

Sim-to-Real Vision-depth Fusion CNNs for Robust Pose Estimation Aboard Autonomous Nano-quadcopters

Luca Crupi¹, Elia Cereda¹, Alessandro Giusti¹, and Daniele Palossi¹²

Abstract— Nano-quadcopters are versatile platforms attracting the interest of both academia and industry. Their tiny form factor, i.e., ~ 10 cm diameter, makes them particularly useful in narrow scenarios and harmless in human proximity. However, these advantages come at the price of ultra-constrained onboard computational and sensorial resources for autonomous operations. This work addresses the task of estimating human pose aboard nano-drones by fusing depth and images in a novel CNN exclusively trained in simulation yet capable of robust predictions in the real world. We extend a commercial off-the-shelf (COTS) Crazyflie nano-drone — equipped with a 320×240 px camera and an ultra-low-power System-on-Chip — with a novel multi-zone (8×8) depth sensor. We design and compare different deep-learning models that fuse depth and image inputs. Our models are trained exclusively on simulated data for both inputs, and transfer well to the real world: field testing shows an improvement of 58% and 51% of our depth+camera system w.r.t. a camera-only State-of-the-Art baseline on the horizontal and angular mean pose errors, respectively. Our prototype is based on COTS components, which facilitates reproducibility and adoption of this novel class of systems.

SUPPLEMENTARY VIDEO MATERIAL

In-field tests: https://youtu.be/p4s2j0_6828.

I. INTRODUCTION

Miniaturized autonomous quadcopters (*nano-drones*) are extending the application areas of aerial robotics, from exploration in narrow spaces [1] to human-robot interaction [2]. With a diameter of ~ 10 cm, nano-drones can reach inaccessible places for bigger flying robots and safely operate near humans. Additionally, nano-drone hardware is relatively cheap compared to bigger and more powerful multi-rotors, making these platforms even more attractive.

Still, these platforms come at the price of extremely limited onboard resources, such as memories, processors, and sensors [3]. This is a significant drawback in comparison to standard-sized drones, with a diameter of ~ 50 cm and a few kg of payload, that can cope with complex environments and sophisticated workloads [4], [5] thanks to onboard powerful processors and rich sensors, such as LIDAR [6] and depth cameras [7]. As an example, the GWT GAP8 System-on-Chip (SoC) that equips our nano-drone platform peaks at 22 GOP/s: three orders of magnitude less than the NVIDIA Jetson Xavier AGX flight computer, which achieves up to 32 TOP/s.

This work was partially supported by the Swiss National Science Foundation (SNSF) through the NCCR Robotics.

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

²D. Palossi is also with the Integrated Systems Laboratory (IIS), ETH Zürich, Zürich, 8092, Switzerland dpalossi@iis.ee.ethz.ch

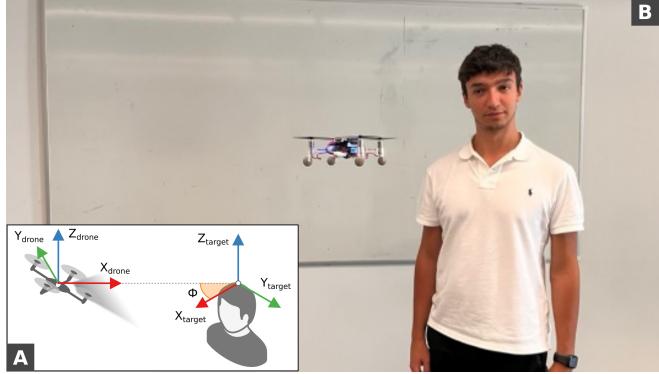


Fig. 1. A) Definition of our person pose estimation task. B) Example of person pose estimation scenario.

When designing nano-drone systems, simple convolutional neural network-based (CNN) approaches are an attractive solution to perception tasks since these models can be used for inference with limited resources. Learning-based approaches are suitable for fusing sensory streams from different onboard sensors [8] since one can learn to manage noisy, low-resolution data and even capture and exploit correlations across sensors. Different types of sensing technologies can complement each other, such as Time-of-Flight (ToF) depth sensors, which are very accurate at a low distance (a few meters), with noisy but longer-range CMOS cameras.

One challenge which arises when adopting this approach is collecting the data to train multi-sensor perception models. Existing large image-only datasets can not be used to train multi-modal CNN since they miss the corresponding inputs from the additional sensors. One solution would be acquiring new data (with all sensors), which is expensive and time-consuming if not unpractical for those cases requiring additional ad-hoc infrastructure such as an external motion capture system (mocap) [9]. An option to extend existing image-only datasets with the missing sensory data can be by developing software models/pipelines to generate additional synthetic samples. However, this strategy is only sometimes applicable and can easily suffer from inaccuracy. A third solution, also used in our work, relies on photorealistic simulators, which can provide abundant data from multiple sensors with little effort and time [9].

Combining *i*) a commercial off-the-shelf (COTS) Bitcraze Crazyflie 2.1 nano-drone, *ii*) an AI-deck expansion board, hosting a parallel ultra-low-power GWT GAP8 SoC [12] and a QVGA monochrome camera, and *iii*) an STM 8×8 multi-zone depth sensor, we introduce and demonstrate our vertically integrated system. Our work addresses estimating

TABLE I
SOA COMPARISON: POSE ESTIMATION TASK ON UAVS.

Work	Pose	Size	Onboard	Sensors	Resolution	Algorithm	Training	Device	Power	Field tested
[10]	object	standard	yes	mono camera	VGA	CNN	real	Jetson TX2	20 W	✓
[4]	object	standard	yes	stereo camera	2× 720p	CNN+geom.	real	Jetson Xavier	20 W	✓
[11]	human	micro	no	mono camera	1080p	CNN+geom.	real	GTX Titan X	100 W	✗
[2]	human	nano	yes	mono camera	QVGA	CNN	real	GWT GAP8	100 mW	✓
Ours	human	nano	yes	depth+camera	8×8 / QVGA	CNN	sim	GWT GAP8	100 mW	✓

the relative pose of a human w.r.t. a nano-drone, as displayed in Figure 1, employing a CNN that fuses data from the two onboard sensors: depth sensor and monocular camera.

We contribute with *i*) the design and thorough analysis of multiple CNN models fed with the two complementary sensory inputs; *ii*) a detailed description of our sim-to-real pipeline, which exploits aggressive photometric augmentations and balanced label distributions; *iii*) comprehensive in-field experimental results, challenging our system in real-world conditions and comparing it across various configurations, including a State-of-the-Art (SoA) baseline [2].

In-field experiments demonstrate *i*) the robustness of our sim-to-real training method; *ii*) the efficiency of the CNN fusing depth and images, which runs aboard the GAP8 SoC up to 45 frame/s within a power envelope of 92 mW; *iii*) the predictive performance of the fusion approach. In particular, on a never-seen-before flying arena, our system significantly outperforms a camera-only SoA baseline: the Mean Absolute Error of the estimated human position and relative orientation angle are reduced respectively by 58% and 51%. Finally, our prototype employs only ready-to-use COTS electronic components to ease the reproducibility and adoption of this novel class of systems.

II. RELATED WORK

Standard/Micro-sized UAVs (50-30 cm diameter) heavily leverage multiple types of sensing devices to address various perception tasks. For example, autonomous drone races, in which a precise understanding of the world is fundamental to navigate at high speeds, exploit depth information for various tasks, including estimating 3D relative poses of gates [4], [10] and supporting visual-inertial odometry pipelines [10]. Similarly, swarm operations of micro-sized drones can exploit depth sensors for mapping and localization tasks [13]. Typical depth sensors used in aerial robotics [14] include stereo cameras like the ZED and Intel RealSense D430 and RGB-D cameras based on ToF technology, such as the Azure Kinect. All these accurate sensing devices come with a form factor, weight ($>100\text{ g}$), and power consumption ($>5\text{ W}$), which prevent them from being also adopted on nano-UAVs. For this reason, aboard our nano-drone, we take advantage of the ultra-compact STM VL53LC5CX 8×8 ToF-based depth sensor (sub-centimeter size and sub-gram weight) and an Himax HM01B0 QVGA camera.

Focusing on our target pose estimation task, CNNs represent a common solution to tackle this problem [4], [5],

[10], [11], also on nano-drones [2]. Sophisticated multi-cameras pipelines can combine CNN-based approaches with geometric ones [4], but at the price of heavier computation. In contrast, multi-modal CNNs directly fuse vision and depth in a unified deep-learning model [8], [15], ensuring the resulting models take full advantage of both modalities [16]. This approach is particularly relevant for our case as we can afford only an extremely low-resolution depth map aboard the nano-drone. Therefore, in this work, we explore multi-modal CNN to maximize the information extracted from the depth map.

Despite the limited computational capabilities of micro-controller units (MCUs) aboard nano-drones, both monocular [3], [17], [18], [19], [2] and stereo [20], [21] vision-based solutions have been proposed. However, this last group of solutions introduces a significant computational load to accomplish relatively low-level control tasks (stabilization and obstacle avoidance). Among these works, the PULP-Frontnet [2] CNN tackles the human pose estimation aboard a Crazyflie nano-drone employing a monocular camera with real-time performance. Starting from this SoA model, which reaches up to 48 frames per second on our platform, we present our novel multi-modal CNN, which fuses images and depth maps by running directly aboard our nano-drone. The constraint posed by the platform in terms of computational capabilities and by the task in terms of minimum frame rate for the in-field tracking, pose huge limits on the type of networks that can be deployed onboard. Networks such as Resnet-18 [22] and transformers [23], with several billion multiply-accumulate operations (MACs) per frame [24], are unsuitable for real-time performance on our platform. Table I summarizes the aforementioned works, comparing them to our proposed approach regarding sensor modalities, adopted algorithms, and computation platforms.

III. BACKGROUND

Robotic platform: our prototype, shown in Figure 2, is based on a Crazyflie 2.1 nano-quadrotor, an open-source 27 g palm-sized COTS nano-drone. An STM32 MCU performs low-level flight control tasks, i.e., state estimation and proportional–integral–derivative (PID) cascade controller. The nano-drone is extended with an AI-deck expansion board tasked with onboard high-level intelligence. The AI-deck features a GWT GAP8 multi-core RISC-V-based SoC, two-level off-chip memories, i.e., 8 MB DRAM and 64 MB Flash,

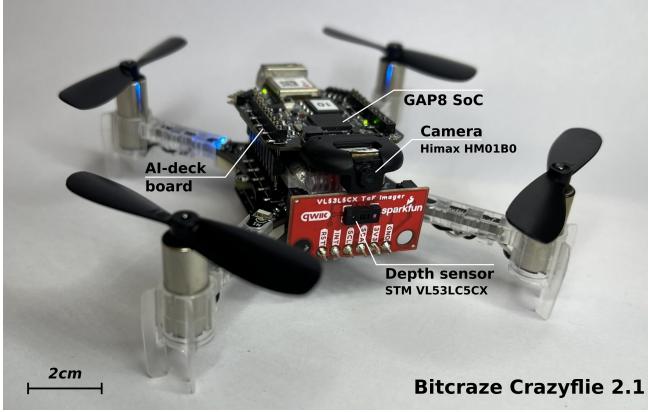


Fig. 2. Our prototype is based on COTS components: the Crazyflie 2.1 nano-drone, the AI-deck board, and an STM VL53LC5CX depth sensor.

and a monocular QVGA Himax HM01B0 @ 60 Hz.

The GAP8 SoC is divided into two power domains: a single-core *fabric controller* which orchestrates accesses to external memories/sensors, and an eight-core *cluster* domain optimized for parallel computation of compute-intense workloads, e.g., CNNs. The on-chip memory hierarchy is composed of a 64 kB low-latency L1 memory shared among the cluster cores and a 512 kB L2 memory. The GAP8 also features two DMA engines that efficiently automate data transfers between memory levels and external peripherals, such as the UART interface, which connects the GAP8 to the STM32. The lack of data caches and floating-point units requires, respectively, explicit data management in software and the use of integer-quantized arithmetic.

We complement our platform with the multi-zone ranging sensor VL53LC5CX¹ from STMicroelectronics, which is based on ToF technology. This sensor is capable of acquiring 8×8 px depth maps at 15 Hz with a 313 mW power consumption. In our nano-drone prototype, we employ a ready-to-use COTS board from SparkFun² mounted on a *Bitcraze prototyping expansion board* connected to the STM32 over the I2C bus. Depth maps are then forwarded over UART to the GAP8 SoC. For distances between 20 mm and 20 cm, the datasheet specifies a measurement accuracy of ±15 mm, while from 20 cm to 4 m the error grows to ±11%.

PULP-Frontnet: this CNN addresses the human pose estimation aboard a nano-drone [2]. It processes monocular gray-scale 160×96 px images and produces the relative pose of the human subject w.r.t. the drone as regression output, represented as 3D position in space (x, y, z) and rotation angle w.r.t. the gravity z-axis (θ). Its training is performed in a supervised manner on real-world data collected in a mocap-equipped room. We adopt this CNN architecture as the backbone for our models, which we extend to take advantage of depth information as described in Section IV-A. In addition, we use it in our experimental evaluation and in-field tests as our SoA baseline.

Simulator: we employ the open-source Webots simulator to generate the training data for our CNN model. In this fashion, we avoid error-prone and time-consuming dataset collection for the CNNs’ training procedure, which would require significant effort given our double sensory sources. We leverage the human models provided by Webots, including 3D models and walking gait animations, and the official Crazyflie 3D model and controllers provided by Bitcraze³. Section IV-B describes our extensions to integrate the multi-zone ranging sensor, model its noise characteristics, and collect data while randomizing the appearance of human subjects and environments.

IV. VISION-DEPTH FUSION

A. Neural networks fusion methods

The sensor fusion and pose estimation task is tackled with the CNN architecture, in Figure 3, based on a PULP-Frontnet [2] backbone. It takes two inputs: a grayscale image (160×96 px) and a depth map (8×8 px) produced by the multi-zone ToF depth sensor; the output consists of 4 variables: (x, y, z), and the angle around the z-axis, (θ).

We compare two approaches to fuse the data from the two sensors. The first approach includes the 8×8 px depth map as one of the 64 channels in input to Block 3 of the backbone, cropped and padded to match the 10×6 shape expected by the block. We name this approach *mid fusion* and show it in Figure 3-B. The second one processes the depth map with a two-layer feed-forward Multi-Layer Perceptron (MLP) network branch, in order to extract a vector of 4 features that are concatenated to the 1920 features of the vision backbone of PULP-Frontnet. We name it *late fusion* MLP and report it in Figure 3-C.

In addition to these two approaches, we consider two baseline methods. These baselines process vision and depth data with separate uni-modal sub-models, that are trained to regress our 4 output variables (x, y, z , and θ). For a given test sample, we then average the outputs of the two sub-models to produce the final prediction. Both baselines share the PULP-Frontnet architecture represented in Figure 3-A as the vision-only sub-model, but differ in the depth-only sub-model architecture. The *Average Depth/Cam Mid* baseline, which reflects the *mid fusion* option above, adopts the last part of the PULP-Frontnet network as a depth-only model: it starts from Block 3 and receives the input depth map as Figure 3-B. On the other hand, for the *Average Depth/Cam Late* baseline, the depth-only model has the architecture represented in Figure 3-C.

All CNNs are trained with SGD at a learning rate of 0.001 over 100 epochs, in which each epoch consists in one pass on 150k images randomly sampled out of our 600k-image training set. We select for testing the network that achieves the best performance on a disjoint 100k-image validation set. The loss function used for training is the sum of the Mean Absolute Error evaluated on each predicted variable x, y, z , and θ .

¹<https://www.st.com/resource/en/datasheet/vl53lc5cx.pdf>

²<https://www.sparkfun.com/products/19013>

³<https://github.com/bitcraze/crazyflie-simulation>

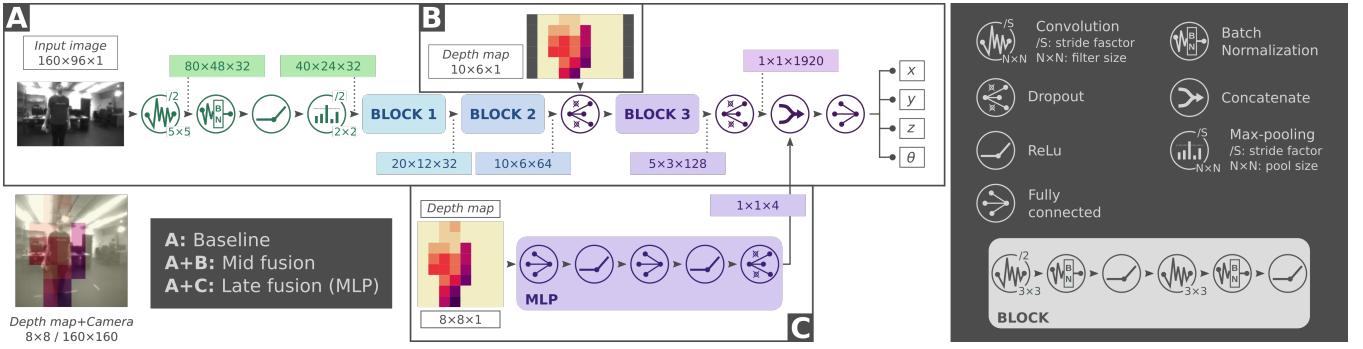


Fig. 3. Our CNNs exploration. A) the SoA baseline [2] used as the backbone of our Depth+Cam fusion. B) Mid-fusion model. C) Late-fusion model.

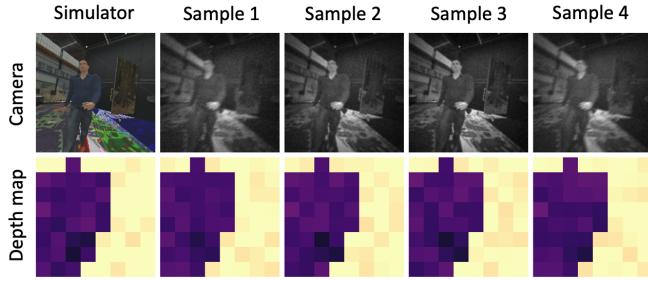


Fig. 4. Four samples of image augmentations starting from the same camera (simulator), coupled with their depth maps.

B. Dataset collection

Our entire train and validation sets are acquired with the open-source simulator Webots, implementing a domain randomization technique [25]. We iterate among 3D models of 27 persons, which vary in height, volume, and texture. The joint poses of the person's skeleton are also randomly sampled among ten extracted from a walking gait animation. The drone is then spawned in the environment with random position (x, y, z) and orientation ($roll, pitch, yaw$), such that it always faces the subject. Finally, we build the environment (background and floor) by randomly sampling among 20 textures, and we populate it with 22 random objects.

On the simulator images, we perform several photometric augmentations, such as motion blur, Gaussian blur, radial distortion, vignetting, per-pixel Gaussian noise, and brightness, to improve our networks' robustness and minimize the sim-to-real gap in the sensor data. For each acquired image, we produce 25 augmented ones by applying all the abovementioned techniques. While on the depth map, we apply only per-pixel Gaussian noise based on the sensor specifications (see Section III). In Figure 4, we report a sample recorded in simulation and four different results of our augmentation pipeline. Finally, we horizontally flip each pair of samples (camera image and depth map), with 50% probability, and adjust the recorded relative pose accordingly. The complete dataset recorded with the simulator consists of more than 700 k samples with output labels distributed as reported in Figure 5. Finally, we complement this training dataset with 3000 additional real-world samples (camera

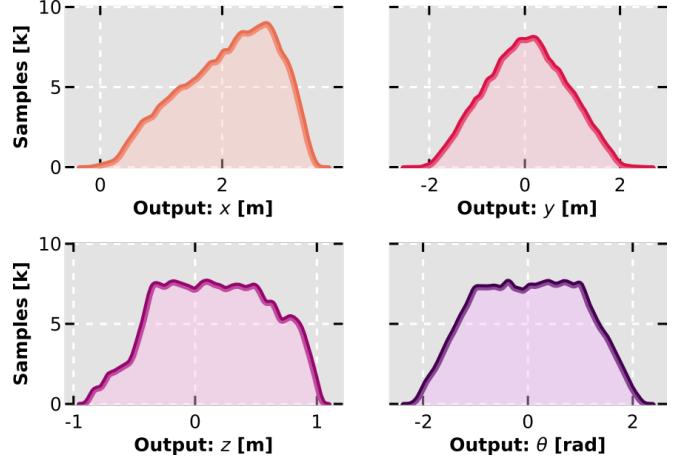


Fig. 5. Sample distribution of our training dataset; each sample consists of one camera image and depth map and the corresponding ground-truth pose.

images, depth maps, and poses) to serve as the testing set.

C. Dropout

We train our networks with two types of dropout. We adopt standard dropout to drop a random subset of activations in our fully-connected layer. In addition, we apply input dropout [16] to force the network to learn from both inputs with equal importance without overfitting on either. More specifically, for every training pair composed of one image and a depth map, we retain both inputs with probability p_{keep} , while with probability p_{drop} one of the two inputs at random is masked. In the mid fusion network, this works by masking the corresponding feature maps in input at the 3rd block. In the late fusion network, the features are masked at the end of Figure 3-C.

D. System integration

Since the GAP8 chip has no floating point unit and performing the computation with software-emulated floats is prohibitively expensive, we quantize weights, biases, and activations, of our CNN, to 8 bits. The deployment phase, with the tiling for efficient memory usage, is performed by a tool that operates on an ONNX file describing the integer-quantized network and produces C code for the management of the memory used by the CNN. Since the network is

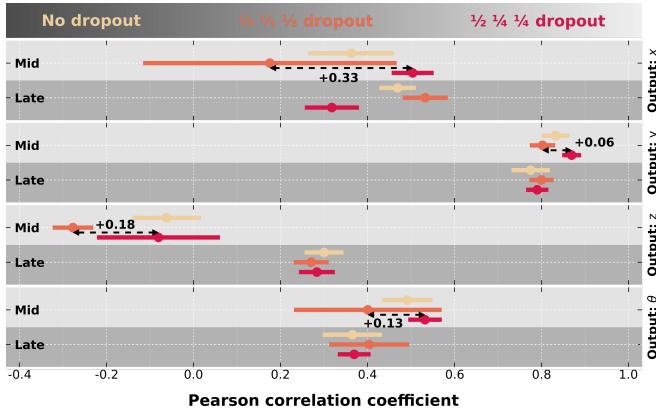


Fig. 6. Average Pearson coefficient over 5 runs changing the dropout rate.

particularly lightweight, the tiling and memory swapping is performed only between L1 and L2 memory, avoiding the use of the L3 memory. The generated C code is integrated into our depth and camera acquisition pipeline onboard the AI-deck. Since the GAP8 SoC does CNN inference on the AI-deck board, while the closed-loop control system runs onboard the Crazyflie, the GAP8 sends the estimated poses to the STM32 via UART. The STM32 integrates them in the closed-loop high-level controller and consequently creates a setpoint for the low-level control system.

V. EXPERIMENTAL RESULTS

A. Regression performance

We evaluate the performance of our regression networks by reporting two metrics on the real-world test set, separately for each output variable: the Mean Absolute Error (MAE) and the Pearson correlation coefficient among predicted and ground-truth values. The former metric measures how close the predictions are to the ground truth. The latter captures whether predictions are linearly correlated to the ground truth but disregards whether there are systematic biases in the predictions. A Pearson score of 0 indicates that an increase in the ground truth value does not yield an expected increase in the predicted value, thus suggesting that the model has no predictive ability and would be useless for control. In contrast, a Pearson score of 1 indicates a model whose outputs are perfectly linearly correlated to the ground truth, but might potentially be offset by some constant or multiplicative factor. When evaluating perception models for downstream control tasks, the Pearson score is indicative of the model’s ability to yield stable in-field performance.

Dropout strategy: to select the best input dropout scheme, we test the three configurations reported in Figure 6: *no dropout* ($p_{\text{keep}} = 1.0$, $p_{\text{drop}} = 0.0$), *uniform dropout* ($p_{\text{keep}} = 1/3$ and $p_{\text{drop}} = 1/3$ for each input), and *non-uniform dropout* ($p_{\text{keep}} = 1/2$ and $p_{\text{drop}} = 1/4$ for each input). We compare these variants by considering the output variables x , y , and θ , disregarding z since it has limited variability in our testing data and therefore does not represent a good benchmark. We observe that, compared to other dropout configurations, non-uniform dropout yields, on average, a lower standard

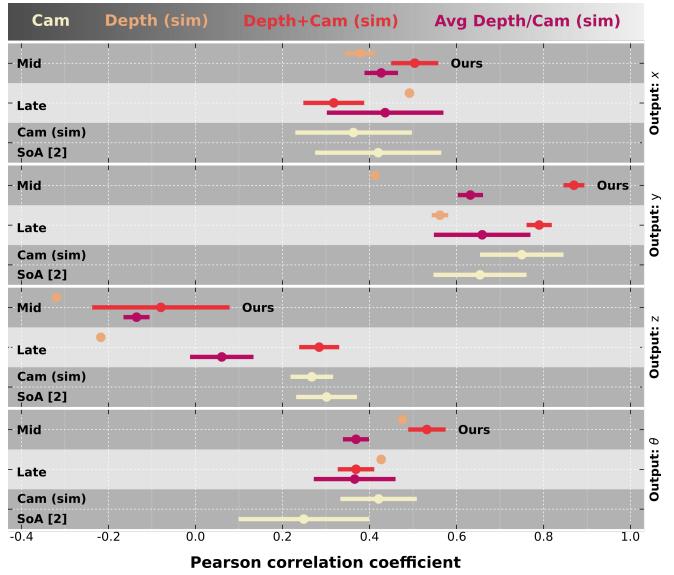


Fig. 7. Means and std. deviations on 5 different training for each model.

deviation and higher Pearson coefficient, highlighted with the black dashed arrows in Figure 6.

Fusion approach: using non-uniform dropout, we then compare the performance of the two fusion approaches: we observe that *mid-fusion* outperforms *late-fusion*. Therefore, we select the mid-fusion model with non-uniform dropout for in-field tests. This model yields a Pearson coefficient for the x , y , and θ variables of 0.50, 0.87, and 0.54, and a MAE of 0.46 m, 0.26 m, and 0.30 rad, respectively.

Comparison with baselines: Figure 7 reports the performance of the mid and late fusion models, named Depth+Cam (sim), against several alternatives: *i*) the **SoA [2]** camera-only approach, i.e., the PULP-Frontnet architecture, trained from real-world data acquired in a different lab; *ii*) **Cam (sim)**, the same model trained on our simulated training set; *iii*) **Depth (sim)**, a depth-only model with an architecture matching the depth branch of the mid-fusion or late-fusion models; *iv*) **Avg Depth/Cam (sim)**: the average of the outputs of the two previous models.

Our *mid-fusion* approach improves the PULP-Frontnet SoA in terms of the Pearson coefficient by 30%, 24%, and 128% on x , y , and θ , respectively. Furthermore, our approach reduces the MAE w.r.t the SoA by 41%, 33%, and 21% on x , y , and θ . The *Cam (sim)* network marginally outperforms the SoA on the x output variable; on the y output variable, it improves both the Pearson and the MAE metrics by 21% and 27%. On the θ variable, we measure a 60% improvement in the Pearson coefficient, while the improvement on MAE is negligible. The baseline Avg Depth/Cam (sim) approach performs similarly to the vision-only based system trained on our simulator training set. Figure 8 illustrates the performance of our approach: a substantial improvement may be observed especially on the y , and θ variables, and for high x values, where the SoA network significantly underperforms.

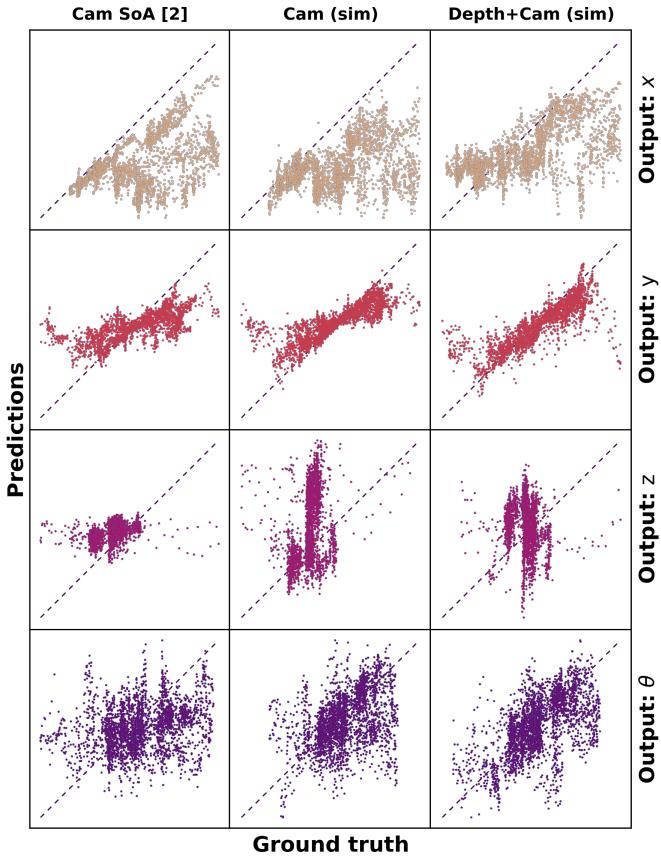


Fig. 8. Predictions (y-axis) vs. ground truth (x-axis) for each system, on all outputs. Dashed diagonal lines correspond to a perfect predictor.

B. In-field evaluation

We put our system to the test in a closed-loop experiment, reproducing the test setup of our baseline [2]: a human subject moves along a predefined path of increasing difficulty while the drone is autonomously controlled to stay in front of the subject, at a distance of 1.5 m. Our experiment comprises two test subjects in a motion capture-equipped environment never seen during training. Flights are performed using three models: the camera-only SoA [2]; Cam (sim); and our proposed approach Depth+Cam (sim). We perform three flights for each combination of subject and model ($2 \times 3 \times 3 = 18$ flights), plus one additional flight in which the drone is controlled based on the perfect mocap position of the subject, for a total of 19 test flights.

Table II reports the results for this experiment broken down into three groups of metrics: overall path completion, regression performance, and control performance. We quantify path completion in terms of mean elapsed flight time and mean percentage of covered distance, terminating an experiment run as soon as the subject leaves the drone camera's field of view. Control performance is quantified with two error metrics: the mean horizontal distance error e_{xy} with respect to the desired position of the drone (i.e., 1.5 meter in front of the subject); the mean absolute angular error e_θ of the drone orientation with respect to its desired orientation (i.e., looking towards the subject).

TABLE II

IN-FIELD EXPERIMENT RESULTS (AVERAGE OVER 6 RUNS). THE SYMBOL * INDICATES THE NETWORKS TRAINED IN SIMULATION.

Network	Flight time [s]	Completed path [%]	MAE			Control error	
			x	y	θ	e_{xy} [m]	e_θ [rad]
Mocap	165	100	0.0	0.0	0.0	0.18	0.21
SoA [2]	140	85	0.79	0.23	0.79	0.99	0.75
Cam*	157	95	0.67	0.39	0.49	0.70	0.53
Depth+Cam*	165	100	0.36	0.12	0.32	0.42	0.37

We observe that the baseline PULP-Frontnet model struggles due to issues in generalization to the unseen environment and subjects, completing only 85% of the path on average. On the other hand, models trained in simulation exhibit higher resiliency, with the Depth+Camera model consistently completing 100% of the path. Regression performance, measured in terms of MAE, captures the ability of a model to estimate the subject's pose accurately. The Depth+Camera model achieves the best absolute performance, especially on y with an MAE as low as 12 cm and less than half the MAE of the PULP-Frontnet baseline on all outputs.

These improvements directly reflect on the control performance, i.e., the accuracy of the closed-loop system in tracking the subject along the path. Depth+Camera shows the best performance by a large margin, less than double the control error of the mocap-based flight, which represents the control error lower bound achievable with perfect sensing. Our supplementary video shows the in-field behavior of the models, where the superior control accuracy of the Depth+Camera (sim) model is clearly noticeable.

C. Discussion

Extensive tests have been done to evaluate how the introduction of the depth sensor affects the physical behavior of the drone. In fact, we tested the SoA with and without the weight introduced by the sensor (not used for the pose estimation). On an average of six runs, the configuration with the depth sensor onboard has remarkably worse in-field control performance. The mean position error, e_{xy} , is 0.99 m for the configuration without the depth sensor and 1.24 m for the configuration with it. Evaluating the mean angular error, e_θ , the performance with the depth sensor onboard deteriorates by 25% starting from 0.75 rad of error of the model without the sensor onboard as reported in Table II.

Further, we analyze the onboard performance of our system. The proposed mid-fusion CNN takes advantage of the additional depth information without increasing the computational or memory requirements of the baseline CNN and thus can reach the same maximum inference throughput of 45.3fps on the GAP8 SoC. The current depth sensor configuration, i.e., 8×8 px @ 15 Hz, results in a given depth map being fed to the CNN for roughly three consecutive inferences when running at the maximum throughput. Future work should explore different trade-offs between depth map resolution and frame rate (up to 4×4 px @ 60 Hz) to determine how they impact our method. Finally, we break

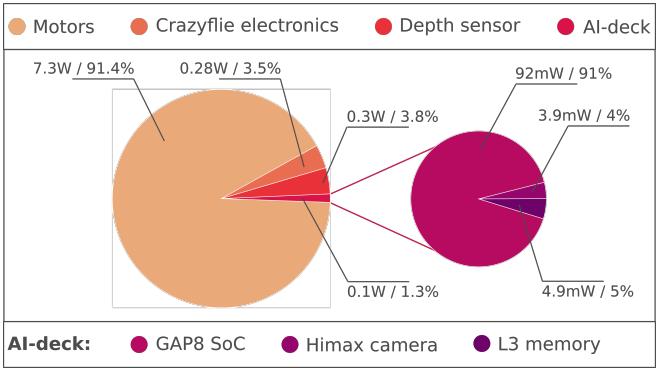


Fig. 9. System power breakdown, while running the Depth+Camera model.

down our system’s power consumption in Figure 9: the depth sensor accounts for an extra 313 mW compared to the baseline, while consumption of the rest of the system remains unchanged. Although only 3.8% of the total power budget, the depth sensor represents 45.8% of the power budget dedicated to sensing and computation. Thanks to the input dropout [16] applied at training time, we expect our CNN’s inference-time performance to gracefully degrade in case of missing inputs, which would enable us to selectively enable the depth sensor only part of the time to save power.

VI. CONCLUSION

This work presents a vertically integrated 10 cm nano-drone system for the human pose estimation task on an ultra-constrained SoC, i.e., sub-100 mW compute power. Combining a COTS 8×8 multi-zone depth sensor with a low-resolution monochrome camera, we explore different CNNs to fuse these complementary inputs. We leverage the Webots simulator to efficiently collect our multi-sensory training data. Thanks to our dataset generation pipeline, which includes aggressive photometric augmentations, balanced label distributions, and multiple environments configurations, we deliver multiple sim-to-real models fully working in real-world testing environments. Finally, we deploy the best models aboard our prototype nano-drone and achieve a real-time inference rate up to 45 Hz, within 92 mW. Our in-field experimental results show an improvement of 58% and 51% of our depth+camera system w.r.t. a camera-only SoA baseline on the horizontal and angular mean pose errors, respectively. Finally, by employing only ready-to-use COTS components, we foster the research community to adopt this novel class of systems.

REFERENCES

- [1] F. Dümbgen *et al.*, “Blind as a bat: Audible echolocation on small robots,” *IEEE Robotics and Automation Letters*, 2022.
- [2] D. Palossi *et al.*, “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, 2021.
- [3] ———, “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)*. IEEE, 2019, pp. 604–611.
- [4] P. Foehn *et al.*, “Alphapilot: Autonomous drone racing,” *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.
- [5] L. O. Rojas-Perez *et al.*, “DeepPilot: A CNN for Autonomous Drone Racing,” *Sensors*, vol. 20, no. 16, p. 4524, Jan. 2020.
- [6] Y. Lin *et al.*, “Mini-uav-borne lidar for fine-scale mapping.” *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 3, pp. 426–430, 2011.
- [7] F. J. Perez-Grau *et al.*, “Multi-modal mapping and localization of unmanned aerial robots based on ultra-wideband and rgb-d sensing,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3495–3502.
- [8] C. Hazirbas *et al.*, “Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture,” in *Computer Vision – ACCV 2016*, S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, Eds. Cham: Springer International Publishing, 2017, pp. 213–228.
- [9] S. Casao *et al.*, “A framework for fast prototyping of photo-realistic environments with multiple pedestrians,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9083–9089.
- [10] S. Jung *et al.*, “Perception, guidance, and navigation for indoor autonomous drone racing using deep learning,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2539–2544, 2018.
- [11] X. Zhou *et al.*, “Human motion capture using a drone,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2027–2033.
- [12] E. Flamand *et al.*, “Gap-8: A risc-v soc for ai at the edge of the iot,” in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2018, pp. 1–4.
- [13] X. Zhou *et al.*, “Swarm of micro flying robots in the wild,” *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.
- [14] L. O. Rojas-Perez *et al.*, “On-board processing for autonomous drone racing: An overview,” *Integration*, vol. 80, pp. 46–59, 2021.
- [15] M. Schwarz *et al.*, “Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1329–1335.
- [16] S. de Blois *et al.*, “Input dropout for spatially aligned modalities,” in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 733–737.
- [17] R. Bouwmeester *et al.*, “Nanoflownet: Real-time dense optical flow on a nano quadcopter,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1996–2003.
- [18] S. Chen *et al.*, “Towards specialized hardware for learning-based visual odometry on the edge,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 10 603–10 610.
- [19] S. Bonato *et al.*, “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.
- [20] C. De Wagter *et al.*, “Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4982–4987.
- [21] K. McGuire *et al.*, “Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1070–1076, 2017.
- [22] K. He *et al.*, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [23] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fdb053c1c4a845aa-Paper.pdf
- [24] C. Liu *et al.*, “Network amplification with efficient macs allocation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 1933–1942.
- [25] J. Tobin *et al.*, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.