2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Ultra-low Power Deep Learning-based Monocular Relative Localization Onboard Nano-quadrotors

S. Bonato[1], S. C. Lambertenghi[1], E. Cereda[1], A. Giusti[1], and D. Palossi[12]

***Abstract*— Precise relative localization is a crucial functional block for swarm robotics. This work presents a novel autonomous end-to-end system that addresses the monocular relative localization, through deep neural networks (DNNs), of two peer nano-drones, i.e., sub-40 g of weight and sub-100 mW processing power. To cope with the ultra-constrained nano-drone platform, we propose a vertically-integrated framework, from the dataset collection to the final in-field deployment, including dataset augmentation, quantization, and system optimizations. Experimental results show that our DNN can precisely localize a 10 cm-size target nano-drone by employing only low-resolution monochrome images, up to ∼2 m distance. On a disjoint testing dataset our model yields a mean R^2 score of 0.42 and a root mean square error of 18 cm, which results in a mean in-field prediction error of 15 cm and in a closed-loop control error of 17 cm, over a ∼60 s-flight test. Ultimately, the proposed system improves the State-of-the-Art by showing long-endurance tracking performance (up to 2 min continuous tracking), generalization capabilities being deployed in a never-seen-before environment, and requiring a minimal power consumption of 95 mW for an onboard real-time inference-rate of 48 Hz.**

SUPPLEMENTARY VIDEO MATERIAL

In-field tests: `https://youtu.be/pUGL1qu3Z1k`.

I. INTRODUCTION

Nano-class size quadrotors are an emerging robot platform with weights below 40 g and a diameter below 10 cm [1], [2]. They unlock attractive novel applications in many scenarios since they are inexpensive and can safely operate in restricted spaces and nearby humans [3]. The potential of this platform is further enhanced when many autonomous nano-drones collaborate in a group (swarm), but achieving this ambitious goal is far from trivial, needing to reliably localize each other while flying, i.e., *relative localization*. This is a significant challenge given the extreme constraints imposed by this class of robots, in terms of sensor types and quality (e.g., QVGA cameras), computational capability (a few 100s MOps/s on single-core microcontroller units), memory size (sub-10 MB off-chip and sub-1 MB on-chip memories) and power consumption (a few 100s mW compute power).

Our work tackles this challenge by designing, implementing and thoroughly validating in the field (closed-loop) a visual relative localization approach that runs fully onboard a 27 g nano-drone up to 48 frame/s within 95 mW at most. Our system, shown in Figure 1, uses input frames from a low-resolution, low-dynamic range tiny camera; it relies on a custom deep convolutional neural network (CNN) with more than 304 k parameters, trained as a regressor to return the relative position of a peer nano-drone. We discuss and motivate design choices spanning the whole system, including *i*) acquisition setup for training and evaluation datasets; *ii*) the selection of the CNN architecture and its training, quantization, and deployment on the target platform; *iii*) analysis of regression performance, throughput, and computational power; *iv*) integration in a control loop that tackles the task of following a peer quadrotor; *v*) validation of the in-field control performance in various environments. Despite being often overlooked with developing systems for larger robots, we highlight how some of these aspects play a fundamental role when targeting nano-quadrotors.

Fig. 1. Our in-field system: nano-drones' relative visual pose estimation.

The main contribution of our system paper is the discussion and extensive quantitative experimental validation of the entire approach, whose video demonstration is released as supplementary material. After reviewing related work in Section II, Section III presents the key aspects of the system (hardware platform, datasets, CNN's architecture/training, and the robot controller). Before concluding the paper in Section V, Section IV outlines experimental results obtained both on static datasets and in-field, showing *i*) a mean R^2 score of 0.42 and a root mean square error (RMSE) of 18 cm on a disjoint testing dataset; *ii*) a mean in-field prediction error of 15 cm which maps in a closed-loop control error of 17 cm, over a ∼60 s-flight tests; *iii*) long-endurance tracking performance, up to 2 min flights; and *iv*) an onboard real-time inference-rate of 48 Hz within 95 mW computational power consumption.

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]S. Bonato, S. C. Lambertenghi, E. Cereda, A. Giusti, and D. Palossi are with the Dalle Molle Institute for Artificial Intelligence, USI and SUPSI, Lugano, 6962, Switzerland `name.surname@idsia.ch`

[2]D. Palossi is also with the Integrated Systems Laboratory, ETH Zürich, Zürich, 8092, Switzerland `dpalossi@iis.ee.ethz.ch`

II. Related work

Estimating the pose of nearby robots is key in many applications, including swarm flocking, formation flying, and obstacle avoidance during navigation, to name a few. Many recent works in the deep learning literature deal with estimating the pose of a known object based on one or more camera images [4], [5], [6], [3], [1]. In robotics, the task is typically solved using the robot's onboard sensors, and estimating the object's position (and sometimes orientation) relative to the robot's reference frame; solving this perception task enables the robot to enact high-level actions like heading towards the object, avoiding it, or grasping it. Our work considers a specific visual object pose estimation case, in which the observing robot and a single target object are a Crazyflie nano-drone.

Object pose estimation subsystems typically rely on data from onboard stereo cameras [7], [8], lidars [9], [10], infrared sensors [11], ultra wideband (UWB) radios [12], [13] or monocular vision [4], [1], [3], being also our case. Despite poor image camera quality, we demonstrate that it is sufficient to localize an observed 10 cm-scale nano-drone, even without visual fiducial markers [4], [14], [15].

Depending on the application, one may need to estimate the full 3D pose of the object, i.e., its position and 3D orientation [8], [4]. In this case, ad-hoc parameterizations can explicitly account for the topology of the SE(3) group of rigid transformations in a machine learning context [16]. In other applications, estimating only the 3D position is sufficient [11], [17], [18], [5], [13]. In this paper, following previous work [3], we address the problem of drone pose estimation using its 3D position.

Visual pose estimation is a challenging pattern recognition problem and poses a significant computational burden, especially given the limited capabilities of our target nano-drone platform. In some cases [19], such computation is offloaded to a more powerful, remote computer that receives sensory data and returns estimated poses. Larger drones [20], [21] are equipped with powerful CPUs (sometimes including GPUs), and high-quality sensors. In contrast, one key aspect of our work is the integration of all computation aboard a memory/computation-limited Crazyflie nano-drone [22], which poses significant challenges due to its microcontroller-class processors (sub-100 mW).

Nano-drones' localization systems often rely on UWB technology, employing either additional ad-hoc infrastructure [23] (UWB fixed anchors) or onboard multilateration algorithms [13]. The former approach has the major disadvantages of minimal flexibility, deployability, and additional cost; the latter introduces a significant onboard power consumption overhead due to the UWB radio communication, up to ∼300 mW in active states [13]. Differently, Li et al. [1] recently tackled a vision-based pose estimation problem very similar to the one presented in this work and with the same hardware (i.e., drone, camera, MCUs). Their contribution focuses on a novel approach to generating UWB-based training labels for a self-supervised CNN. The UWB-based labels are used to train a CNN, which is then validated only offline on a limited 48-image testing set. In contrast, our contribution lies in the system design, integration, and thorough experimental validation: our chosen architecture is larger (8 vs. 5 convolutional layers, 12× more parameters) yet can run 30% faster as it requires fewer operations. We also study real-time performance, provide an offline quantitative evaluation on a 15× larger testing set, and report a quantitative analysis of the in-field performance of the entire system integrated with a controller during uninterrupted flights. Ultimately, our main contribution is application-agnostic, whereas our experimental validation explores a simple formation-flying task.

III. System implementation

A. Robotic platform

In this work, we employ the Bitcraze Crazyflie 2.1 open-source nano-drone, which features as main *flight controller* an STM32 single-core microcontroller unit (MCU). In our configuration, we extend it with a commercial-off-the-shelf (COTS) Flow-deck to improve the onboard state estimation thanks to an optical-flow camera and time-of-flight sensor measuring the distance to the ground. Additionally, we empower our nano-drone with an additional parallel ultra-low power (PULP) MCU, a QVGA monochrome camera, a WiFi radio, and 8/16MB of DRAM/Flash memory, all hosted on the AI-deck COTS companion board. Including all these additions, our nano-drones weights 33 g and has a diameter of 10 cm.

The PULP MCU is the Greenwaves Technologies GAP8[1], a RISC-V-based 9 cores processor. The GAP8 has two power domains: the *fabric controller* (FC), which features one core interacting with external memories/interfaces (e.g., DRAM/Flash and UART communication channel on the AI-deck), and with the second domain, the parallel *cluster* (CL). The CL takes care of intensive computation using its 8-cores. The overall memory hierarchy of this processor is organized into two layers; 64 kB of low-latency L1 memory shared among all cluster cores and 512 kB of L2 memory within the FC domain. However, the GAP8 does not provide data caches or hardware floating-point units (FPUs), dictating explicit data management and the adoption of integer-quantized arithmetic, respectively.

B. Training and testing datasets

Our work relies on labeled datasets acquired ad-hoc; two disjoint datasets are acquired for training and quantitative testing. Each sample in a dataset is composed of: an input image (320×320 gray-scale) and ground truth data for the four output variables, namely x, y, z, and ϕ, i.e., the relative orientation of the target drone w.r.t. the observer one. Ground truths are generated from the absolute poses of the flying drones, recorded with an 18-camera Optitrack infrared motion tracking setup. During data acquisition, the drones are controlled by a ROS node via a predefined sequence of

[1] https://greenwaves-technologies.com/gap8_mcu_ai

waypoints. ROS also handles the precise matching between drones' poses and onboard images streamed via WiFi from the nano-drones.

To collect the training datasets, we leverage this setup to record multiple flights featuring different paths of the two drones, aiming at maximising pose variability for each component of the relative pose, and at recording images with a variety of backgrounds. In particular, we use two motion patterns. In one pattern, the two drones rotate around the center of the room while facing each other; both drones record data simultaneously, thus acting as both an observer and a target. Multiple flights are repeated while modulating: the radius of the circle that the two drones follow; the relative height between the two drones; and the angular offset of the two drones along the circle. In another pattern, the observer drone is static while the target flies along circles lying on a vertical plane, orthogonal to the optical axis of the observer; the target follows multiple of these circles, at different distances from the observer and with different radii.

After data collection, all samples for which the target drone is not in the field of view of the observer are discarded; we obtain a total of 21001 samples, which we randomly divided in 19664 and 1337 samples respectively for training and validation. The images are augmented on-the-fly during the training phase. Random augmentations affect image exposure, gamma, and dynamic range; in addition, noise, blur and/or vignetting are randomly applied to each image.

For the testing dataset, we recorded a new session with a behavior that is not represented in any of the training datasets; this ensures that our quantitative evaluation does not reward any potential overfitting on the training data. In particular, the target drone follows a spiral-like trajectory, which enforces significant variability for all components of the relative pose. The drones are initially aligned on the x axis, 0.2 m apart and facing each other. While the observer drone is static, the target drone slowly flies farther away from the observer along the x axis. While doing so, it alternates inward and outward spiral motions on said axis, as depicted in Figure 2. This procedure generated a total of 754 samples.

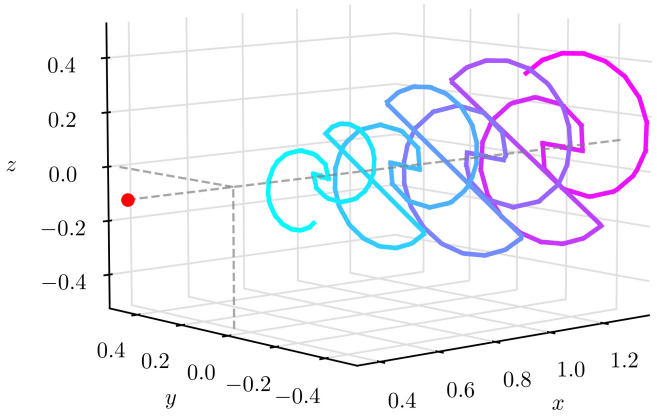

Fig. 2. 3D trajectory flown by the target nano-drone to collect our testing dataset. The observer drone, shown in red, is placed at $(x, y, z) = (0, 0, 0)$ pointing towards positive x

C. Deep neural network

Two CNNs are tested: PULP-Frontnet, a small-sized network proofed for perception tasks and nano-drones, and MobileNetV2, a popular network for edge devices. In the case of Frontnet, the network starts with a 5×5 convolutional layer, followed by a 2×2 max-pooling layer, and then three repetitions of a custom block made of two 3×3 convolutions. The first convolution layer of each block has a stride of 2, further reducing the feature maps size and increasing the number of output channels. All convolutional layers are followed by a batch-normalization layer and a ReLU activation.

In the case of MobileNetV2, we consider 16 variants that are lighter than the original architecture in terms of both memory and computational needs. These networks consist of a standard convolutional layer, four inverted residual blocks, an average pooling layer and a fully connected layer. Figure 3 shows the template of our reduced MobileNetV2. Our search space is parametrized by t and n, which refer to the expansion factor and the number of repetitions of the two intermediate blocks, respectively. We consider the 15 models defined as $(t, n) \in \{6, 8, 10, 12, 14\} \times \{2, 3, 4\}$, where $\times$ denotes the cartesian product; we further include the case $(t, n) = (2, 2)$, i.e. the smallest possible variant considering our stride layout constraints.

D. System deployment

Once the models are trained in PyTorch, we need to change their numerical representation from a 32-bit floating point representation (i.e., full precision) to an 8-bit integer fixed-point arithmetics due to the missing FPU in our target processor (Section III-A). We use the NEMO [24] library, which uses parametrized clipping activation technique [25], to quantize the CNN's weights and activations in three sequential steps. The first one trains a floating-point CNN to minimize the sum of the L1 loss for the pose estimation vector (x, y, z, ϕ). Then, we convert the model with the lower validation loss to the so-called *fake-quantized* version, where weights and activations magnitude can assume only a discrete (0-255) set of values, still using float32 data-types. Before moving to the last stage, we perform a fine-tuning PACT-based quantization stage over ten epochs. We conclude the process with the *integer deployable* stage, where all tensors are represented by integer numbers and a floating point scale factor. At this point, the network can be executed entirely by multiplying and accumulating only integer values.

The second tool employed is called DORY [26], which, starting from the quantized model, produces optimized C code. By combining optimized basic kernels (e.g., convolutions, linear layers, max-pooling layers, etc.) in a template-based wrapping code, DORY also takes care of the memory orchestration to exploit data-locality in the GAP8's L2 and L1 memories and off-chip DRAM.

To complete the development of our pipeline, we extend the inference routine generated by DORY, with a *i*) image acquisition routine, *ii*) image cropping (from 160×160 to 160×96pixels), and *iii*) the final UART communication to

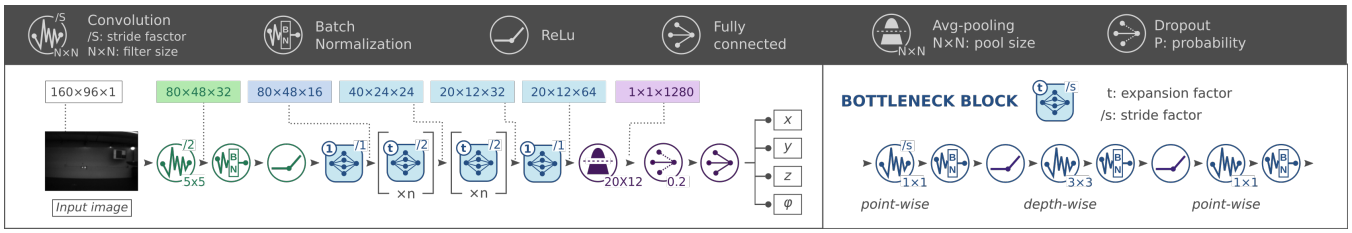

Fig. 3. Template of the network structure used for our lightweight MobileNetV2 variants.

forward the CNN's output to the main STM32 MCU aboard the nano-drone. Finally, our pipeline overlaps the inference time (the longest of all routines) with image acquisition and cropping (together ~ 0.6 Mcycles) in a double-buffered fashion. With this mechanism, the FC can start acquiring and cropping a new image when the CL is computing the previous one, limiting the overall pipeline's latency only to the inference routine.

Once a new prediction (x, y, z, and ϕ) is sent from the GAP8 to the STM32, the pose is filtered by a Kalman filter, smoothing the sequence of poses. The output of the filter is fed to a velocity controller, as in [3], that converts this temporally-consistent series of poses into velocity setpoints to keep the target nano-drone at a constant distance of 0.8 m, and then forwarded to the Crazyflie PID-cascade controller.

IV. Results

A. Comparison of network architectures

In this section, we compare the PULP-Frontnet model with 16 variants of the MobileNetV2 CNN. Figure 4 reports three metrics for each architecture: the memory footprint (i.e., number of 8-bit weights); the number of multiply-accumulate (MAC) operations, which directly affects the achievable throughput in terms of frames per second; and the regression performance.

We quantify regression performance separately for each output variable using the R^2 score (i.e., the *coefficient of determination*), a standard metric [27] that measures the proportion of the variation in the output that the model explains. A dummy predictor that always outputs the average of the testing samples scores $R^2 = 0$; instead, an ideal regressor yields $R^2 = 1$. Figure 4 reports the R^2 score averaged over the x, y, and z components; it ignores the performance on the ϕ component, which is poor for all models, as we discuss below.

The best regression performance is given by the largest MobileNetV2 model, which is closely followed by PULP-Frontnet ($R^2 = 0.43$ vs. 0.42, respectively). We observe that MobileNetV2 architectures show a consistent pattern and yield better regression performance for larger values of n. This pattern suggests that deeper MobileNetV2 models are more favorable for our problem than wider ones with the same number of parameters.

In terms of memory required to store weights, MobileNetV2 models span from 111 kB to 340 kB, while the PULP-Frontnet model requires 304 kB; all such models fit the L2 memory of the GAP8 SoC. The difference in terms of MACs is much more relevant, with all MobileNetV2 models requiring significantly more operations than PULP-Frontnet, up to 10.4×. Assuming a computational efficiency of 4 MAC/cycle for all models and a CL frequency of 170 MHz, this would result in 20 frame/s and 4.6 frame/s, for the smallest and the largest MobileNetV2 respectively, while PULP-Frontnet would score 48 frame/s. Therefore, we select PULP-Frontnet as our candidate to be deployed and field-tested on the nano-drone.

In Figure 5-A, we report the R^2 score for each output variable of the PULP-Frontnet CNN, including both full precision and quantized versions. x, y, and z outputs yield a good regression performance up to $R^2 = 0.53$, while the ϕ component scores nearly 0, i.e., does not outperform a trivial predictor. Such poor performance on the ϕ component is expected: correctly predicting ϕ implies understanding the nano-drone orientation through low-resolution, monochrome and noisy images; it is a next-to-impossible task also for human observers, since the target appearance is not significantly affected by its orientation, as shown in Figure 5-B/C. Figure 5-A also shows that the post-training quantization stage has a negligible impact on regression performance: R^2 drops by at most 3 percentage points (0.52 to 0.49).

B. Regression performance

Figure 6 further analyzes how the PULP-Frontnet model predictions compare to ground truth for each output variable; an ideal estimator would yield a scatterplot in which all points lie on the diagonal line.

We first observe that the prediction of x (i.e., the target's distance) is challenging. This is expected as x can only be inferred by the size of the target's image, which only covers a few pixels and is, therefore, difficult to assess for the model (and for a human observer). Outputs y and z are estimated with significantly higher accuracy; these variables are related to the position of the target's image on the horizontal and vertical axis, respectively: easier to predict. We further observe that when the target is far, the predictions tend to be more conservative, i.e., closer to 0. This occurs because the model is not confident about the target's position since its image becomes smaller and harder to detect. Furthermore, the y and z components of the relative pose depend on the position of the target's image in the frame but also on the target's distance; therefore, the uncertainty that affects the distance estimation is mirrored in the predictions for y and z.

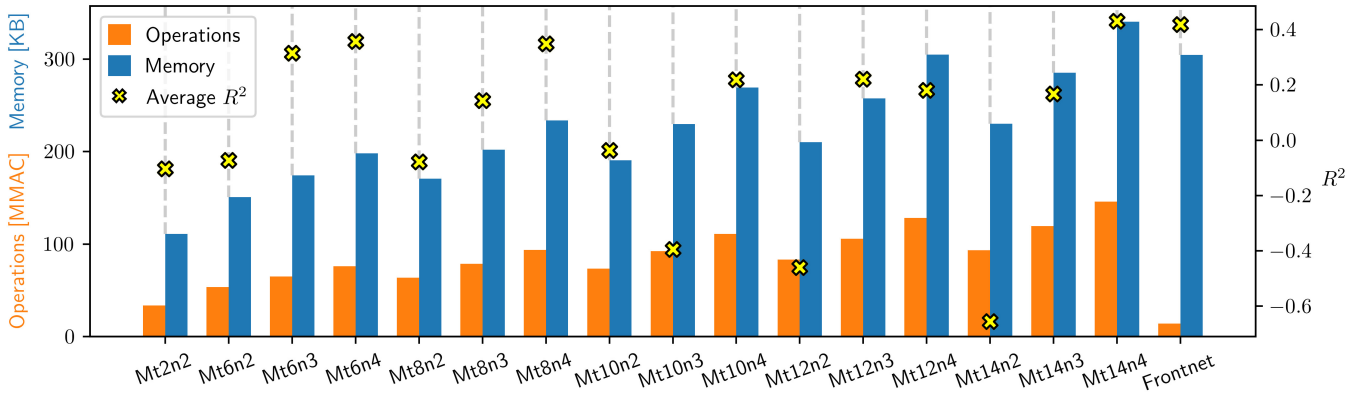

Fig. 4. Overview of all tested architectures; bars refer to the left axis and denote the number of MAC operations (orange) and number of 8-bit weights (blue); crosses refer to the right axis and denote the R^2 score averaged over the x, y and z outputs.

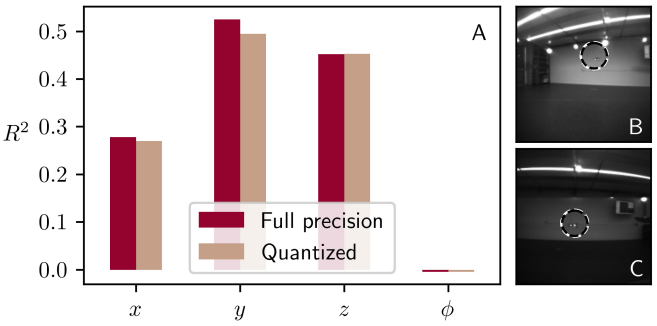

Fig. 5. A) Regression performance of PULP-Frontnet for each output variable; full-precision (red) and quantized (tan) models. B,C) two images with different ϕ (target drone is circled).

On the z component, i.e., the relative height between the two drones, we observe that the predictions are slightly, but consistently, underestimated (by $\sim$5 cm). We correlate this effect to the differences in the training and testing datasets' data distribution or a slight misalignment in the camera pitch on the quadrotor. The rightmost plots of Figure 6 show that the model, as previously explained, does not return any useful information for ϕ, i.e., the orientation component of the relative pose.

C. Onboard real-time performance

This section presents the onboard performance evaluation of the PULP-Frontnet CNN running at different operative points, i.e., voltage and frequencies. As introduced in [3], we explore three configurations, namely *minimum power* (VDD@1.0 V FC@25 MHz CL@25 MHz), *most energy efficient* (VDD@1.0 V FC@25 MHz CL@75 MHz), and *maximum performance* (VDD@1.2 V FC@250 MHz CL@175 MHz). The minimum power is particularly relevant for all those scenarios aiming at maximizing the system's lifetime, for example, when the nano-drone lands and starts behaving as an ubiquitous smart-sensor. Instead, the most energy-efficient configuration should always be preferred if its throughput fulfills the mission's requirements. Lastly, the maximum performance operative point should be considered to maximize the CNN's inference rate, for example, when agility and responsiveness are keys.

TABLE I
PULP-FRONTNET THROUGHPUT AND POWER CONSUMPTION.

Frequencies [MHz]	VDD [V]	Frame-rate [fps]	Power [mW]
FC/CL @ 25/25	1.0	6.8	9.9
FC/CL @ 25/75	1.0	19.7	25.1
FC/CL @ 250/175	1.2	48.3	95.4

Table I reports performance, in frames-per-second (fps), and mean power consumption profiling the GAP8 SoC with a RocketLogger data logger [28] (64 ksps). Results span from a minimum of 6.8 fps@9.9 mW up to 48.3 fps@95.4 mW. The overall nano-drone power breakdown, in Figure 7, shows the expected trends, with the actuators' power consumption dominating (95% of the total) and leaving a small fraction (5%) to the electronics. Finally, the GAP8 SoC, running our monocular relative localization CNN accounts only for the 1.2% of the total, well aligned with the SoA [3], [29].

D. In-field control performance

We perform an in-field test with two flying drones to prove our system. One drone takes the role of the target and performs pre-defined movements, following the same spiral-like trajectory described in Section III-B. The other drone (observer) tries to keep a fixed relative pose w.r.t. to the target drone: a distance of 1 m ($x = 0.8$m); the same height ($z = 0$m); and a position that keeps the target in the center of the frame ($y = 0$m) while keeping the observer's yaw fixed. Our CNN runs in real-time on the observer nano-drone, and its predictions are used for closed-loop control. During this experiment, the pose of both drones is recorded by a motion capture system. The observer drone is oriented in such a way that its x axis (i.e., the forward direction) is aligned with the world's x axis. For quantitative performance evaluation, we compare the observed drone's recorded pose (in world coordinates) with the expected one, as shown in Figure 8. We also include the mean absolute error (MAE) scores for an easier comparison of the three components.

The observer can reliably estimate the relative position for the entire test ($\sim$60 s), as shown in the supplementary video.

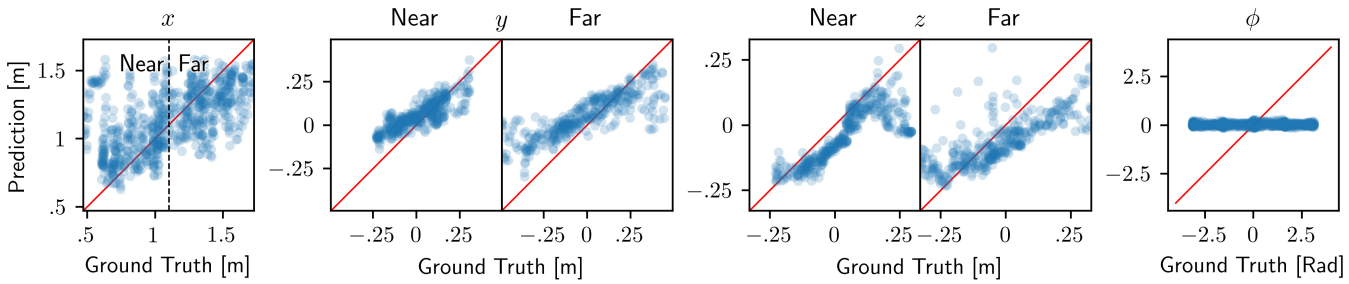

Fig. 6. Predictions (ordinates axis) vs ground truth (abscissae axis) for each of the output variables, for the entire testing set (one point per sample). For y, z and ϕ outputs we separately report samples in which the target is near (x <1.1 m) and far (x ≥1.1 m).

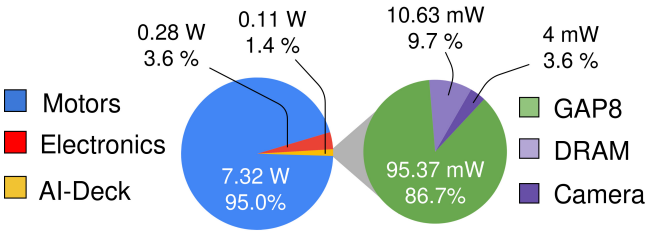

Fig. 7. Robotic platform's power breakdown running PULP-Frontnet at VDD@1.2 V, FC@250 MHz, and CL@175 MHz.

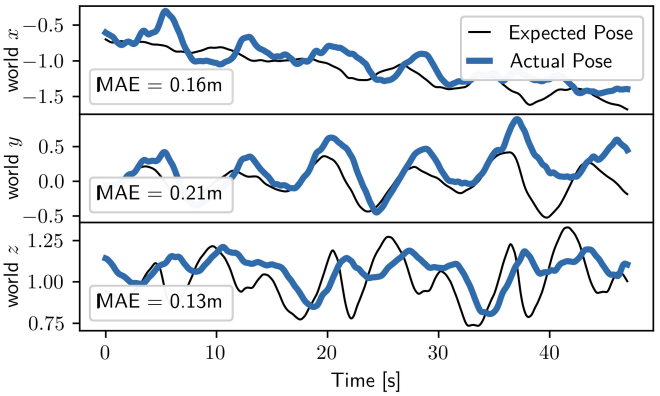

Fig. 8. The x, y and z components of the absolute world pose of the observer drone during in-field tests. The thick blue line represents the true pose of the drone; the thin black line is the ideal pose that the drone is expected to track, i.e. a fixed relative pose from the moving target.

Despite the small latency due to the controller computation, we can observe a close match in absolute coordinates for all the components. The x and z components are the most accurate, with a MAE of 16 cm and 13 cm, respectively. Instead, on the y output, we see that the observer slightly overshoots whenever it needs to move to the left, resulting in a MAE of 21 cm On the other hand, we can observe that the responsiveness on this component is the highest, with the drone swiftly inverting the y direction when needed. Targeting a nano-drone pose estimation task is highly challenging as we need to estimate the pose of a tiny object, which is as big as $\sim 8 \times 18$ pixels at a distance of 0.4 m or $\sim 2 \times 5$ pixels at 1.5 m, i.e., 0.9% and 0.06% of the input image, respectively. Comparing these results with the SoA PULP-Frontnet baseline [3], targeting human pose estimation, the same level of complexity would require a human subject more than 60 meters away from the nano-drone.

We also tested the model in a scenario where the target drone performs a random path instead of the predefined one. The observer successfully locks and holds its relative pose w.r.t. the target drone for the entire duration of a 2-minutes experiment. This experiment is representative of swarm operations where peer drones need to establish line-of-sight conditions. The available supplementary video material shows the ability of the drone to continuously maintain line of sight for approximately half of the drone's lifetime.

Finally, to stress the generalization capabilities of our final CNN, we perform one additional in-field test in a *never-seen-before* indoor environment, which is different from both previous sections' experiments and not present in any dataset. The target drone is remotely controlled by a human pilot along 3D trajectories spanning a volume of approximately $4 \times 4 \times 3$ m. The autonomous observer drone starts its mission at a distance of 1.2 m from the target drone. The experiment results are reported in the supplementary video. Since this environment (i.e., student room) is not equipped with a motion tracking system, we can not compute quantitative metrics on the position tracking performance. The observer drone behaves as expected and successfully tracks the target and keeps it in the field of view for a duration of 53 s while maintaining a set relative distance of x=0.8 m. In contrast, depending on the run, a non-moving observer would have lost view of the target after a timespan of ∼2 to 10 s.

V. Conclusion

This paper addressed the perceptive problem of monocular relative pose estimation between two resource-limited nano-drones. We presented a vertically-integrated system based on deep learning using only visual information. After characterizing multiple (i.e., 17) CNNs, we select the PULP-Frontnet model to run aboard our nano-drone. We cope with the end-to-end development and deployment pipeline, from the dataset collection, augmentation, quantization, and system optimizations. In summary, our results demonstrate precise localization (average control error of 17 cm) of a 10 cm-size target nano-drone by employing only low-resolution monochrome images, up to ∼2 m distance, with long-endurance in-field performance, as much as 2 min experiments.

REFERENCES

[1] 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)*, May 2022, pp. 9689–9695.

[2] D. Palossi, J. Singh, M. Magno, and L. Benini, "Target following on nano-scale unmanned aerial vehicles," in *2017 7th IEEE international workshop on advances in sensors and interfaces (IWASI)*. IEEE, 2017, pp. 170–175.

[3] D. Palossi, N. Zimmerman, A. Burrello, F. Conti, H. Müller, L. M. Gambardella, L. Benini, A. Giusti, and J. Guzzi, "Fully onboard AI-powered human-drone pose estimation on ultra-low power autonomous flying nano-UAVs," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[4] A. Breitenmoser, L. Kneip, and R. Siegwart, "A monocular vision-based system for 6d relative robot localization," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 79–85.

[5] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, "Drone detection using depth maps," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1034–1037.

[6] F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, 2021.

[7] A. Milella, F. Pont, and R. Siegwart, "Model-Based Relative Localization for Cooperative Robots Using Stereo Vision," -, p. 8, 2005.

[8] M. J. Schuster, K. Schmid, C. Brand, and M. Beetz, "Distributed stereo vision-based 6D localization and mapping for multi-robot teams," *J Field Robotics*, vol. 36, no. 2, pp. 305–332, Mar. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.21812

[9] T. Schüppstuhl, K. Tracht, and A. Raatz, Eds., *Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2021*. Cham: Springer International Publishing, 2022. [Online]. Available: https://link.springer.com/10.1007/978-3-030-74032-0

[10] A. Wasik, R. Ventura, J. N. Pereira, P. U. Lima, and A. Martinoli, "Lidar-Based Relative Position Estimation and Tracking for Multi-robot Systems," in *Robot 2015: Second Iberian Robotics Conference*, L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, Eds. Cham: Springer International Publishing, 2016, pp. 3–16.

[11] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "3-d relative positioning sensor for indoor flying robots," *Autonomous Robots*, vol. 33, no. 1, pp. 5–20, Aug 2012. [Online]. Available: https://doi.org/10.1007/s10514-012-9277-0

[12] F. Shan, H. Huo, J. Zeng, Z. Li, W. Wu, and J. Luo, "Ultra-Wideband Swarm Ranging Protocol for Dynamic and Dense Networks," *IEEE/ACM Transactions on Networking*, pp. 1–15, 2022, conference Name: IEEE/ACM Transactions on Networking.

[13] 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*, pp. 1–1, 2022.

[14] F. Vanegas and F. Gonzalez, "Enabling uav navigation with sensor and environmental uncertainty in cluttered and gps-denied environments," *Sensors*, vol. 16, no. 5, 2016. [Online]. Available: https://www.mdpi.com/1424-8220/16/5/666

[15] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1699–1706.

[16] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2174–2182.

[17] M. e. a. Achtelik, "Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments." *The International Society for Optical Engineering*, 2009. [Online]. Available: https://dspace.mit.edu/handle/1721.1/52660

[18] N. Kirchner and T. Furukawa, "Infrared localisation for indoor uavs," -, 01 2005.

[19] V. Walter, M. Saska, and A. Franchi, "Fast mutual relative localization of uavs using ultraviolet led markers," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 1217–1226.

[20] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, "Drone detection using depth maps," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1034–1037.

[21] F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, 2021.

[22] D. Palossi, F. Conti, and L. Benini, "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019, pp. 604–611.

[23] W. Zhao, A. Goudar, J. Panerati, and A. P. Schoellig, "Learning-based bias correction for ultra-wideband localization of resource-constrained mobile robots," *arXiv preprint arXiv:2003.09371*, 2020.

[24] F. Conti, "Technical report: NEMO DNN quantization for deployment model," *CoRR*, vol. abs/2004.05930, 2020. [Online]. Available: https://arxiv.org/abs/2004.05930

[25] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[26] 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*, vol. 70, no. 8, pp. 1253–1268, 2021.

[27] N. J. Nagelkerke *et al.*, "A note on a general definition of the coefficient of determination," *Biometrika*, vol. 78, no. 3, pp. 691–692, 1991.

[28] L. Sigrist, A. Gomez, R. Lim, S. Lippuner, M. Leubin, and L. Thiele, "Rocketlogger: Mobile power logger for prototyping iot devices: Demo abstract," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, 2016, pp. 288–289.

[29] V. Niculescu, L. Lamberti, F. Conti, L. Benini, and D. Palossi, "Improving autonomous nano-drones performance via automated end-to-end optimization and deployment of dnns," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 4, pp. 548–562, 2021.