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://github.com/idsia-robotics/FCNN.

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].

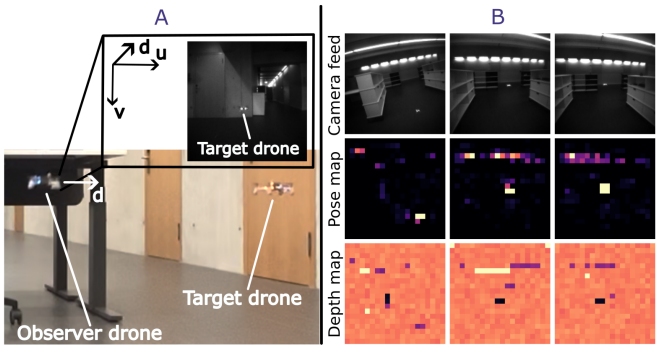

Fig. 1. A) The observer nano-drone performing the tracking of a target nano-drone. B) Three samples recorded infield with the camera feed, the pose, and the depth map computed by the fully convolutional neural network.

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 of them are prevented aboard nano-drones due to power consumption, weight, and form factor (e.g., LIDAR [10]), while others require power-hungry radio (up to 100s mW) [11] 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 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

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, USI-SUPSI, Lugano, 6962, Switzerland name.surname@idsia.ch

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

estimation but outside the scope of this work.

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 fully convolutional neural network (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 ultra wideband (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 the case of [16] 342 mW are necessary to power the UWB module onboard the nano-drone.

Vision-based systems, instead, use cameras that are available in lightweight and reduced form factors fitting the payload and size of our nano-drone. Furthermore, they require reduced power, and they do not need ground infrastructure.

Vision-based systems, instead, do not require any ground infrastructure as well as additional systems mounted onboard drones since cameras are generally part of the standard equipment.

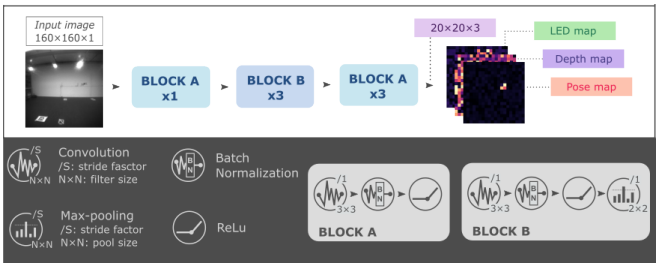

Fig. 2. Architecture of our model.

convolutional neural networks (CNNs) offer a viable solution to meet the real-time constraint and are explored for drone pose estimation tasks [3]. In [3], the authors propose 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 as output four scalar variables representing the relative position of the peer drone expressed in x, y, z, and ϕ, i.e., the relative yaw angle. This approach can only work with one nano-drone as a target since the network estimates only one position. In contrast, our FCNN can tackle multi-drone pose estimation since the output is not bounded apriori to estimate the position of only one drone. Furthermore, the output is produced having as a receptive field the entire image possibly failing to capture local details [18] that are crucial for this application since the nano-drone can be detected in only 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 pose 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 images for [4], [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 Mmultiply-accumulate operations (MACs), which is more than 8.3× the MACs required by our FCNN. Finally, assuming the same efficiency achieved with our FCNN, i.e., 2.2 MACs/cycle, these networks would run on the GAP8 SoC at a maximum of 4.6 frame/s, insufficient for an effective tracking.

III. System design

Robotic platform: The nano-drone employed for this work is a 27 g Bitcraze Crazyflie 2.1, featuring an STM32 MCU 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 gray-scale camera (HIMAX HM01B0), a GWT GAP8 System-on-Chip (SoC) and 8/64 MB off-chip DRAM/Flash memories. The STM32 and the GAP8 can communicate via a UART bidirectional interface. To increase the drone's stability, we

employ a second extension board named 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 transfer and code offloading to the latter, the cluster (CL), which boosts the execution of computationally intensive kernels thanks to its eight parallel cores. The on-chip memory hierarchy is organized into two levels: a fast 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 with memories and peripherals. It lacks 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 first represents drone presence in the corresponding area of the input; the second is its depth; the third is the LED state. 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 presence 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 10x10x2.6 m room equipped with an 18-camera OptiTrack motion capture system. We recorded 72 flights of ∼210 s each that are equally split between the training and the testing sets. Considering an average acquisition speed of 4 fps, we recorded in total ∼60 k samples. Consequently, both the testing and the training sets count ∼30 k samples each. Every sample is composed of a 160×160 pixels camera image, the 3D relative pose between the two drones, and the state, i.e., on or off, of the LEDs onboard the target drone. The 3D pose and the LED state are converted into three maps to allow the training of our FCNN.

Deployment: Our robot platform 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 domain. We convert our floating point network into an int8 network thanks to the QuantLib tool. Furthermore, we automatically generate C code for the target drone using DORY [20], a tool that relies on the PULP-NN kernels [21]. These kernels require all the data to be stored in L1 to be processed. As such, DORY takes care of the memory management to exploit the full memory hierarchy of GAP8 and automatically load tensors in L1 from L2 when needed. DORY is not limited to L1 transfers and can exploit the full hierarchy of memories available on the platform, which includes L1, L2, RAM, 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 RAM memory 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 position 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 for regression performance and reaches 1.0 only for the ideal regressor. The latter captures the linear correlation between predictions and the ground truth and is unaffected by additive and multiplicative bias. Table I reports the performance of our FCNN compared with State-of-the-Art (SoA) approaches by:

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[2][3]; and using the training approach described by the authors, fine-tuned on our testing environment following the procedure and dataset sizes described in the respective papers [4], [5]. More specifically, approach a) was first trained on the 800 images training set provided by the authors and subsequently fine-tuned with 192 images from our testing environment; for b) we first trained 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 that provides outputs as coordinates in 3D space; to compare them to ours, they are projected back in the u, v, and d coordinates.

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 on approach c) on d. It is worth noting that approaches a) and b) are trained and finetuned on different (and smaller) datasets than c) and ours.

Figure 3 compares the predictions vs. ground truths for each output (columns) for each of the six models (rows). Each dot in each plot represents one testing sample. An ideal predictor yields points on the diagonal (dashed line). Note that, even though our FCNN approach provides as output a position map quantized as 20x20 pixels, u, and v coordinates are extracted as the barycenter of such map, which yields continuous values.

Figure 4 reports the distribution of image-space distances between the predicted (u, v) point and the ground truth. As a lower bound, all graphs report in the background the performance of the dummy predictor that always returns the center of the image. The median error 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 yields precise predictions for most samples but yields large errors

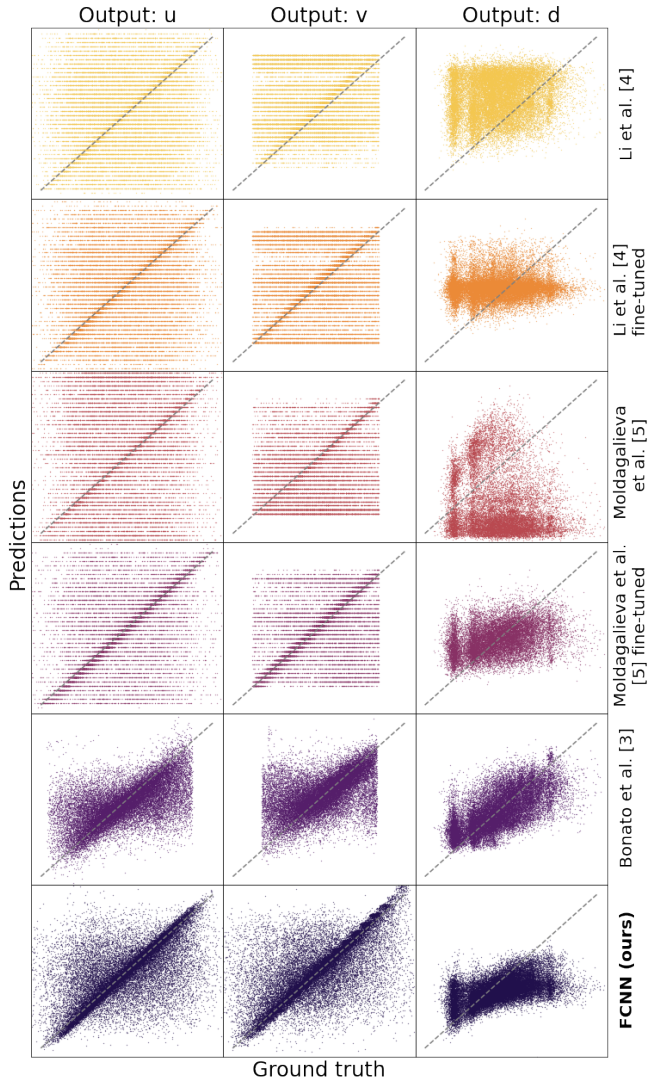

Fig. 3. Predictions vs. ground truths for each model (rows), for different outputs (cols). The dashed lines represent a perfect predictor.

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 the two is clearly visible in Figure 3.

In terms of 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-fps 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 in order to achieve good agile control performance, maximizing the onboard inference rate of the model. In the minimum power configuration, we are able to process 5.7 fps with a power consumption of 10.7 mW. In the maximum performance configuration, we achieved up to 39 Hz inference rate - equivalent to almost 4.4 M cycles per frame on the GAP8 SoC - with a combined power consumption of the fabric controller and the cluster that reaches 100.8 mW. The total power requirement necessary for the full perception task reaches 100.8 mW; it includes the camera acquisitions, the memory transfers, and the GAP8 SoC processing. If considered together with the power required for the motors and the Crazyflie electronics, it accounts only for 1.43% of the entire power budget as reported in figure 5.

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. fine-tuned [4]	-75	-66	-29	26	23	2
Moldagalieva et al. [5]	-161	-94	-368	18	24	-16
Moldagalieva et al. fine-tuned [5]	-12	-8	4	50	46	32
Bonato et al. [3]	32	18	42	58	47	**66**
FCNN	**47**	**55**	**42**	**75**	**75**	65

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 binary classification performance with the Area Under the ROC Curve (AUC) metric: as reported in Figure 6, the approach yields an AUC 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 very good classification performance and enables applications in which LEDs are used as a low-bandwidth communication device. In contrast, when the target drone flies higher, 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 by Bonato et al. [3]. The networks based on [4] and [5] are not tested in-field due to the limited achievable frame rate (5.2 fps).

The setup consists of two drones: one acts as a target and flies a scripted 3D spiral path at a constant speed, as defined in [3]; the second acts as an observer performing a 3D position tracking task of the target drone, freely moving along the three spatial coordinates but keeping a constant yaw. Without loss of generality, we assume that the observer drone's x axis is aligned with the world's x axis. The desired position of the observer drone is such that the target drone is 0.8 m in front of it.

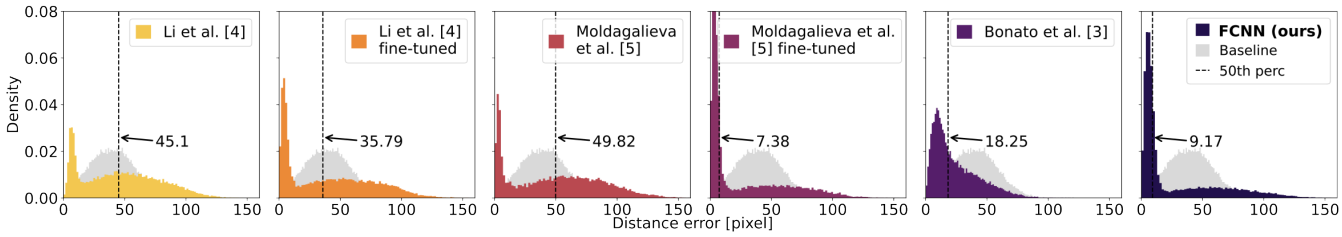

Fig. 4. Distribution of image-space distance error between (u, v) predictions and ground truths.

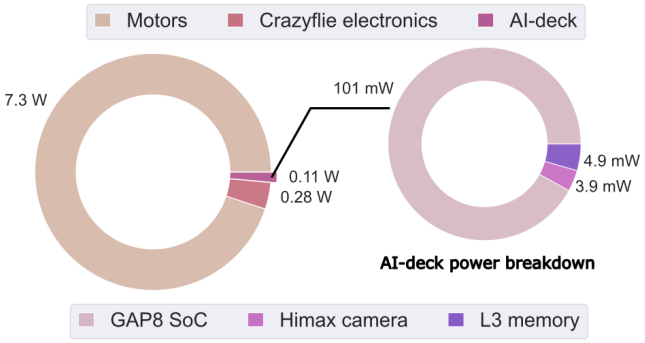

Fig. 5. System power breakdown while running the our approach in the *maximum performance* configuration.

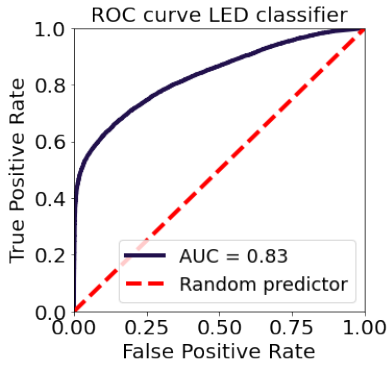

Fig. 6. LED prediction ROC curve for our FCNN

Figure 7 reports the in-field performance of the two tested approaches. The performance of the drone controlled with our approach is remarkably better for the whole duration of the experiment, achieving a position tracking error under 0.1 m on each axis. Furthermore, as displayed in figure 7, after the vertical dashed line, we achieved precise tracking of the target drone also 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. All the reported metrics have been computed in the time window where both systems were working, i.e., before the vertical

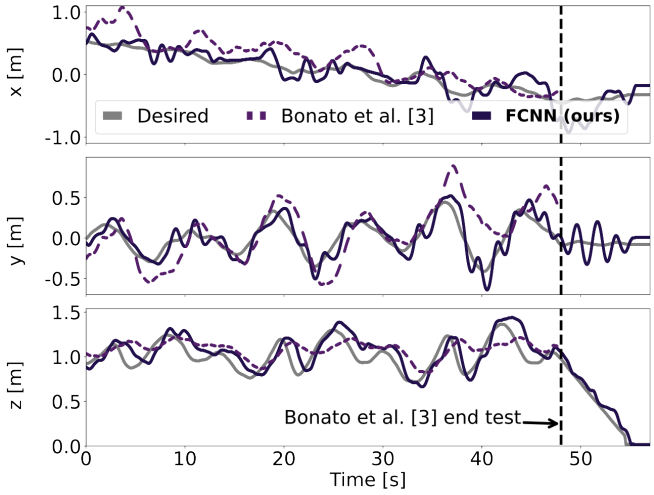

Fig. 7. Trajectory comparison of the observer drone controlled with two approaches vs the desired trajectory.

dashed line in figure 7. With the FCNN approach proposed in this work, we are able to track the target drone with an error below 10 cm on each axis, which is comparable to the diameter of our Crazyflie nano-drone.

Increasing target velocities. The system with our FCNN has been tested five times on the same 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. The maximum average speed at which our system can operate is 2.8× higher than the speed reported in Bonato et al. [3].

Endurance test. In figure 8, we report an endurance test performed on 240 seconds, reaching the maximum time of safe operation with one single battery. For the entire path, our observer drone is capable of tracking the target on all the 3 axes. The target performs movements separately for each axis as well as combined, such as circles stressing the accuracy of the predictions in the 3 Degrees of Freedom (DoF). Considering the average of 5 experiments, we achieved an average tracking error of 0.08 m, 0.07 m, and 0.06 m for x, y, and z respectively.

TABLE II

IN-FIELD TRACKING PERFORMANCE (AVERAGE OVER 5 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

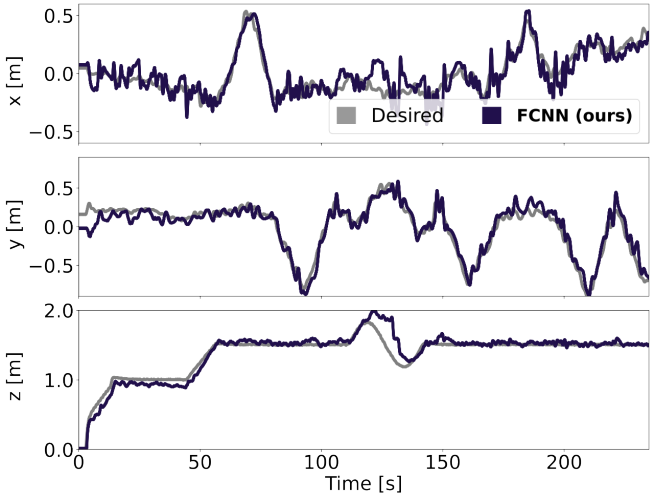

Fig. 8. Infield endurance test of our FCNN model for up to 240s.

Generalization test. Furthermore, our system has been tested in three other environments different from the one in our training set, with several objects never seen in the training set, such as chairs, sofas, and bookcases. Although we are not able to provide accurate quantitative measurements of the tracking performance due to the lack of a motion capture system in these environments, we provide a video to allow a qualitative evaluation of our system in these never-seen-before rooms. The target nano-drone for generalization tests is operated through a controller while the observer drone performs tracking in 3 DoF.

V. CONCLUSION

In this work, we address 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], which we vertically integrate down to the onboard deployment on a Crazyflie 2.1 nano-drone extended with a GWT GAP8 SoC. On our 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. When deployed on the nano-drones our lightweight FCNN reaches a real-time inference rate up to 39 Hz with a minimal power consumption of 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 reaching real-time throughput, i.e., 48 Hz, tracking a peer nano-drone. Furthermore, with our FCNN, we achieve continuous tracking of a peer nano-drone for the entire Crazyflies's battery lifetime, i.e., 4 min. Finally, our FCNN shows remarkable generalization capabilities, by continuously tracking a target nano-drone for more than 1 minute, when deployed in three never-seen-before environments, namely coffee corner, office, and corridor.

Fig. 9. Generalization in three different environments.

NOTES

[1]https://github.com/idsia-robotics/drone2drone_dataset
[2]https://github.com/shushuai3/deepMulti-robot
[3]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. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/PerComWorkshops56833.2023.10150322

[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," 2023.

[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. [Online]. Available: https://doi.org/10.48550/arXiv.2304.02359

[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. [Online]. Available: https://www.mdpi.com/1424-8220/19/3/478

[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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322027549

[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," in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 56–68. [Online]. Available: https://doi.org/10.1145/3498361.3538936

[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. [Online]. Available: https://api.semanticscholar.org/CorpusID:4714433

[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.