical-flow camera.

The GAP8 SoC is designed with two power domains. The former, called fabric controller (FC), has one core in charge of data and heavy computation transfer to the latter, the cluster (CL), which leverages eight parallel cores to perform the execution of computationally intensive kernels. The on-chip memory hierarchy is organized into two levels: a 1-cycle latency 64 kB L1 memory within the CL and a slower 512 kB L2 memory. The SoC provides two direct memory access (DMA) engines that enable efficient transfers between memories and peripherals. The GAP8 lacks hardware floating-point units, forcing the adoption of integer-quantized arithmetic to avoid expensive soft-float computations.

Neural network model: We tackle the drone pose estimation problem with the FCNN architecture depicted in Figure 2. Given a 160×160 grayscale image, it outputs three 20×20 maps: the LED map expresses the state, on or off, of an LED on the target drone, the depth map represents the distance between the observer drone and the target one, while the position map reports the probability of having the target drone in each pixel. From the first map, we extract the (u, v) drone image-space coordinates by calculating the barycenter of the activations. Drone depth is extracted as the weighted average of the depth map, using values of the position map (rescaled such that they sum to 1) as weights. The same approach extracts a scalar value for the LED state (probability that the LED is on) from the LED map.

Datasets: Training and testing datasets[1] have been recorded in a 10×10×2.6 m room equipped with an 18-camera OptiTrack motion capture system (mocap). We recorded 72 flights of ∼210 s, each equally split between the training and the testing sets. Considering an average acquisition rate of 4 frame/s, we record ∼60 k samples, where the testing and the training set count ∼30 k samples each. Every sample comprises a 160×160 pixels camera image, the 3D relative pose between the two drones, and the LED state, i.e., on or off, of the target drone. To train our FCNN, we derive the ground-truth depth map and position map from the recorded 3D pose, and the LED map is obtained from the binary LED state recordings. All the ground truth maps (160x160 pixels) are created starting from a map filled with zeros and placing a circle of radius $r = 4$ pixels centered in the position ground truth that has a decreasing value from the maximum to 0 with a soft edge transition. The maximum value depends on the type of map in fact, the LED map has a maximum value of 1 when the LED is on or 0 when the LED is off, while the position map has a maximum value always equal to 1. Finally, the maximum value of the depth map is equal to the distance between the observer and the target drone.

Deployment: Since the GAP8 SoC does not feature a floating point unit, to avoid the expensive overheads of emulating floating points with integer arithmetic, we deploy our network in the int8 data type using the QuantLib[2] open-source tool. Furthermore, we automatically generate C code for the target processor, the GAP8, using DORY [20], a tool that relies on the PULP-NN kernels [21]. DORY creates the call to the DMA in order to move the tensors from the L2 memory to the L1 memory, which allows the optimized execution of the computations. DORY is not limited to L2-L1 transfers and can exploit the complete hierarchy of memories available on the AI-deck, which includes L1, L2, DRAM, and flash memories. Our network is designed in such a way that all weights, runtime code, and images (double buffered) fit in the L2 memory, i.e., they are under 512 kB; this prevents heavy overheads for DRAM transfer.

IV. Results

A. Regression performance

We evaluate the performance of our model on the testing set by measuring separately for each of the three outputs (u, v, and d): the coefficient of determination metric (R^2) and the Pearson correlation coefficient w.r.t. the ground truth. The former is a standard metric that measures regression performance in a normalized range $[-\infty, 1]$ and reaches 1.0 for a perfect regressor. A regressor that always predicts the average of the testing set scores 0 if measured with the R^2 metric. The latter metric, the Pearson correlation coefficient, captures the linear correlation between predictions and the ground truths and is unaffected by additive and multiplicative biases. Table I reports the performance in terms of R^2 and the Pearson correlation coefficient of our FCNN compared with the following SoA approaches: a) Li et al. [4]; b) Moldagalieva et al. [5]; c) Bonato et al. [3]. Approaches a) and b) are tested in two configurations each: using the pre-trained networks provided by the authors[3][4]; and using the training approach described by the authors, fine-tuned on our testing environment following the procedure defined in the respective papers [4], [5]. In fact, [4], [5] use images from their testing environment to fine-tune their networks. More specifically, approach a) is first trained on the 800 images training set provided by the authors and then fine-tuned using 192 images from our testing environment; for b) we first train the network with the 50 k images provided by the authors, and then we fine-tuned it with more than 8216 from our testing environment.

Approach c) is a regression model, trained from scratch on our training set, since the pre-trained model and the original dataset are not available, that provides outputs as coordinates in 3D space; to compare them to ours, the outputs in the 3D space are projected back in the u, v, and d coordinates according to the camera HIMAX camera matrix. We observe that, on both metrics, our model significantly outperforms all the competing approaches on the u and v variables and performs on par with approach c) on d. It is worth noting that approaches a) and b) are trained and fine-tuned on different (and smaller) datasets than c) and ours.

Figure 3 compares the predictions vs. ground truths for all the outputs (columns) for each model (rows) with a single dot in a plot representing one testing sample. The different clipping in the output position v is due to the various input resolutions of each network ranging between

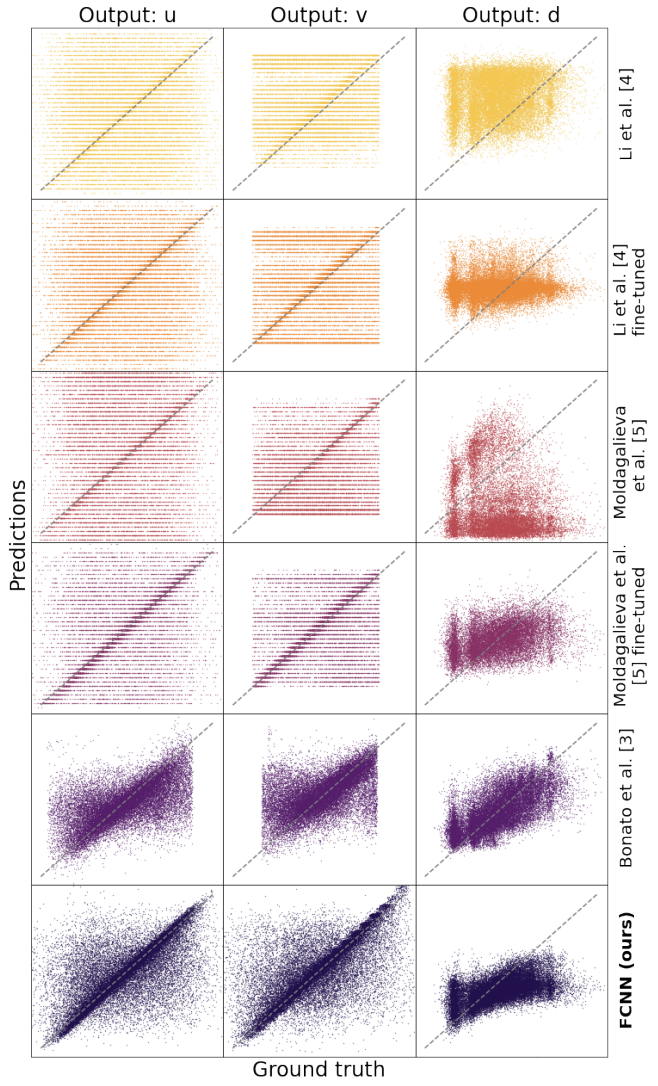

Fig. 3. Predictions vs. ground truths for each model (rows), for different outputs (cols). The dashed lines represent a perfect predictor. Some scatter plots are clipped on the output variable v due to the resolution of the input image ranging from 96 to 160 pixels.

96 and 160 pixels, depending on the model, as described in the previous section. It's worth noting that the clipping on the output v also reduces the ground truth range on the output u variable in the case of Bonato et al. [3]. The scatter plots highlight that our FCNN results in the best-performing network on u and v. In fact, the distributions of the test samples are the closest to the diagonals (dashed lines), with each diagonal representing a perfect predictor. Even if our FCNN produces a position map discretized as 20×20 pixels, u, and v coordinates are extracted as the barycenter of such map, which yields continuous values. This is not the case for Li et al. and Moldagalieva et al., who provide discretized values (horizontal lines) since the prediction is computed by selecting the column and row with the maximum value in the output map, i.e., integer values.

Figure 4 reports the distribution of image-space distances between the predicted (u, v) point and the ground truth.

TABLE I

TEST SET REGRESSION PERFORMANCE.

Network	R2 score [%]			Pearson [%]		
	u	v	d	u	v	d
Li et al. [4]	-88	-68	-174	18	17	16
Li et al. [4] fine-tuned	-75	-66	-29	26	23	2
Moldagalieva et al. [5]	-161	-94	-368	18	24	-16
Moldagalieva et al. [5] fine-tuned	-12	-8	4	50	46	32
Bonato et al. [3]	32	18	42	58	47	**66**
FCNN	**47**	**55**	**42**	**75**	**75**	65

As a lower bound, all graphs report in the background the performance of the dummy predictor baseline (in gray) that always returns the center of the image. The median error (vertical dashed line) of our approach is ∼9 pixels, halving the value achieved by model c); on the other hand, the fine-tuned variant of b) yields a lower median error (7.38 pixels). This is likely since this model relies on an `argmax` operation to obtain the output coordinates from the activation map in the last layer, compared to our barycenter approach. The former is an aggressive approach that provides precise predictions for most samples but yields significant errors for ambiguous cases since it commits to the most likely output; the latter is more conservative, sacrificing accuracy on easy samples to reduce errors on challenging images. The difference between our FCNN and the fine-tuned version of model b) is visible in Figure 3. Furthermore, considering also the computational requirements, a) and b) come at the disadvantage of being 8.3 times more computationally expensive than our method and, as such, are unsuitable for applications with high-throughput requirements, such as the tracking task we describe below.

B. Onboard performance assessment

We evaluate the performance of our FCNN on the GAP8 SoC for what concerns the power consumption and the inference rate in two conditions: *minimum power* (VDD@1.0 V FC@25 MHz CL@25 MHz) and *maximum performance* (VDD@1.2 V FC@250 MHz CL@175 MHz). While the first is useful when the drone acts as a smart sensor, as reported in [3], the latter is crucial when the drone flies at high speed, maximizing the onboard inference rate, which feeds the control loops. In the minimum power configuration, we can process 5.7 frame/s with a power consumption of 10.7 mW. In the maximum performance configuration, we achieved up to 39 frame/s inference rate – requiring almost 4.4 M cycles per frame on the GAP8 SoC – with a power consumption up to 100.8 mW. The total power requirement, including the camera acquisitions, memory transfers, and the GAP8 SoC processing, grows to 109.6 mW. This power envelope accounts only for 1.43% of the total power consumption when also considering the drone's electronics and motors, as shown in Figure 5.

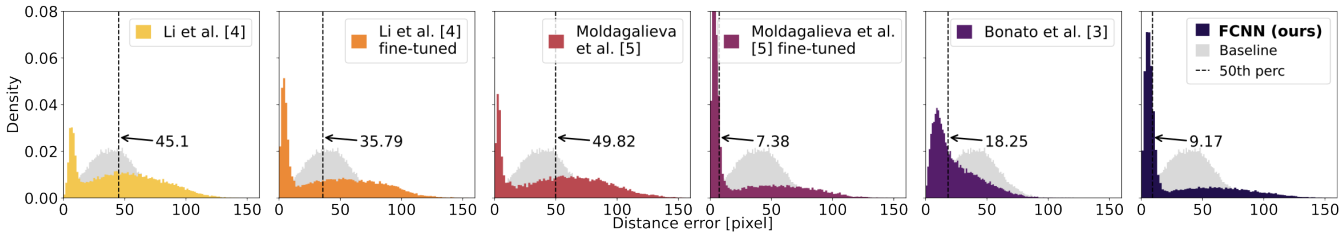

Fig. 4. Distribution of image-space distance error between (u, v) predictions and ground truths. All the networks are trained and tested on our dataset.

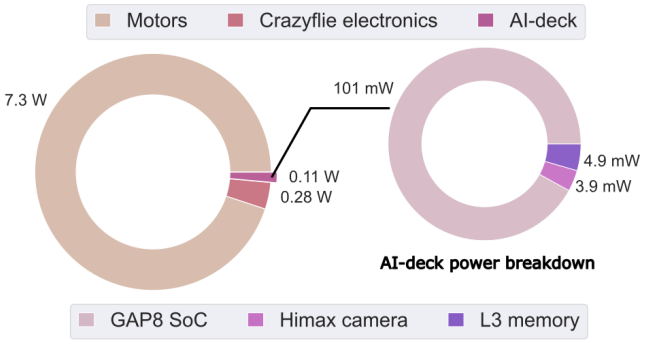

Fig. 5. System power breakdown while running the our approach in the *maximum performance* configuration.

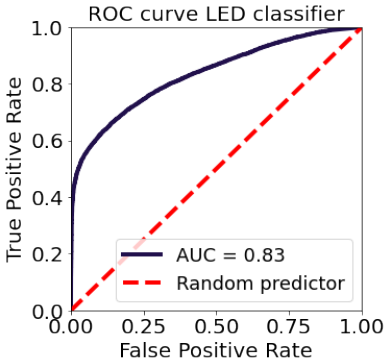

Fig. 6. LED prediction ROC curve for our FCNN.

C. LED state classification

We use the third map produced as output by our network to perform the LED classification task as reported in Section III. We quantify the binary classification performance of the LED state with the Area Under the ROC Curve (AUC) metric. Figure 6 reports the ROC curve for our LED classification task on our testing set, highlighting an AUC score of 0.83. A more detailed analysis shows that when the target drone flies at the same or lower height as the observer, the AUC exceeds 0.90. This indicates excellent classification performance and enables applications with LEDs used as low-bandwidth communication channels. In contrast, when the target drone flies higher than the observer one, the LEDs – placed on the top side of the drone frame – are invisible, and classification becomes impossible in many frames (AUC < 0.70).

D. In-field evalutation

Comparison with the SoA. We perform extensive in-field tests of our FCNN compared with the SoA approach of Bonato et al. [3]. The networks based on [4] and [5] are not tested in-field due to their limited frame rate (i.e., 5.2 frame/s). The setup consists of two drones: a target drone and an observer one. The target drone flies a pre-defined 3D spiral path at a constant speed (∼0.21 m/s), as defined in [3]. The observer tracks and follows the target employing u, v, and d outputs of our FCNN. We assume the observer drone's x axis is aligned with the world's x axis. The desired position of the observer drone is in front of the target one, keeping a constant distance of 0.8 m (x-axis).

Figure 7 reports the in-field performance of the two tested approaches, i.e., Bonato et al. (dashed violet line) and our FCNN (continuous violet line), w.r.t. the desired position (gray line) over a time-frame of ∼60 s. The performance of the drone controlled with our FCNN is remarkably better, if compared to Bonato et al., for the whole duration of the experiment, achieving an average position tracking error comparable to the diameter of our Crazyflie nano-drone, i.e., the error is lower than 10 cm on each axis. Furthermore, as displayed in Figure 7, after the vertical dashed line, we achieved precise tracking of the target drone while landing. The observer drone lands with a position error of ∼15 cm with respect to the desired landing position. The in-field tracking error of this test is reported in II for the two networks with the postfix text "v 0.21". The notation "v 0.21" represents the average speed of the target during the path, namely 0.21 $\frac{\text{m}}{\text{s}}$. In this setting, our approach reduces the average position tracking error by 37%, 52%, and 23%, respectively, on x, y, and z if compared to the approach proposed by Bonato et al. [3]. All the reported metrics have been computed in the time window where both systems were working, i.e., before the vertical dashed line in Figure 7.

Increasing target velocities. The system with our FCNN has been tested five times on the same 3D spiral path described previously, per each configuration of speed, in particular for $v = 0.21\,\frac{\text{m}}{\text{s}}$, $v = 0.34\,\frac{\text{m}}{\text{s}}$, and $v = 0.59\,\frac{\text{m}}{\text{s}}$. In Table II, we report the average tracking error on each world-axis and the aggregate tracking error ($|p - p_d|$) based on the distance between the observer pose (p) and the desired pose (p_d) in the 3D space. In all the experiments, our FCNN running onboard the observer drone successfully tracks the target drone for the complete path. Our observer drone can

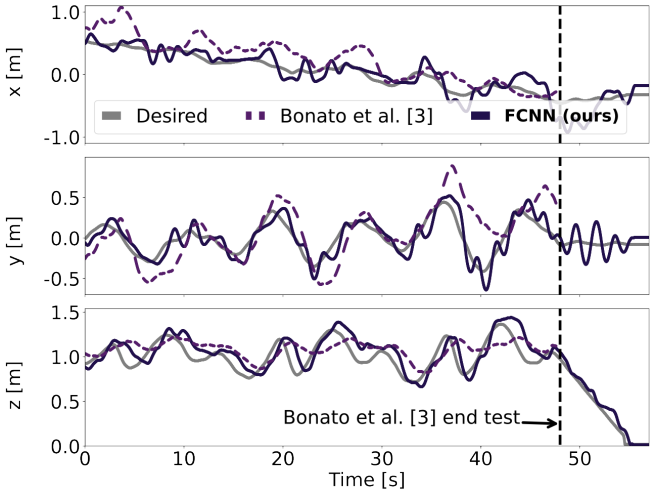

Fig. 7. Trajectory comparison of the observer drone controlled with two approaches vs the desired trajectory.

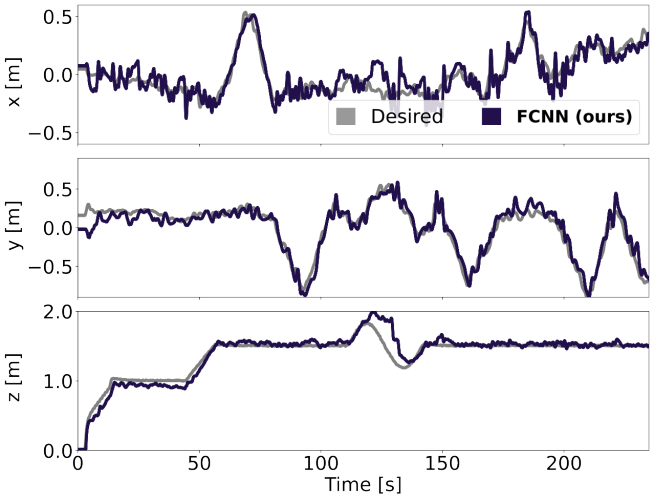

Fig. 8. In-field endurance test of our FCNN model for up to 240 s.

TABLE II

IN-FIELD TRACKING PERFORMANCE (AVERAGE OVER FIVE RUNS) WITH CHANGING AVERAGE SPEED

Configuration	completed runs	**Avg tracking error [m]**				
		x	y	z	$\lvert p-p_d\rvert$	$\sigma\lvert p-p_d\rvert$
Bonato et al. [3] v 0.21	not reported	0.16	0.21	0.13	not rep.	not rep.
FCNN (ours) v 0.21	5/5	0.09	0.10	0.10	0.19	0.08
FCNN (ours) v 0.34	5/5	0.10	0.15	0.14	0.26	0.12
FCNN (ours) v 0.59	5/5	0.12	0.31	0.16	0.41	0.15

Fig. 9. Generalization in three different environments.

track a target drone in front of it with a maximum speed of 0.61 m s, which is 2.8× higher than the speed of the target drone in Bonato et al. [3]. This is possible thanks to the higher throughput (39 Hz) and the improvement in the position estimation of the target drone, as reported in Table I.

Endurance test. In Figure 8, we report an endurance test running for the entire duration of the nano-drone's battery, i.e., 380 mA h, which lasts for 240 s. The target performs linear movements separately for each axis and circles composed of movements on the 3 axes stressing the accuracy of the predictions in the 3 DoF. The observer drone can track the target for the entire experiment duration, achieving on average (5 runs) a tracking error of 0.08 m, 0.07 m, and 0.06 m for x, y, and z, respectively.

Generalization test. Furthermore, our system has been tested in three additional environments different from the one in our training set, with several objects never seen before, such as chairs, sofas, and bookcases. Despite, we can not provide accurate quantitative measurements of the tracking performance due to the lack of a mocap system in our three generalization environments, i.e., coffee corner, office, and Corridor (Figure 9), we provide a video (see supplementary video material) as a qualitative evaluation of our system in these never-seen-before rooms. In this generalization test, the target nano-drone is remotely operated through a controller while the observer drone runs the FCNN.

V. CONCLUSION

This work addresses the drone-to-drone visual localization task by employing resource-constrained nano-drones. We propose a novel lightweight FCNN, i.e., 8× fewer operations than SoA solutions [4], [5], integrated and deployed on a nano-drone extended with a GWT GAP8 SoC. On a 30 k samples real-world testing dataset, our model marks an R^2 score of 0.48 while [3] obtains 0.3, [4] scores -0.57, and [5] achieves -0.05. Our FCNN reaches an inference rate up to 39 Hz within only 101 mW. In-field tests demonstrate on average 37% lower tracking error, compared to [3], which, to the best of our knowledge, is the only SoA approach deployable onboard a nano-drone to perform a pose estimation task of another nano-drone in front of it. Furthermore, with our FCNN, we continuously track a peer nano-drone for the entire nano-drone's battery lifetime, i.e., 4 min. Finally, our FCNN shows remarkable generalization capabilities by continuously tracking a target nano-drone even in three never-seen-before environments.

NOTES

[1]https://github.com/idsia-robotics/drone2drone_dataset
[2]https://github.com/pulp-platform/quantlab
[3]https://github.com/shushuai3/deepMulti-robot
[4]https://tubcloud.tu-berlin.de/s/Sa5rN5JK7poGawrref

REFERENCES

[1] S. Guo, B. Alkouz, B. Shahzaad, A. Lakhdari, and A. Bouguettaya, "Drone formation for efficient swarm energy consumption," in *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. Los Alamitos, CA, USA: IEEE Computer Society, mar 2023, pp. 294–296.

[2] G. A. Cardona and J. M. Calderon, "Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations," *Applied Sciences*, vol. 9, no. 8, p. 1702, 2019.

[3] S. Bonato, S. C. Lambertenghi, E. Cereda, A. Giusti, and D. Palossi, "Ultra-low power deep learning-based monocular relative localization onboard nano-quadrotors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3411–3417.

[4] S. Li, C. De Wagter, and G. C. H. E. De Croon, "Self-supervised monocular multi-robot relative localization with efficient deep neural networks," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9689–9695.

[5] A. Moldagalieva and W. Hönig, "Virtual omnidirectional perception for downwash prediction within a team of nano multirotors flying in close proximity," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2023, pp. 64–70.

[6] K. Wahba and W. Hönig, "Efficient optimization-based cable force allocation for geometric control of multiple quadrotors transporting a payload," *CoRR*, vol. abs/2304.02359, 2023.

[7] J. Burgués, V. Hernández, A. J. Lilienthal, and S. Marco, "Smelling nano aerial vehicle for gas source localization and mapping," *Sensors*, vol. 19, no. 3, 2019.

[8] D. Palossi, N. Zimmerman, A. Burrello, F. Conti, H. Muller, L. M. Gambardella, L. Benini, A. Giusti, and J. Guzzi, "Fully onboard ai-powered human-drone pose estimation on ultralow-power autonomous flying nano-uavs," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1913–1929, 2022.

[9] D. Palossi, F. Tombari, S. Salti, M. Ruggiero, L. Di Stefano, and L. Benini, "Gpu-shot: Parallel optimization for real-time 3d local description," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2013, pp. 584–591.

[10] I. Ouattara, V. Korhonen, and A. Visala, "Lidar-odometry based UAV pose estimation in young forest environment," *IFAC-PapersOnLine*, vol. 55, no. 32, pp. 95–100, 2022, 7th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2022.

[11] V. Niculescu, D. Palossi, M. Magno, and L. Benini, "Energy-efficient, precise uwb-based 3-d localization of sensor nodes with a nano-uav," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5760–5777, 2023.

[12] G. Chi, Z. Yang, J. Xu, C. Wu, J. Zhang, J. Liang, and Y. Liu, "Wi-drone: Wi-fi-based 6-dof tracking for indoor drone flight control," ser. MobiSys '22. New York, NY, USA: Association for Computing Machinery, 2022.

[13] K. N. Tahar and S. Kamarudin, "UAV onboard GPS in positioning determination," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B1, pp. 1037–1042, 06 2016.

[14] Z. Farid, R. Nordin, and M. Ismail, "Recent advances in wireless indoor localization techniques and system," *Journal of Computer Networks and Communications*, vol. 2013, 01 2013.

[15] J. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "3-d relative positioning sensor for indoor collective flying robots," *Autonomous Robots*, vol. 33, 08 2012.

[16] M. Pourjabar, A. AlKatheeri, M. Rusci, A. Barcis, V. Niculescu, E. Ferrante, D. Palossi, and L. Benini, "Land & localize: An infrastructure-free and scalable nano-drones swarm with uwb-based localization," 2023.

[17] M. Strohmeier, T. Walter, J. Rothe, and S. Montenegro, "Ultra-wideband based pose estimation for small unmanned aerial vehicles," *IEEE Access*, vol. 6, pp. 57 526–57 535, 2018.

[18] S. Gao, Z. Li, Q. Han, M. Cheng, and L. Wang, "Rf-next: Efficient receptive field search for convolutional neural networks," *IEEE Transactions on Pattern Analysis; Machine Intelligence*, vol. 45, no. 03, pp. 2984–3002, mar 2023.

[19] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[20] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[21] A. Garofalo, M. Rusci, F. Conti, D. Rossi, and L. Benini, "Pulp-nn: A computing library for quantized neural network inference at the edge on risc-v based parallel ultra low power clusters," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2019, pp. 33–36.