IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Self-Supervised Learning of Visual Robot Localization Using LED State Prediction as a Pretext Task

Mirko Nava[1], Nicholas Carlotti[1], Luca Crupi[1], Daniele Palossi[12], and Alessandro Giusti[1]

***Abstract*—We propose a novel self-supervised approach for learning to visually localize robots equipped with controllable LEDs. We rely on a few training samples labeled with position ground truth and many training samples in which only the LED state is known, whose collection is cheap. We show that using LED state prediction as a pretext task significantly helps to learn the visual localization end task. The resulting model does not require knowledge of LED states during inference.**

We instantiate the approach to visual relative localization of nano-quadrotors: experimental results show that using our pretext task significantly improves localization accuracy (from 68.3% to 76.2%) and outperforms alternative strategies, such as a supervised baseline, model pre-training, and an autoencoding pretext task. We deploy our model aboard a 27-g Crazyflie nano-drone, running at 21 fps, in a position-tracking task of a peer nano-drone. Our approach, relying on position labels for only 300 images, yields a mean tracking error of 4.2 cm versus 11.9 cm of a supervised baseline model trained without our pretext task. Videos and code of the proposed approach are available at https://github.com/idsia-robotics/leds-as-pretext.

***Index Terms*—Deep Learning for Visual Perception; Deep Learning Methods; Micro/Nano Robots**

I. Introduction

THE ability to estimate the position of a target robot in a video feed is crucial for many robotics tasks [1], [2], [3]. State-of-the-art (SoA) approaches use deep learning techniques based on Convolutional Neural Networks (CNNs) [4]: given a camera frame, they segment the target robot, regress the coordinates of its bounding box or its position in the image. Training these approaches to handle new robots or environments requires extensive labeled datasets, which are time-consuming and expensive to acquire, often relying on specialized hardware, e.g., motion tracking systems, to generate ground truth labels.

This article presents an approach to drastically reduce the labeled data required to train such models, building upon recent results in Self-Supervised Learning [5]. In the robotics literature (see Section II), the term Self-Supervised denotes two distinct paradigms. In the first, a robot system autonomously generates labeled data for the task of interest, named *end task*, and is trained in a standard supervised way. This paradigm has been used in robotics since the mid 2000s [6], [7], [8], [9]. As a recent example, Li et al. [4] use nano-drones equipped with SoA algorithms to automatically acquire camera frames and the corresponding relative location of the target drone. In the second paradigm, a robot system autonomously generates abundant labeled data for a *pretext task*: the pretext task requires similar perception skills as the end task while relying on cheaper ground truth that is easier or free to collect. Then, a model is trained to solve both tasks simultaneously using an additional dataset containing only few labels for the end task. Despite not being useful during deployment, the pretext task forces the model to learn meaningful features, boosting the performance on the end task. The paradigm is widely successful in the deep learning literature [5] and has only recently been adopted for perception applications [10], [11].

This article introduces a novel approach based on the second paradigm, tailored to robotics applications, and suitable for deployment on resource-constrained platforms. Our **contribution**, presented in Section III, is the use of target robot LED state prediction (ON or OFF) as a pretext task to improve the learning process of a visual localization end task. By learning to predict the state of the LEDs aboard, the model learns features that are also useful to localize the target robot. The idea is compelling because most robot platforms feature controllable LEDs: during data collection, the target robot blinks its LEDs and radio-broadcast their state; at the same time, another robot automatically collects images annotated with LED state ground truth.

We instantiate this general idea to a specific, challenging end task: predict the image-space position of a target nano-drone given a low-resolution, low-dynamic-range image acquired by the camera of a peer nano-drone, as shown in Figure 1. A Fully Convolutional Network (FCN) model [12] simultaneously learns to solve pretext and end tasks using the dataset described in Section IV, containing only few samples labeled with the drone's location. We provide detailed comparisons and an ablation study on the Bitcraze Crazyflie 2.1[1] nano-drone in Section V. Results show the proposed pretext task to significantly improve performance over a supervised baseline, different pre-training strategies, and an autoencoding pretext task. The model generalizes well to unseen environments,

Manuscript received: September 13, 2023; Revised November 9, 2023; Accepted January 26, 2024. This paper was recommended for publication by Editor X. Liu upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Swiss National Science Foundation, grant number 213074.

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

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

Digital Object Identifier (DOI): see top of this page.

[1]https://www.bitcraze.io/products/Crazyflie-2-1

and is capable of localizing multiple drones simultaneously. Finally, we deploy the model aboard the target platform to complete a vision-based position tracking task. Conclusions are drawn in Section VI.

II. Related Work

A. Relative Visual Localization of Drones

Drone-to-drone relative localization approaches rely on various sensors, including microphones, infra-red sensors, Ultra-Wide Band (UWB), color and depth cameras. In particular, microphones can be used for localization, integrating distance estimates from a drone beacon emitting a specific sound [13]. Multiple infra-red sensors with known geometry allow the triangulation of a drone equipped with infra-red emitters [14]. Camera-based approaches rely on visual fiducial markers such as circles printed on paper [15], light-emitting markers [16], by detecting the drone in depth images with handcrafted [17], or learned [18] models. In our work we use monocular grayscale images as the model's input. LEDs, which come already integrated with the adopted platform, are exclusively used to generate data for the self-supervised pretext task and are not used during inference.

UWB is a radio communication technology recently adopted for localization tasks [19], [20], [21], [22], [23], enabling communication between multiple robots and providing a distance measurement through the Received Signal Strength Intensity (RSSI). RSSI measures the amount of radio signal received from a source and is used to derive its distance. Using three non-collinear UWB sensors enables the triangulation of robots [19]. A single sensor requires more complex approaches, such as integrating distance measurements from UWB beacon drones moving in a pattern [20]. Communication is used during localization to combine distance measurements with broadcasted state-estimates [21], [22] and optimizing a camera-based initial guess [23].

In contrast, our approach does not require specialized hardware that supports the communication, and does not assume the target drone to share information with the observer drone during inference.

B. Self-Supervised Relative Drone Localization

Self-supervised approaches proposed for object localization tasks [9], [24], [25], [26] can, in principle, be used to localize drones. Objects are localized by fitting their known 3D model onto a monocular image [24] or onto a point-cloud obtained by segmenting multiple RGB-D images [9]. Other approaches do not require knowledge of the 3D model of the objects of interest. Instead, they learn by using a pre-trained model and moving the object to generate more training data [25] or by combining state estimates with sparse trusted information, e.g., that coming from a fiducial marker [26].

Self-supervised relative drone localization approaches learn a model with limited access to labeled data, using UWB to provide ground truth [4], or a stereo microphone for an audio-based pretext task [10]. In detail, Li et al. [4] pre-train a purely visual estimator using synthetic data, then fine-tune it using a small labeled dataset generated autonomously from UWB nodes [22]. In contrast, our approach introduces a pretext task defined on images with no ground truth for the target position: it is based solely on LED state estimation, and does not require additional hardware besides controllable LEDs – which are present on most robot platforms.

We explored cross-modal self-supervised learning of visual quadrotor localization in recent work [10], using images acquired by a ground robot equipped with a stereo microphone. The pretext task consists of predicting features (intensity in various frequency bands) of the perceived sound of a quadrotor, given an image. By solving this pretext task, the model is forced to learn features of the perceived sound that, in turn, are informative of the drone's location.

The present work proposes a more general pretext task that does not rely on additional sensors, such as a microphone, and is suitable for applications with limited power budget. The only requirement is that the target robot is able to vary its appearance for the observer: in the absence of controllable LEDs, which are the most straightforward and convenient way to achieve this, one may rely on any other actuator that affects the robot appearance, e.g., raising a limb.

III. LED State Prediction as a Pretext Task

We consider visual robot-to-robot localization problems, in which an observer robot has to predict the position of a target robot on the image plane. The observer robot takes a monocular image from its forward-looking camera and predicts the position of the target robot visible in the image. Additionally, we require the target robot to be equipped with controllable LEDs.

We collect tuples consisting of $\left\{\langle \boldsymbol{i}_j, \boldsymbol{p}_j, l_j \rangle\right\}_{j=1}^{N}$ where $\boldsymbol{i} \in \mathbf{R}^{whc}$ denotes a camera frame of $w \times h$ pixels and c channels, $\boldsymbol{p} \in \mathbf{R}^2$ the image-space position of the robot, and l the shared state of the four robot LEDs, which can be either all OFF, represented with a 0, or all ON, represented with a 1.

In the following, we call samples *labeled* when the drone's position $\boldsymbol{p}$ is known or *unlabeled* otherwise. We denote the set containing the (possibly small) amount of labeled samples with $\mathcal{T}_\ell$ and the unlabeled set with $\mathcal{T}_u$. We also collected a separate labeled set $\mathcal{Q}$ that serves as a testing set and on which we compute performance metrics.

We learn a Neural Network (NN) model $\boldsymbol{m}$ that, given a monocular image, predicts two maps: the location map $\hat{\boldsymbol{q}}$ containing likely drone locations, and the LED state map $\hat{\boldsymbol{l}}$ the probability of seeing a drone with its LEDs on, $\left(\hat{\boldsymbol{q}}, \hat{\boldsymbol{l}}\right) = \boldsymbol{m}\left(\boldsymbol{i}|\boldsymbol{\theta}\right)$, where $\boldsymbol{\theta}$ is the set of trainable network weights. Specifically, we consider a Fully Convolutional Network (FCN) architecture consisting solely of convolutional layers and whose output consists of two maps. Using a map to represent the drone's location has two advantages compared to using the drone's coordinates [27]. First, it allows one to handle images with zero, one or more visible drones [4]. Second, it enforces an inductive bias by limiting the receptive field of each cell of the output map; in fact, we expect the target drone to cover a small portion of the input image [28].

A ground truth location map $\boldsymbol{q} \in [0, 1]^{wh}$ of $w \times h$ cells is generated from the robot's position $\boldsymbol{p} = (u, v)$: we start with

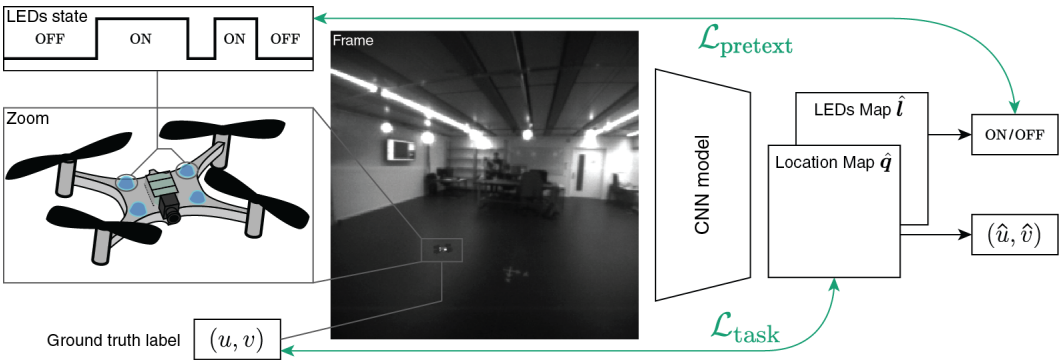


Fig. 1. A fully convolutional network model is trained to predict the drone position in the current frame by minimizing a loss $\mathcal{L}_{\text{task}}$ defined on a small labeled dataset $\mathcal{T}_l$ (bottom), and the state of the four drone LEDs, by minimizing $\mathcal{L}_{\text{pretext}}$ defined on a large dataset $\mathcal{T}_l \cup \mathcal{T}_u$ (top).

a map filled with zeros and place a circle of radius $r = 4$ pixels centered in $\boldsymbol{p}$, filled with ones and with a soft-edge transitioning to zero.

We train $\boldsymbol{m}$ by optimizing the weights $\boldsymbol{\theta}$ through gradient descent steps, minimizing the loss function $\mathcal{L}$. The loss, in turn, is defined as the weighted sum of two terms: the first term $\mathcal{L}_{\text{task}}$ consists of a regression loss computed on the labeled training set $\mathcal{T}_\ell$, whose aim is to learn the robot localization task, and defined as

$$\mathcal{L}_{\text{task}} = \frac{1}{|\mathcal{T}_\ell|} \sum_{i=1}^{|\mathcal{T}_\ell|} \text{mean}\left(|\hat{\boldsymbol{q}}_i - \boldsymbol{q}_i|^2\right) \tag{1}$$

where mean is the average of the map cells. The second term $\mathcal{L}_{\text{pretext}}$ consists of a classification loss defined on the union of the training sets $\mathcal{T}_\ell \cup \mathcal{T}_u$, learning the LED state prediction task, and defined as

$$\mathcal{L}_{\text{pretext}} = \frac{1}{|\mathcal{T}_u \cup \mathcal{T}_\ell|} \sum_{i=1}^{|\mathcal{T}_u \cup \mathcal{T}_\ell|} \text{BCE}\left(\text{mean}\left(\hat{\boldsymbol{l}}\right), l\right) \tag{2}$$

where BCE is the binary cross-entropy. To obtain the scalar $\hat{l}$ representing the probability of the drone's LEDs being on, we compute the average of the LED state map $\hat{\boldsymbol{l}}$[2].

The complete loss function is $\mathcal{L} = (1-\lambda)\,\mathcal{L}_{\text{task}} + \lambda\,\mathcal{L}_{\text{pretext}}$, where $\lambda \in [0, 1]$ controls the tradeoff between the two loss terms during training. In (1) and (2), each loss is weighted by the reciprocal of the dataset size on which it operates, ensuring that the impact of each loss during training is comparable when working on differently-sized datasets.

IV. Experimental Setup

In the following, we instantiate the presented approach to the challenging task of drone-to-drone localization, as shown in Figure 2. This task represents the broader set of image-based robot localization tasks, as many mobile robots feature cameras and controllable LEDs. Among platforms to which our approach is applicable, we specifically selected nano-drones for our experiments: they are difficult to localize due to their small dimensions and complex shape. Additionally, they have constrained resources: the camera is low resolution, low dynamic range, and has a limited field of view; the onboard microprocessor imposes limitations on the breadth and depth of the NN, especially for real-time applications.

A. Robot Platform

The platform of choice is the Bitcraze Crazyflie 2.1, a nano-drone measuring 10 cm in diameter and weighing only 27 g, extended by the Ai-deck companion board, see Figure 2. The Ai-deck provides a forward-looking monocular camera, acquiring 320 × 320 pixels grayscale images, and a GWT GAP8 Parallel Ultra-Low Power (PULP) System-on-Chip (SoC) [29] extending the basic computational capabilities, i.e., state estimation and low-level control, offered by the STM32 microcontroller available on the nano-drone. We employ the GAP8 SoC to boost the execution of NNs, which require integer quantization to exploit its 8-core general-purpose cluster due to the lack of floating point support. Additionally, the drone features on its body four controllable LEDs, which we exploit to define the pretext task.

We consider a scenario in which two identical Crazyflie drones fly in the environment: one drone takes the observer role, acquiring camera frames in which the other drone (target) is visible.

B. Datasets

Our experimental validation is based on Nano2nano[3], a dataset collected in a 10 × 10m lab equipped with a motion-tracking system and consisting of 72 different sequences. For each sequence (average length of 210 seconds and 830 frames), the target Crazyflie flies a pseudo-random trajectory with the four controllable LEDs switched either ON or OFF. At the same time, the observer drone continuously moves to increase the variability of represented backgrounds. To further increase data variability, in each sequence the camera exposure setting is set to one of three possible values. The trajectory is computed so as to keep the target in the camera view and cover the image space as uniformly as possible, with distances ranging from 0.2 meters to 2 meters. Each frame is labeled with the target pose relative to the observer's camera, its position in the image, and the state of the LEDs.

[2]One may obtain $\hat{l}$ as the average of $\hat{\boldsymbol{l}}$ weighted by the map $\hat{\boldsymbol{q}}$; in our experiments, this resulted in less stable training and a lower performance.

[3]https://github.com/idsia-robotics/drone2drone_dataset

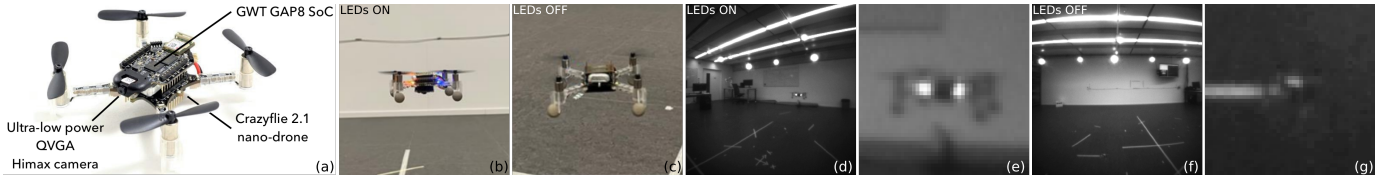


Fig. 2. The palm-sized Bitcraze Crazyflie 2.1 nano-drone platform (10 cm in diameter). (a) The drone's hardware and its four controllable LEDs; (b, c) high-resolution pictures of the flying drone; (d, f) samples from our dataset; (e, g) zoom-in on the drone using the model's receptive field (45×45 pixels).

Half of the 72 sequences are used as the testing set $\mathcal{Q}$ (30k samples). Data from remaining 36 sequences is partitioned into the labeled training set $\mathcal{T}_\ell$ (1k samples) and the unlabeled training set $\mathcal{T}_u$ (29k samples). For an approach described in Section IV-C, we employ the synthetic training dataset $\mathcal{T}_s$ proposed in [4, Section IV] consisting of 800 random-background images depicting the drone in a random pose. Images are converted to grayscale and padded with a solid random gray value to match size and channels of our data.

Additional generalization experiments are reported in Section V-D and shown in the supplementary video; these experiments use data recorded in different rooms, without ground truth for the drone location.

C. Alternative Strategies

We assess the validity of our approach, named LED state prediction Pretext (LED-P), against various alternatives. First, we consider a naive model (DUMMY) that always predicts the mean position on the labeled training set $\mathcal{T}_\ell$. The Baseline (BAS) strategy involves training using only $\mathcal{L}_{\text{task}}$ (achieved with $\lambda = 0$) on the labeled training set $\mathcal{T}_\ell$. The Upper Bound (UB) strategy is used to estimate the maximum achievable performance. It minimizes only $\mathcal{L}_{\text{task}}$, assuming to have access to ground truth position labels for both $\mathcal{T}_\ell$ and $\mathcal{T}_u$, representing a fully-supervised scenario where ground truth is cheap and abundant.

We also consider alternative strategies: Autoencoding Pretext (AE-P), Contrastive Language-Image Pre-training (CLIP) and Efficient Deep Neural Networks (EDNN).

Undercomplete autoencoders are a frequently-adopted strategy for taking advantage of unlabeled data, defining an image-reconstruction pretext task [5]. The intuition is that by learning to compress and decompress an image, autoencoders learn a high-level representation that can be useful to solve the end task. In AE-P, we train an autoencoder on $\mathcal{T}_\ell \cup \mathcal{T}_u$ by minimizing the MS-SSIM [30] between input and reconstructed images; then, an additional NN head learns the localization task using $\mathcal{L}_{\text{task}}$ on $\mathcal{T}_\ell$, taking as input features computed by the autoencoder's bottleneck.

CLIP is a powerful bi-modal feature extractor trained to minimize the distance of embeddings between an image and its caption [31]. The learned image encoder is shown to outperform supervised models in many zero- and few-shot tasks. In CLIP, we take the features extracted from the pre-trained image encoder and pass them to a NN head trained for the localization task using $\mathcal{L}_{\text{task}}$ on $\mathcal{T}_\ell$.

In EDNN, we consider supervised pre-training on synthetic images, as described in [4], to cheaply generate labeled data. The strategy consists first in training using the synthetic dataset $\mathcal{T}_s$ on the localization task with a focal loss [32], and then fine-tune the NN parameters using $\mathcal{L}_{\text{task}}$ on $\mathcal{T}_\ell$.

D. Network Architectures and Training

BAS, UB, EDNN and LED-P share the same tiny FCN [12] architecture consisting of nine convolution blocks, in order: two blocks with 8 channels, $2\times$ max-pooling, three with 16, $2\times$ max-pooling, three with 32, $2\times$ max-pooling and one with 2 channels as the output, totalling 22.1k trainable parameters. Convolution blocks consist of a convolution layer, batch normalization and ReLU activation. The model's input is a grayscale image of 320×320 pixels, normalized between zero and one. The model produces two maps of 40×40 cells, each cell with a 45×45 pixels receptive field, as illustrated in (e, g) of Figure 2. Cells of the first map denote drone presence in the corresponding area of the input image[4], while cells of the second denote whether the drone has its LEDs on (1.0) or off (0.0); cells with no visible drone are expected to have a value close to 0.5.

AE-P uses an encoder with four convolution blocks with 4, 8, 16 and 32 channels, interleaved by $2\times$ max-pooling layers, and terminating with the Fully-Connected (FC) bottleneck; in our experiments we considered bottlenecks of 512 and 1024 neurons. The decoder uses four convolution blocks with channels symmetrical to the encoder and interleaved by $2\times$ bilinear upsampling. We attach to the bottleneck a convolution head responsible for localizing the drone, consisting of two convolution blocks with 32 and 1 channels and interleaved by $2\times$ bilinear upsampling.

CLIP uses the pre-trained image encoder of the homonymous model [31] as a feature extractor; specifically, we adopt the variant using the vision transformer ViT-B/32 producing 512 features. We follow a similar approach to what is presented in [31, Section 3] but keeping CLIP parameters frozen and replacing the logistic regression with a FC head. A FC head uses two blocks composed of a FC layer, batch normalization and ReLU activation. We conducted many trials to find performing architectures that use the 512 CLIP features; we report here the best two: the top performer with 16 neurons and the second with 512, followed by the output block of 400 neurons reshaped into a 20×20 grid.

Finally, we compare LED-P with Frontnet [27], an approach that directly regresses the drone's coordinates.

All NNs are trained using Adam [33] as the optimizer, running for a total of 200 epochs. We adopt a scheduler that

[4]Map cells are independent one another and assume values in the range from 0 (no drone) to 1 (drone present in given cell).

divides the learning rate by a factor of 5 every 50 epochs, starting with a learning rate $\eta_{\text{start}} = 1e^{-2}$ and reaching a final learning rate $\eta_{\text{final}} = 8e^{-5}$. In each epoch we randomly draw mini-batches of 64 examples from the two joined training sets and minimize the loss function described in Section III. Specifically, we minimize $\mathcal{L}_{\text{pretext}}$ (setting $\mathcal{L}_{\text{task}}$ to 0) for examples taken from $\mathcal{T}_u$ since there are no known labels, and the complete loss $\mathcal{L}$ when fed samples from $\mathcal{T}_\ell$.

Additionally, to increase the variability of the drone's visual appearance, we apply the following augmentations: horizontal flip (50% probability), random rotation (uniform $\pm 9°$), random translation (uniform ± 32 pixels), and apply multi-frequency simplex noise.

E. From Grid Map to Robot Position

We consider two approaches to recover $\hat{\boldsymbol{p}}$ from the model's predicted map $\hat{\boldsymbol{q}}$ named argmax and barycenter: argmax selects the coordinate of the cell of $\hat{\boldsymbol{q}}$ whose value is largest; barycenter computes the expected drone position by averaging the coordinates of each cell weighted by the corresponding probability of depicting a drone [34]. The probability of each cell is obtained by normalizing $\hat{\boldsymbol{q}}$ s.t. its sum equals one. As shown in Figure 3, barycenter returns conservative estimates, biased towards the dataset's mean. In contrast, argmax yields unbiased results at the expense of larger errors for frames with no detected drone. Banding artifacts are present with argmax since it cannot represent positions inbetween cells of the 40×40 map, i.e., it discretizes the input image coordinates into 8-pixel-wide bins. In the following experiments we use the argmax approach.

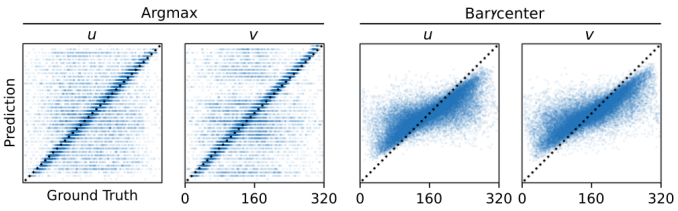


Fig. 3. LED-P model predictions on the test set $\mathcal{Q}$ with argmax and barycenter approaches for the u and v components of the drone's position.

F. Evaluation Metrics

We compare models on different metrics computed on the test set $\mathcal{Q}$. The Pearson correlation coefficient ρ, computed separately for the horizontal u and vertical v components of $\boldsymbol{p}$; it measures the linear correlation between predicted and ground truth values. We also compute the model's error distribution using the euclidean distance between $\boldsymbol{p}$ and $\hat{\boldsymbol{p}}$. From this distribution, we derive the median value in pixels $\widetilde{\text{D}}$ and a precision score P_k. We chose median instead of mean for being a robust central tendency estimate for skewed distributions. P_k is the fraction of samples whose position error is lower than k pixels, considering predictions with a distance smaller than the threshold k as correct, similarly to ADD for 6 Degrees of Freedom pose estimation [24]. Additionally, we report P_k^+, defined as the relative improvement of LED-P models with respect to the corresponding baseline BAS, such that BAS represents 0%, and UB 100%. In our experiments we consider $k = 30$ pixels, i.e., approximately 10% of the edge length of images.

Even though LED state prediction is not our end task, we also report the Area Under the Receiver Operating Characteristic Curve (AUC) for the LED classification output: it measures how well a model distinguishes the two classes at various thresholds, i.e., telling between a drone with LEDs off or on. A random classifier achieves an AUC score of 50%, while an ideal classifier achieves a score of 100%.

V. Results and Discussion

This section reports the performance of our proposed strategy (Section V-A), how performance changes as a function of λ and of the amount of labeled examples (Section V-B), a comparison with alternative approaches (Section V-C), generalization ability of our proposed strategy (Section V-D) and a deployed in-field experiment on nano-drones (Section V-E).

A. LED State Prediction Improves Performance

In Table I, first and last panels, we report the performance of our LED-P strategy against a dummy model (DUMMY), baseline (BAS) and an upperbound (UB). We observe that LED-P-100, trained leveraging unlabeled images, performs moderately better than BAS-100 across all evaluation metrics. The P_{30} metric indicates that 86.9% of predictions fall within 30 pixels from the respective ground truth position, with a median error of only 8 pixels out of a 320×320 pixels image. We also computed P_{30} scores on two subsets of $\mathcal{Q}$ containing examples with LEDs off and on, on which our LED-P-100 model scores 83.3% and 91.2% respectively, showing more difficulties in localizing drones with LEDs off. On the same metric, we report LED-P-100 to score higher than BAS-100 with a p-value of 0.029, computed with the non-parametric one-sided Mann-Whitney U test. This model achieves a P_{30} of 88.3% when localizing the drone in brighter images and 85.3% in darker ones, showing a slight performance drop in the latter case.

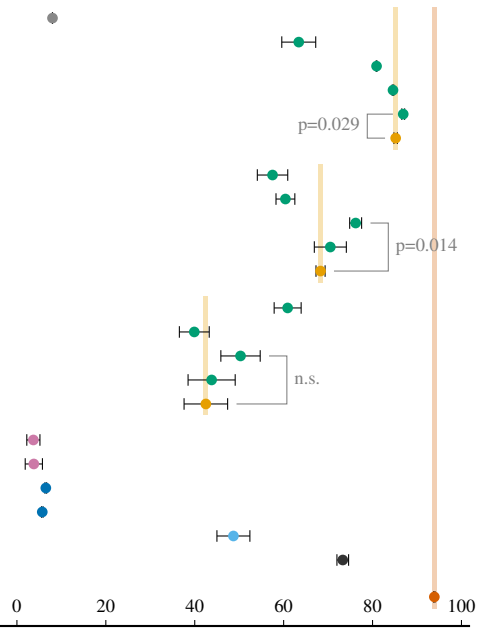
B. Impact of λ and amount of Labeled Examples

In Figure 4, we inspect the LED-P-30 ($\lambda = 0.001$) prediction against BAS-30 on samples taken from $\mathcal{Q}$, where it scores 30.9% in P_{30}^+, a considerable improvement over the respective baseline. Our model demonstrates an overall good performance when the target drone flies at or below the camera height as the drone's LEDs are more easily visible, and to a lesser extent when the target flies higher, which reduces the LEDs' visibility. On the LED state prediction, our model scores 74% in AUC despite the drone LEDs not being visible in many of $\mathcal{Q}$ samples. Failed detections occur when the model predicts the position of similar looking areas of the image; this happens less frequently when LEDs are on since their presence improves the drone's visibility.

The approach is robust to the unlabeled training set $\mathcal{T}_u$ containing some frames in which the drone is out of the field

TABLE I
COMPARISON OF MODELS, 5 REPLICAS PER ROW. PEARSON CORRELATION COEFFICIENT ρ_u AND ρ_v, PRECISION P_{30} AND MEDIAN OF THE ERROR $\widetilde{D}$.

Model	Training set for task End	Training set for task Pretext	λ	ρ_u [%] ↑	ρ_v [%] ↑	P_{30} [%] ↑	P_{30}^{+} [%] ↑	$\widetilde{D}$ [px] ↓
DUMMY	–	–	–	–	–	8.0	–	79.0
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0100	55.2	59.0	63.4	-250.6	13.2
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0050	74.3	78.0	80.9	-49.4	8.8
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0010	79.6	81.7	84.6	-6.9	8.3
LED-P-100	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0005	**81.3**	**83.6**	**86.9**	**19.5**	**8.0**
BAS-100	$\mathcal{T}_\ell$	–	0.0000	79.0	82.7	85.2	0.0	8.2
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0100	50.1	55.5	57.5	-43.0	14.9
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0050	51.0	57.3	60.4	-30.8	14.3
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0010	**68.5**	**71.4**	**76.2**	**30.9**	**9.9**
LED-P-30	30%$\mathcal{T}_\ell$	30%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0005	61.7	66.3	70.5	8.6	11.1
BAS-30	30%$\mathcal{T}_\ell$	–	0.0000	59.0	66.2	68.3	0.0	10.9
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0100	18.3	28.7	25.1	-33.9	94.1
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0050	32.3	42.2	39.9	-5.1	61.0
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0010	**42.1**	**48.0**	**50.3**	**15.2**	**33.6**
LED-P-10	10%$\mathcal{T}_\ell$	10%$\mathcal{T}_\ell \cup \mathcal{T}_u$	0.0005	36.1	44.7	43.8	2.5	53.1
BAS-10	10%$\mathcal{T}_\ell$	–	0.0000	34.7	41.6	42.5	0.0	57.8
AE-P-512	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	–	0.5	1.1	3.7	–	137.9
AE-P-1024	$\mathcal{T}_\ell$	$\mathcal{T}_\ell \cup \mathcal{T}_u$	–	-0.9	-2.2	3.8	–	130.1
CLIP-16 [31]	$\mathcal{T}_\ell$	–	–	2.0	8.9	6.5	–	102.7
CLIP-512 [31]	$\mathcal{T}_\ell$	–	–	1.4	7.8	5.7	–	109.1
Frontnet [27]	$\mathcal{T}_\ell$	–	–	**69.5**	**74.1**	48.7	–	31.0
EDNN [4]	$\mathcal{T}_\ell$	$\mathcal{T}_s$	–	64.7	68.5	**73.3**	–	**10.2**
UB	$\mathcal{T}_\ell \cup \mathcal{T}_u^*$	–	0.0000	91.7	89.3	93.9	100.0	6.8



of view, thus making its LED state impossible to predict. We explore the approach robustness in an experiment where we add 3.4k such samples to $\mathcal{T}_u$, a 10% increase. After training on the modified $\mathcal{T}_u$, LED-P-30 ($\lambda = 0.001$) scores 10.3 in $\widetilde{D}$ and 73.2% in P_{30}, a small decline in performance when compared to the same model trained using the original $\mathcal{T}_u$.

In Figure 5, we show the P_{30}^{+} relative improvement score for BAS and LED-P strategies as the amount of labeled training examples and the weight of the loss λ vary. LED-P outperforms BAS when training on as few as 100 labeled examples (10% of $\mathcal{T}_\ell$). The optimal value of λ for our loss is 0.001 for 100 and 300 labeled examples (10% and 30% of $\mathcal{T}_\ell$ respectively), and 0.0005 for the full labeled dataset.

Inspecting the two loss term values, we note that $\mathcal{L}_{\text{task}}$ is in the order of magnitude of 10^{-4} and $\mathcal{L}_{\text{pretext}}$ in 10^{-1}. The optimal λ values are scaling the loss terms to be in the same order of magnitude, striking a balance between the two.

C. Alternative Training Strategies

We investigate strategies using a different architecture, model pre-training, or different pretext tasks, described in Section IV-C, and whose performance metrics are reported in Table I, fourth panel. EDNN scores 10.2 pixels in $\widetilde{D}$ demonstrating some degree of accuracy; however, it scores lower than LED-P-100 and BAS-100 that use the same amount of labeled data. This result suggests that task similarity is less influential than dataset relevance, i.e., training on the same task with data vastly different (in appearance) from testing achieves lower scores than solving a different task on similar data, e.g., our LED state prediction pretext task.

Frontnet achieves 31 pixels in $\widetilde{D}$ despite having been trained similarly to BAS-100, which achieves only 8.2 pixels. The increase in error demonstrated by Frontnet indicates that using a FCN model producing a map-based representation leads to a better performance than direct regression.

AE-P successfully learns to reconstruct input images using the bottleneck features. However, the model focuses on large-scale aspects of the environment, such as floors, walls and fixtures, distinctive of background elements and disregards high frequency elements such as the drone. This tendency results in a feature space, regardless of bottleneck size, that is *not informative* of the drone's position, rendering this pretext task inadequate for localizing small objects. Even in CLIP's case, we note how the provided features do not translate in good localization performance; this confirms previous reports [31] that CLIP's image encoder features underperform on highly specialized end tasks, such as ours.

AE-P and CLIP highlight the importance of having good features, which can be obtained by choosing the right kind of pretext task, and promoting the recognition of patterns similar to those required to solve the end task.

D. Generalization Ability

In Figure 6, we show the prediction of the observer drone using LED-P-30 ($\lambda = 0.001$) on images featuring multiple target drones collected in another lab environment [35], and synthetic frames from a simulator [4]. For this scenario, we modified the argmax approach by thresholding the localization map $\hat{\boldsymbol{\ell}}$ with its 95-percentile, extracting the maximum value of each connected component, discarding components whose maximum is below 0.2 and returning their centroid as the drone locations. For the most part, our model correctly localizes the drones despite the motion blur and defocus, with

Fig. 4. LED-P-30 with $\lambda = 0.001$ (small green circle), BAS-30 (yellow cross) predictions and ground truth (large magenta circle) on frames taken from $\mathcal{Q}$ with the drone's LEDs turned on (first three) and off (last three), and featuring different camera exposure settings.

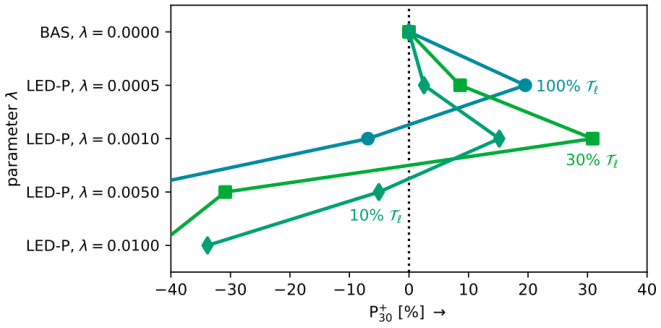


Fig. 5. P^+_{30} score for BAS ($\lambda = 1$) and LED-P ($\lambda < 1$) strategies as the amount of labeled training examples $\mathcal{T}_\ell$ and the weight of the loss λ vary.

all examples featuring at least two correct predictions. Failed detections on the edge of the field of view occur due a strong vignetting effect, which degrades the image quality.

E. In-field Experiment

The LED-P-30 ($\lambda = 0.001$) and BAS-30 models are deployed aboard the observer nano-drone, using the academic NEMO/DORY framework[5]. NEMO provides post-training quantization-aware fine-tuning, to convert deep learning models from floating-point to integer arithmetic, needed due to the absence of floating point units on the GAP8 SoC. DORY, instead, produces a template-based C implementation, which takes care of data movements across the memory hierarchy of the GAP8 SoC. This stage is fundamental in achieving fast inference, as sub-optimal data tiling/transfers might lead to poor performances. Our optimized NN pipeline achieves an in-field inference rate of 21 frames per second.

In the observer drone, the model output is used as feedback to a visual servoing controller, designed to keep the target drone in the center of the image. The controller moves the observer drone on a vertical plane, orthogonal to its camera axis, while keeping a constant yaw. Without loss of generality, we consider the motion of the observer drone to take place on the yz world plane, with $x = 0$. In the experiment, the target drone follows a predefined, scripted trajectory. The ideal trajectory for the observer drone is the same as the target, projected on $x = 0$ vertical world plane.

Figure 7 reports the y and z components of the measured trajectory of the observer drone, controlled using position estimates from LED-P-30, compared to the ideal one. We observe that the drone follows very closely the ideal trajectory. The same experiment run using BAS-30 yields worse position tracking: the mean and standard deviation σ of the absolute position error on the yz plane is 4.2 cm ($\sigma = 4.0$ cm) for LED-P-30, and 11.9 cm ($\sigma = 8.3$ cm) for BAS-30.

[5]https://github.com/pulp-platform/nemo

VI. Conclusions

We propose LED state estimation as a self-supervised pretext task, applied to the end task of visually localizing robots from small labeled datasets. The pretext task is optimized on large, cheaply-collected datasets that only have ground truth for the LED state of the observed robot. The approach is instantiated on localizing nano-quadrotors in low-resolution images, observing improved localization accuracy compared to baselines and alternative techniques for self-supervision. In-field experiments used a 27-g Crazyflie nano-drone to track the position of a peer drone; the proposed approach reduces mean position-tracking error from 11.9 to 4.2 cm. The resulting detector can be used, for example, within a tracking-by-detection approach [36] to integrate predictions over time and track the target even in presence of occlusions.

Current work aims at extending the approach to estimate the distance and orientation of the target, handling sequential inputs, and exploiting the known state of multiple LEDs as a localization cue at inference time.

References

[1] B. Taha and A. Shoufan, "Machine learning-based drone detection and classification: State-of-the-art in research," *IEEE Access*, vol. 7, pp. 138 669–138 682, 2019.

[2] M. Pavliv, F. Schiano, C. Reardon, D. Floreano, and G. Loianno, "Tracking and relative localization of drone swarms with a vision-based headset," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1455–1462, 2021.

[3] M. Ciani, S. Bonato, R. Psiakis, A. Garofalo, L. Valente, S. Sugumar, A. Giusti, D. Rossi, and D. Palossi, "Cyber security aboard micro aerial vehicles: An opentitan-based visual communication use case," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

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

[5] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[6] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, "Self-supervised monocular road detection in desert terrain," in *Robotics: Science and Systems*, 2006.

[7] A. Lookingbill, J. Rogers, D. Lieb, J. Curry, and S. Thrun, "Reverse optical flow for self-supervised adaptive autonomous robot navigation," *International Journal of Computer Vision*, vol. 74, pp. 287–302, 2006.

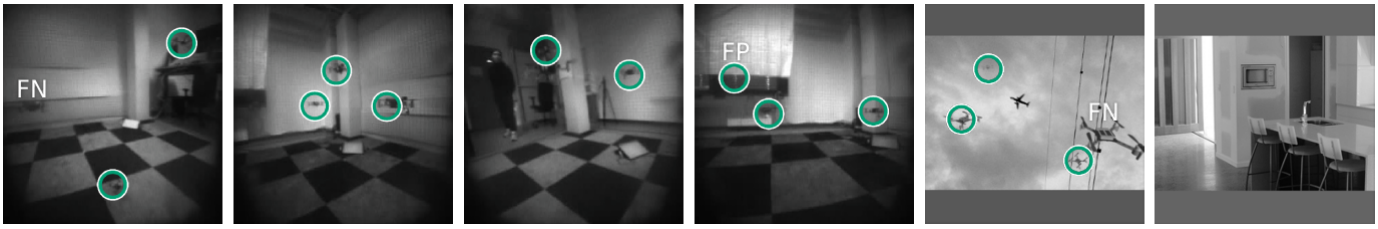


Fig. 6. Generalization and multi-drone localization examples using LED-P-30 ($\lambda = 0.001$). The first four images are taken from an unseen lab environment [35] and the last two from an unseen synthetic one [4]. Errors are visually marked as false positives (FP) or false negatives (FN).

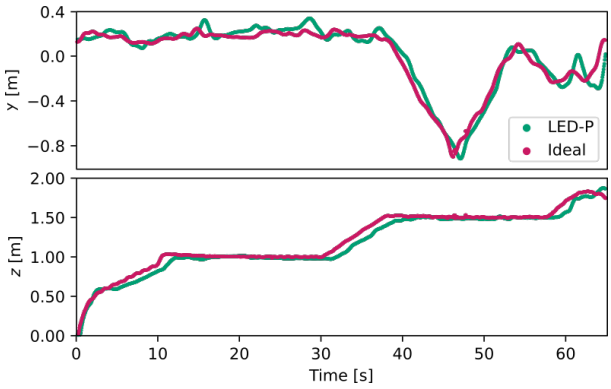


Fig. 7. In-field experiment: measured vs ideal trajectory of the observer drone when using LED-P-30 ($\lambda = 0.001$) for estimating the target position.

[8] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning Long-Range Vision for Autonomous Off-Road Driving," *Wiley Online Library Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[9] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1386–1383.

[10] M. Nava, A. Paolillo, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning visual localization of a quadrotor using its noise as self-supervision," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2218–2225, 2022.

[11] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, "Real-world robot learning with masked visual pre-training," in *PMLR Conference on Robot Learning*, 2023, pp. 416–426.

[12] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[13] M. Basiri, F. Schill, D. Floreano, and P. U. Lima, "Audio-based localization for swarms of micro air vehicles," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 4729–4734.

[14] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "3d relative positioning sensor for indoor flying robots," *Autonomous Robots*, vol. 33, no. 1, pp. 5–20, 2012.

[15] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar, "System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization," *Autonomous Robots*, vol. 41, no. 4, pp. 919–944, 2017.

[16] D. Dias, R. Ventura, P. Lima, and A. Martinoli, "On-board vision-based 3d relative localization system for multiple quadrotors," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1181–1187.

[17] M. Vrba, D. Heřt, and M. Saska, "Onboard marker-less detection and localization of non-cooperating drones for their safe interception by an autonomous aerial system," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3402–3409, 2019.

[18] A. Carrio, J. Tordesillas, S. Vemprala, S. Saripalli, P. Campoy, and J. P. How, "Onboard detection and localization of drones using depth maps," *IEEE Access*, vol. 8, pp. 30 480–30 490, 2020.

[19] S. Güler, M. Abdelkader, and J. S. Shamma, "Peer-to-peer relative localization of aerial robots with ultrawideband sensors," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 1981–1996, 2020.

[20] T.-M. Nguyen, Z. Qiu, T. H. Nguyen, M. Cao, and L. Xie, "Distance-based cooperative relative localization for leader-following control of mavs," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3641–3648, 2019.

[21] M. Coppola, K. N. McGuire, K. Y. Scheper, and G. C. de Croon, "On-board communication-based relative localization for collision avoidance in micro air vehicle teams," *Autonomous robots*, vol. 42, no. 8, pp. 1787–1805, 2018.

[22] S. Van Der Helm, M. Coppola, K. N. McGuire, and G. C. de Croon, "On-board range-based relative localization for micro air vehicles in indoor leader–follower flight," *Autonomous Robots*, vol. 44, no. 3, pp. 415–441, 2020.

[23] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, "Decentralized visual-inertial-uwb fusion for relative state estimation of aerial swarm," in *IEEE International Conference on Robotics and Automation*. IEEE, 2020, pp. 8776–8782.

[24] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *Robotics: Science and Systems*, 2018.

[25] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 3665–3671.

[26] M. Nava, A. Paolillo, J. Guzzi, L. M. Gambardella, and A. Giusti, "Uncertainty-aware self-supervised learning of spatial perception tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6693–6700, 2021.

[27] 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 *IEEE International Conference on Robotics and Automation*, 2023, pp. 3411–3417.

[28] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting flying objects using a single moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 879–892, 2016.

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

[30] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *IEEE Signals, Systems & Computers*, vol. 2, 2003, pp. 1398–1402.

[31] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *PMLR International Conference on Machine Learning*, 2021, pp. 8748–8763.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE/CVF International Conference on Computer Vision*, 2017, pp. 2980–2988.

[33] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[34] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 512–519.

[35] A. Moldagalieva and W. Hönig, "A dataset and comparative study for vision-based relative position estimation of multirotor teams flying in close proximity," *arXiv preprint*, p. 2303.03898, 2023.

[36] Y. Tian, A. Dehghan, and M. Shah, "On detection, data association and segmentation for multi-target tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2146–2160, 2018.