Delft University of Technology

# Self-supervised Monocular Multi-robot Relative Localization with Efficient Deep Neural Networks

Li, S.; de Wagter, C.; de Croon, G.C.H.E.

**DOI**
10.1109/ICRA46639.2022.9812150

**Publication date**
2022

**Document Version**
Final published version

**Published in**
2022 IEEE International Conference on Robotics and Automation, ICRA 2022

**Citation (APA)**
Li, S., de Wagter, C., & de Croon, G. C. H. E. (2022). Self-supervised Monocular Multi-robot Relative Localization with Efficient Deep Neural Networks. In G. J. Pappas, & V. Kumar (Eds.), *2022 IEEE International Conference on Robotics and Automation, ICRA 2022* (pp. 9689-9695). (Proceedings - IEEE International Conference on Robotics and Automation). https://doi.org/10.1109/ICRA46639.2022.9812150

**Important note**
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

**Copyright**
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

**Takedown policy**
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

*This work is downloaded from Delft University of Technology.*
*For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.*

***Green Open Access added to TU Delft Institutional Repository***

***'You share, we take care!' - Taverne project***

***https://www.openaccess.nl/en/you-share-we-take-care***

Otherwise as indicated in the copyright section: the publisher
is the copyright holder of this work and the author uses the
Dutch legislation to make this work public.

2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

# Self-supervised Monocular Multi-robot Relative Localization with Efficient Deep Neural Networks

Shushuai Li, Christophe De Wagter and Guido C. H. E. de Croon

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812150

***Abstract*— Relative localization is an important ability for multiple robots to perform cooperative tasks in GPS-denied environments. This paper presents a novel autonomous positioning framework for monocular relative localization of multiple tiny flying robots. This approach does not require any groundtruth data from external systems or manual labeling. Instead, the proposed framework is able to label real-world images with 3D relative positions between robots based on another onboard relative estimation technology, using ultra-wideband (UWB). After training in this self-supervised manner, the proposed deep neural network (DNN) can predict relative positions of peer robots by purely using a monocular camera. This deep learning-based visual relative localization is scalable, distributed, and autonomous. We also built an open-source and lightweight simulation pipeline by using Blender for 3D rendering, which allows synthetic image generation of other robots, and generalized training of the neural network. The proposed localization framework is tested on two real-world Crazyflie2 quadrotors by running the DNN on the onboard AIdeck (a tiny AI chip and monocular camera). All results demonstrate the effectiveness of the self-supervised multi-robot localization method. Video: `https://youtu.be/7arkaIblPps`**

## I. INTRODUCTION

Relative localization is necessary for a robot to interact with peer robots, underlying a wide range of distributed and cooperative tasks, e.g., formation flight [1], cooperative construction [2], flocking behaviour [3], etc. However, most of the multi-robot systems rely on external devices such as the global positioning system (GPS) or motion capture systems, for providing the relative positions between robots. These systems cannot work in unknown, GPS-denied environments.

Onboard relative localization methods have been recently proposed for achieving fully autonomous operation of multi-robot systems. Relative estimation based on sound [4] or infra red [5] is impractical for nano robots as larger sensor arrays need to be mounted. Relative localization with communication chips is very suitable for tiny robots thanks to their light weight. For example, multiple tiny quadrotors can avoid each other based on the received signal strength (RSS) [6]. More precise ranging from UWB can be implemented on the same tiny robots for more accurate relative estimation [7]. However, these methods suffer from band-width limitations, leading to poor scalability for a larger number of robots.

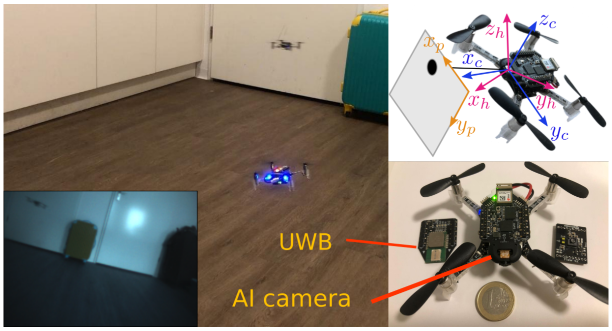


Fig. 1: Multiple tiny Crazyflie quadrotors localize peer robot 3D positions with deep neural networks based on self-supervised labels from an ultra-wideband relative localization method. Top right: different coordinate frames. Bottom right: a tiny quadrotor, the AI camera and the UWB module. Bottom left: onboard captured image.

Vision-based methods are scalable and distributed for multi-robot localization. These methods can be divided into two main categories: marker-based traditional methods and marker-less learning based detection. The methods with markers consist of relative localization for multiple micro aerial vehicles (MAVs) with onboard markers [8], [9], collaborative localization for a swarm of MAVs relying on salient external features for sparse reconstruction [10], estimation of relative pose between two ground robots by observing a pair of 3D points [11], and two drones tracking same target cooperatively [12]. The performance of these methods is easily degraded due to the size of markers (which should be very small for tiny robots) and the marker pose in the image. Although there is a swarm of quadrotors based on general textures by using visual inertial odometry (VIO) [13], the relative pose will drift with time and they must take off from known locations. An improvement for VIO-based relative localization is combining it with the UWB measurement [14]. However, the VIO part requires considerable computation power, which is impracticable for tiny flying robots.

Learning-based visual localization is marker-independent and directly robot-oriented. For example, 3D positions of a drone can be estimated by using depth images and deep neural networks [15], [16]. DeepURL proposes a deep estimation method for relative localization of underwater vehicles based on keypoint prediction and PnP which, however, requires each robot's 3D model information [17]. Another state-of-the-art work uses YOLOv3-tiny to detect the drones by training the network with mask images from a static camera and background subtraction method [18], in which the detection can be subject to a larger reality gap caused by more cluttered environments, motion blur and lighting conditions. In [19], the drone positions can be automatically annotated with localization sensor UVDAR, which lacks an efficient solution for deployment on tiny drones.

We also review the related references in computer vision and robotic grasping. Classical pose networks require

---

All authors are with Faculty of Aerospace Engineering, Delft University of Technology, 2629 HS Delft, The Netherlands (e-mail: s.li-6@tudelft.nl; c.deWagter@tudelft.nl; g.c.h.e.decroon@tudelft.nl)

manual annotations such as SSD [20], PoseCNN [21], and PoseNet [22]. Model-based methods can extract more pose information without or with less annotations. For example, EPOS predicts 3D fragment coordinates only with coordinate annotation, and then uses PnP-RANSAC to get 6D object pose [23]. Instead of using coordinate annotation, a deep neural network is designed by detecting 2D projections of robot joints, combined with PnP and model information to estimate the camera-to-robot pose from a single image [24]. However, annotations are time-consuming, and 3D model information is not significant for tiny robots.

Extended visual information can facilitate the 6D pose estimation, such as RGB-D images for object pose prediction, based on a model [25], or in a self-supervised way by capturing images from different views [26]. A pair of images is used for self-supervised depth estimation [27]. These extended visual sensors are usually heavy, power hungry, and high-cost for tiny robots compared with a monocular camera.

This paper proposes a self-supervised framework for autonomous, scalable and low-cost relative localization of multiple tiny robots. The proposed network draws from the object detection research YOLOv3, but is adapted to the multi-robot localization domain. We do not predict bounding boxes. Rather, we predict the 2D pixel position of the robot center and depth from camera to robot. Here we adopt our previous work of UWB-based relative localization as an auxiliary localization method to label the images automatically [7], to make the training process self-supervised. Notice that the UWB localization component is soft- and hardware open-source, and lightweight such that it can be used by other drones. The **contributions** of this paper are summarized as: 1) an efficient deep neural network for integrated monocular multi-robot detection and depth prediction; 2) a novel self-supervised system framework for data labeling and network training; 3) a lightweight 3D rendering simulation for multi-robot image generation with arbitrary pose states; 4) the first implementation of deep neural networks into light tiny (33 gram) flying robots for visual relative localization; 5) Public release of all code to the community [1].

The remainder of the paper is organized as follows. Section II introduces the system definition and the relative localization problem. Section III gives the detailed design of the proposed framework and the deep neural network. Section IV shows the 3D multi-robot visual rendering pipeline, and performance of the deep relative localization on synthetic images. Section V validates the localization efficacy on real-world experiments, including dataset collection, network refining, and deep inference onboard an AI camera.

## II. PRELIMINARIES

A monocular camera mounted on one robot can observe $n$ peer robots in a single RGB image. This section gives the preliminaries of the multi-robot visual system, the auxiliary onboard relative localization, and the problem definition.

### A. Multi-robot system

For clarity, the spatial model of two robots is considered. As shown in Fig. 1, we define three coordinates: 1) image coordinate with yellow axes, where $\boldsymbol{\xi}_p = [x_p, y_p, 1]^T$ denotes pixel positions of peer-robot center in the image with top-left origin; 2) camera coordinate with blue axes, where $\boldsymbol{\xi}_c = [x_c, y_c, z_c]^T$ represents 3D positions of peer robot with respect to the camera; 3) horizontal coordinate with red axes, which is an inertial frame fixed to the robot with a vertical $z$ axis, while $x$ and $y$ axes point forward and left horizontally.

Camera coordinates can be transformed to image coordinates by the intrinsic matrix $M_{\text{itr}}$ of the camera. The intrinsic parameters of the AIdeck camera is calibrated with a chessboard, and $R_c$ means the rotation of the camera coordinate, which are shown as follows:

$$x_c\boldsymbol{\xi}_p = M_{\text{itr}}R_c\boldsymbol{\xi}_c = \begin{bmatrix} 183.73 & 0 & 166.90 \\ 0 & 184.12 & 77.51 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \boldsymbol{\xi}_c \tag{1}$$

The auxiliary 3D relative estimation is represented in the horizontal coordinate, denoted as $\boldsymbol{\xi}_h = [x_h, y_h, z_h]^T$, which facilitates real-world 3D multi-robot control. This coordinate can be transformed into camera coordinate by rotation in xy sequence

$$\boldsymbol{\xi}_c = R(\phi,\theta)\boldsymbol{\xi}_h = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ s(\phi)s(\theta) & c(\phi) & -c(\theta)s(\phi) \\ -c(\phi)s(\theta) & s(\phi) & c(\theta)c(\phi) \end{bmatrix} \boldsymbol{\xi}_h, \tag{2}$$

where $\phi$ and $\theta$ denote roll and pitch attitude along axis $x_c$ and $y_c$ respectively.

### B. Onboard auxiliary localization

This subsection gives a brief review of the UWB-based relative estimation [7], which is used for generating labels to teach the deep neural network to learn peer-robot positions from monocular images.

As can be seen in Fig. 2, the yellow box illustrates the auxiliary localization. The $i^{\text{th}}$ robot takes as inputs the peer velocity $v_j$, peer yaw rate $r_j$, peer height $h_j$, self velocity $v_i$, self yaw rate $r_i$, self height $h_i$, and range $d_{ij}$. Afterwards, a Kalman filter is implemented to estimate the relative position $[x_{ij}, y_{ij}, h_{ij}]^T$, which is equal to $\boldsymbol{\xi_h}$. More details can be found in [7].

### C. Self-supervised localization problem

Suppose the multi-robot system is equipped with the aforementioned localization techniques. Each robot captures the monocular image and obtains the corresponding label $\boldsymbol{\xi}_h$ automatically. Given the image $Im$ with a peer robot in it, the self-supervised deep localization problem is to find a deep neural network $f_n(Im)$ that can predict peer-robot relative positions $\boldsymbol{Y}$ which satisfies $\boldsymbol{Y} = [x_p, y_p, x_c]^T$. According to (1) and (2), the desired relative positions $\boldsymbol{Y}$ can be derived from automatic label $\boldsymbol{\xi}_h$ as:

$$\begin{aligned} \boldsymbol{Y} &= F(R(\phi,\theta)\boldsymbol{\xi}_h) = [\boldsymbol{\xi}_p^{[0:2]}; x_c] \\ &= [(M_{\text{itr}}R_cR(\phi,\theta)\boldsymbol{\xi}_h/x_c)^{[0:2]}; x_c] \end{aligned} \tag{3}$$

---

[1] Code: `https://github.com/shushuai3/deepMulti-robot`

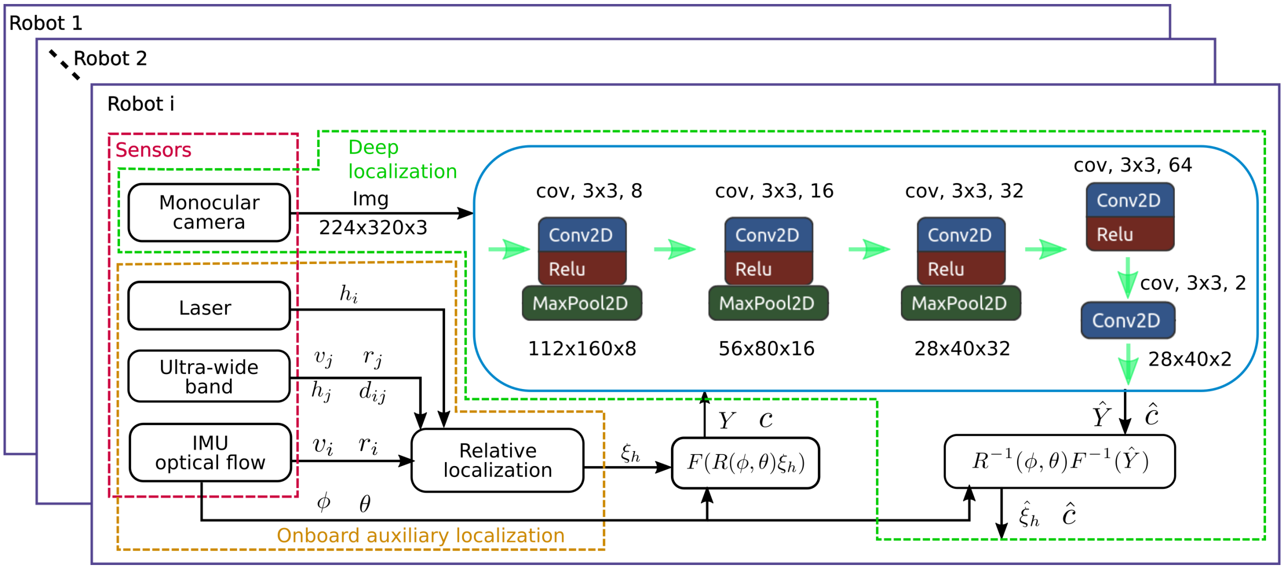

Robot 1
Robot 2
Robot i
Sensors
Deep localization
Monocular camera
Img
224x320x3
Laser
$h_i$
Ultra-wide band
$v_j$ $r_j$
$h_j$ $d_{ij}$
IMU optical flow
$v_i$ $r_i$
$\phi$ $\theta$
Relative localization
$\xi_h$
$F(R(\phi,\theta)\xi_h)$
$Y$ $c$
Onboard auxiliary localization
cov, 3x3, 8
Conv2D
Relu
MaxPool2D
112x160x8
cov, 3x3, 16
Conv2D
Relu
MaxPool2D
56x80x16
cov, 3x3, 32
Conv2D
Relu
MaxPool2D
28x40x32
cov, 3x3, 64
Conv2D
Relu
cov, 3x3, 2
Conv2D
28x40x2
$\hat{Y}$ $\hat{c}$
$R^{-1}(\phi,\theta)F^{-1}(\hat{Y})$
$\hat{\xi}_h$ $\hat{c}$

Fig. 2: System framework for deep relative localization of tiny flying robots. Red block shows all onboard sensors. Yellow block is the onboard auxiliary localization, which outputs the relative position label. This data is transformed to robot pixel position and depth for training the neural network as shown in the green block. After training, the deep localization can use purely monocular image to estimate the 3D multi-robot relative pose. In addition, the network architecture consists of convolution layers, Relu activation, and max pooling layers as shown in the blue block. It also shows the kernel size, input and output channels, and the image size for each step.

where $\boldsymbol{\xi}_p^{[0:2]}$ means the first two rows of the vector $\boldsymbol{\xi}_p$. In addition, multiple robots can appear on one monocular image. Thus, the deep inference function $f_n(Im)$ must be able to predict 3D positions of multiple robots.

## III. METHODOLOGY

This section describes the detailed approach to self-supervised monocular relative localization. The system framework, network architecture and loss functions are explained, respectively.

### A. Network output and self-supervised dataset

The data flow of the whole system is shown in Fig. 2. As can be seen, it starts from the auxiliary localization block, which generates the labeled dataset automatically for self-supervised training. The dataset consists of monocular images and the corresponding inter-robot positions $\boldsymbol{Y} = [x_p, y_p, x_c]^T$. To detect multiple robots, the network output $f_n(Im)$ is designed as a 28x40 grid map with the predicted depth channel $\hat{d}(i,j)$ and the confidence channel $\hat{c}(i,j)$. A higher dimensional grid map leads to more accurate pixel localization, which however increases the ambiguity between detections at neighbor pixels. The grid labels are created by the following rules:

$$\begin{cases} c(i,j)=1, d(i,j)=x_c, & \mathrm{if(i,j)} = (\mathrm{x_p}/8, \mathrm{y_p}/8) \\ c(i,j)=0, d(i,j)=0, & \mathrm{otherwise} \end{cases} \tag{4}$$

which means a grid that contains a robot has confidence of 1 and the depth of $x_c$ in camera coordinate. Since $x_p \in [0, 320)$ and $y_p \in [0, 224)$, $(x_p, y_p)/8$ fits with the network output size.

In order to obtain a more generalized network, we pretrain the network on a synthetic dataset. In this dataset, the grid labels can be created automatically by the masks of robots during 3D rendering. The synthetic dataset contains more different backgrounds and arbitrary random attitudes and positions of the drones.

### B. Network architecture

Our deep network is inspired by YOLOv3 network which can detect multiple small objects with bounding boxes [28]. We modify the YOLOv3 network to solve the localization problem as described in Section II-C. The proposed network predicts the pixel position and depth of peer robots as can be seen in the blue block in Fig. 2, instead of bounding boxes as the original YOLOv3 network. The feature maps and layers of the original network are largely reduced, as we only predict one class of robots. Also, this simplified network fits with the implementation on a resource-limited tiny AI chip, the GAP8 microprocessor, which was first demonstrated for autonomous corridor following in [29].

Specifically, the proposed network is an encoder with eight main layers including convolution and max pooling. The activation adopts the Relu function, and there is batch normalization during the training process. No anchors are required as it focuses on the center of the object. The output layers are modified to predict the confidence grid map for localizing the robot center, and the depth grid map.

### C. Loss functions

The loss functions are also different from those in object detection networks. The total loss of the proposed network is composed by two individual items:

$$l = l_{\mathrm{d}} + l_{\mathrm{c}} \tag{5}$$

Depth loss $l_{\mathrm{d}}$ denotes the mean of square errors between estimated depth $\hat{x}_c$ and real value $x_c$ in grid $(i,j)$, which is represented by

$$l_{\mathrm{d}} = \mathrm{mean}(\sum_{\mathrm{i=1}}^{\mathrm{N_{yc}}}\sum_{\mathrm{j=1}}^{\mathrm{N_{xc}}}\mathrm{c(i,j)}(\hat{\mathrm{d}}_{\mathrm{c}}\mathrm{(i,j)} - \mathrm{d(i,j)})^2) \quad (6)$$

where $c(i,j)$ is the real confidence of whether there exists an robot center in grid $(i,j)$ defined in (4). $N_{yc} = 28$ and $N_{xc} = 40$ are the sizes of output grid maps in the proposed network.

Confidence loss item $l_{\mathrm{c}}$ is designed by softmax cross entropy function [28], and the formula is

$$l_{\mathrm{c}} = \mathrm{mean}(\sum_{\mathrm{i=1}}^{\mathrm{N_{yc}}}\sum_{\mathrm{j=1}}^{\mathrm{N_{xc}}}(\mathrm{c} - \hat{\mathrm{c}})^2[-\mathrm{c}\cdot\log(\hat{\mathrm{c}}) - (1-\mathrm{c})\log(1-\hat{\mathrm{c}})]) \quad (7)$$

where $\hat{c}(i,j)$ and $c(i,j)$ are the predicted and real confidence in grid $(i,j)$ defined in (4).

An optional loss item is to distinguish multiple classes of robots by using the one-hot vectors. $l_{\mathrm{prob}}$ demonstrates the class probability error, written as

$$l_{\mathrm{p}} = \mathrm{mean}(\sum_{\mathrm{i=1}}^{\mathrm{N_{yc}}}\sum_{\mathrm{j=1}}^{\mathrm{N_{xc}}}\mathrm{c(i,j)}[-\mathrm{p}\cdot\log(\hat{\mathrm{p}}) - (1-\mathrm{p})\log(1-\hat{\mathrm{p}})]) \quad (8)$$

where $\hat{p}(i,j)$ and $p(i,j)$ are predicted and real probability of different classes in grid $(i,j)$. This item has been tested to be effective for robot classification, but not included in this paper for a better quantization of the network to run in the microprocessor.

### D. Training and post processing

The proposed deep relative localization network is trained from scratch in the environments of Anaconda and Tensorflow2. There are 25 epochs for total training including 2 warm epochs to reduce the primacy effect and avoid the early over-fitting. Given 800 training images and a batch size of five images, each epoch has 160 steps. The learning rate changes adaptively with Adam method. The training can be either run on GPU or CPU as the network size is very small.

After training the network on synthetic images, the network is refined on the real-world dataset (192 training images) captured on two Crazyflies. The refined model file is quantized into int8 format by the GAP8 tool chains. Finally, it is compiled to a C function that can run on the GAP8 microprocessor.

Based on the prediction $\hat{c}(i,j)$ and $\hat{d}(i,j)$ from the network $f_n(Im)$, the final relative position $\boldsymbol{Y}$ can be calculated by

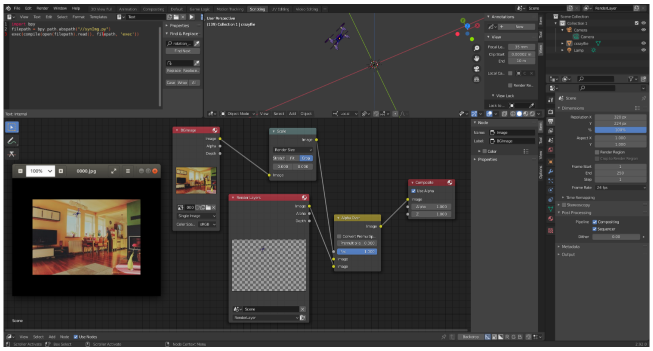
$$[\hat{x}_p, \hat{y}_p, \hat{x}_c]^{\mathrm{T}} = \{[8i, 8j, \hat{d}(i,j)]^{\mathrm{T}})|\hat{c}(i,j) > T_c\}. \quad (9)$$

where $T_c$ denotes a proper confidence threshold, which was empirically selected as 0.33 for synthetic dataset and 0.23 for real-world tests.

## IV. SIMULATION

This section shows the developed simulator for synthetic image generation, training and testing results of the network on the synthetic dataset. This simulated environment is not necessary for an onboard deep neural network, but it can generate a flexible and rich multi-robot visual dataset easily for preliminary validation of the DNN. Refining the network on the simulated dataset saves training time and promises more generality of the neural network.

### A. Simulated pipeline for 3D multi-robot rendering

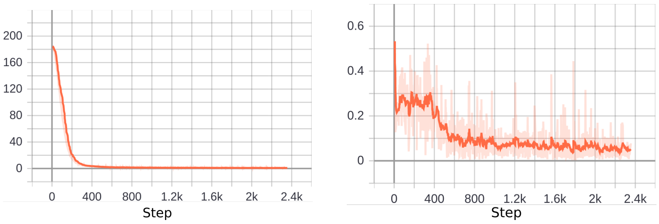


```
import bpy
filepath = bpy.path.abspath("//synImg.py")
exec(compile(open(filepath).read(), filepath, 'exec'))
```

Fig. 3: The developed 3D rendering simulation environment. It can render multiple 3D robots in any background images. The attitudes and positions of robots and camera can be set in python code.

We built a Blender-based 3D rendering environment as shown in Fig. 3 for generating synthetic images in a multi-robot domain. The whole pipeline can render images with a random number of 3D robots in the images with random attitude and positions. The attitude of the camera, and lighting conditions can be also set arbitrarily. The operational relative depth is randomized from 0.2m to 2m to make the drone feasible in the camera. The background images are from the COCO dataset, specifically the 2017-Val-images set including 5000 images. Annotation of robot positions and depth to the camera can be obtained from the prior known groundtruth in Blender. The rendered examples can be seen in Fig. 5. This tool is lightweight, open-source and easily modified for generating multi-robot images for other applications.

### B. Training on synthetic dataset



(a) Confidence loss (b) Depth loss (meter)

Fig. 4: Loss changes with training steps on synthetic dataset. The network is trained from scratch on a training dataset with 800 images. Two individual loss items are shown in this figure.

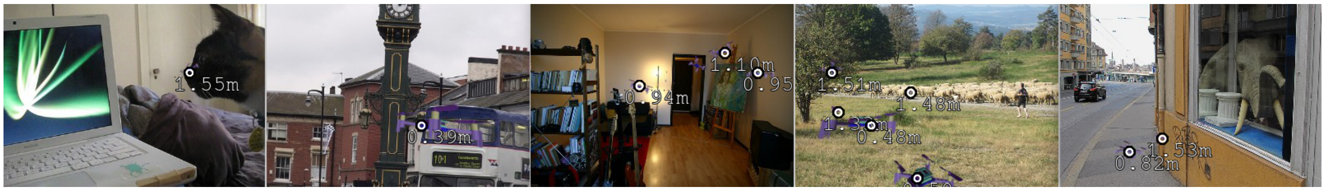


Fig. 5: Testing results of multi-robot localization on synthetic images. The white circle with an outer black circle represents the predicted robot position in the image, while the value means the estimated robot depth in the camera. Different robot attitude and position with respect to the camera are demonstrated in these figures, as well as multiple robots in indoor and outdoor environments.

From Fig. 4, we can see all loss items decrease as the training steps increase. They are stable at about 1000 steps. Specifically, confidence error tends to converge to zero which indicates that the deep neural networks encode the robot pixel position in the output grid maps. The stable depth error is about 0.05m, which is not approximating zero probably because different robot attitudes lead to slight depth changes. However, it is accurate enough for robot position estimation.

### C. Testing results on synthetic images

The prediction errors are shown in this subsection. All testing results come from network prediction on the test dataset which contains 200 new images with different backgrounds and different poses of peer robots.

From 5, we can see the proposed network can detect different sizes of robots in outdoor and indoor environments. In addition, thanks to the underlying (modified) YOLOv3 framework, the network is capable to detect multiple robots in one image at same time, even though it is only trained on images with single robot. There are a false-positive detection on the fourth image and a true-negative detection on the third image, potentially due to similar background and less features. These outliers can be rejected by filters in real-time sequences of images. Note that these five test images are selected randomly. Hence, the deep network has similar effectiveness on the other testing images.

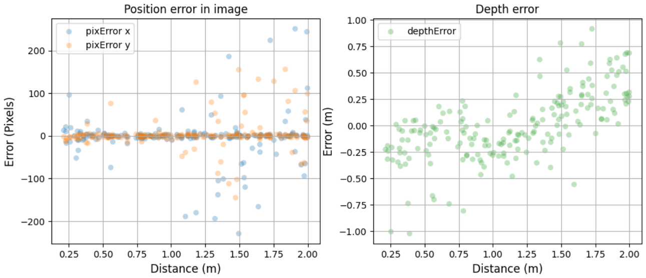


Fig. 6: 2D robot position error (left figure) in image and depth error (right figure) between prediction and groundtruth are shown with respect to inter-robot distances. This distribution is based on 200 testing images with single robot in each image.

For explicit demonstration, the statistical 3D estimation error is depicted by Fig. 6. From the left figure, we can see the robot localization prediction in image has a zero average pixel error. Most position error is within 20 pixels and with few outliers caused by a few false detections. The right figure demonstrates the error distribution of depth predictions, where mean and medium values are approximating zero. Both errors get larger as the distance between drones increase since the appearance is not significantly changed when robots are far away. However, only close neighbors are considered for swarm robotics.

Compared to other state-of-the-art research in relative localization with deep learning, our proposed network is much smaller and thus efficient for both training (20 minutes on i7 CPU) and testing. Besides, the drone depth is predicted simultaneously with the drone detection, while other references require the prior-known drone size to estimate robot depth.

## V. EXPERIMENTS

This section presents practical experiments: flight for real-world dataset collection, refining of the neural network on the real dataset, porting the Tensorflow-based network to AIdeck microprocessor, and onboard deep visual localization.

### A. Hardware

The multi-robot platform is composed by two Crazyflie2 quadrotors, which are tiny flying robots with only 33 grams and pocket size (12.5 cm in diameter). Three decks are required for our work. The first one is AI deck [29], which is composed by a GAP8 RISC-V processor, a Himax HM01B0 RGB camera, 512 Mbit HyperFlash for storing dataset, and an ESP32 wifi module for remote streaming. Another deck is the optical flow sensor for estimating robot velocity and altitude. The last one is the loco deck with a UWB module for inter-robot ranging measurements and auxiliary localization.

The last two decks are used for generating relative position labels automatically, while only the first deck is used for deep visual multi-robot localization.

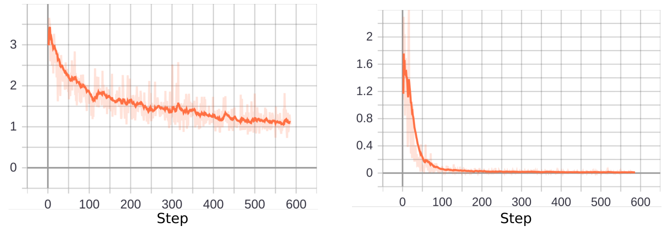
### B. Real-world dataset acquisition

Due to the reality gap between synthetic and real images, a real-world dataset with 240 images is collected for refining the network. During data collection, one drone flies with randomly velocities by a remote controller, while another drone flies with random velocities but in the view of the camera on the first drone. The example dataset can be found in following sections.

To get a dataset with two flying tiny drones, a specific procedure is designed: a quadrotor sends its 2D attitude (roll and pitch) and UWB-based 3D auxiliary relative position to the GAP8 chip, which combines these variables with the monocular image data and stores them in HyperFlash memory. Afterwards, the GAP8 sends the data from flash memory to a computer via a Olimex ARM-JTAG-20-10 debugger.

### C. *Refining and testing on real-world dataset*

The training process on the real-world dataset is the same as that in Section IV.



(a) Confidence loss (b) Depth loss (meter)

Fig. 7: Loss changes of refining the neural network on real-world dataset. The initial weights are those trained by the synthetic images in last section. All training configurations remain the same during refining.

From Fig. 7, we can see all loss items drop down again within 600 steps. The confidence loss decreases a bit slowly than that of depth loss, because the appearance changes largely while drone sizes are easily to learn. The real-world dataset is divided into 192 training images and 48 testing images with self-supervised labels of relative position, all obtained from two randomly flying Crazyflies, without relying on any external systems such as motion capture system or GPS.

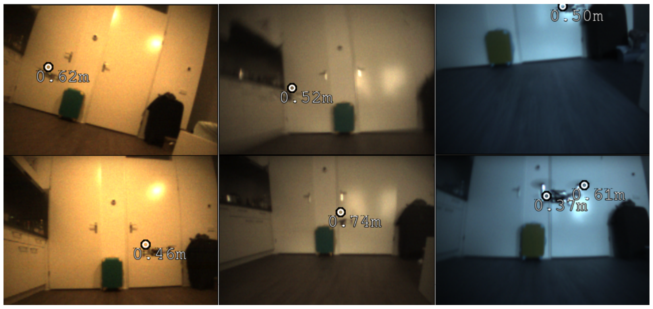


Fig. 8: Testing results of the refined network on randomly selected testing images. The white circle in images shows the predicted pixel position of peer robot center. The value on the image means the depth of the robot from the camera. All testing images are captured onboard with different flying attitude and velocity of both quadrotor, captured at different day times.

Fig. 8 shows the localization performance of the refined neural network. Both training and testing images have three light conditions (evening, afternoon and morning, respectively). These testing results indicate that the refined network can localize the real-world flying robots with high accuracy due to the previous training on a more generalized synthetic dataset. It also works even with large motion blur, partial occlusion, and different positions and attitudes of the robots.

Fig. 9 demonstrates the statistical prediction errors in image coordinate, by comparing the deep visual localization with the auxiliary UWB localization. The refined network has even smaller localization and depth prediction errors than on the synthetic dataset.

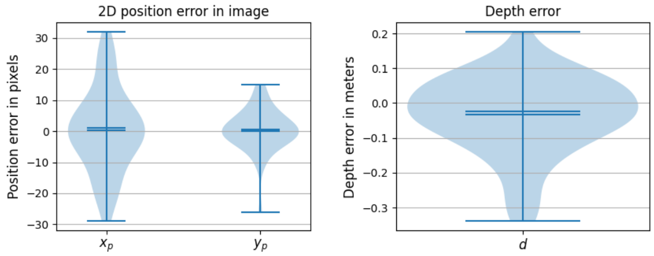


Fig. 9: Left: the 2D pixel position error of robot center between deep inference and UWB localization. Right: the depth error distribution in camera coordinate. These results are based on testing dataset with 48 new real-world images with size of 224x320.

### D. *Onboard deep relative localization with AIdeck*

This subsection shows the implementation of the deep localization network into a low-cost and ultra low-power AI chip. After quantization from float to int8, the network can run on the AIdeck microprocessor to predict peer-robot position onboard a tiny flying robot.

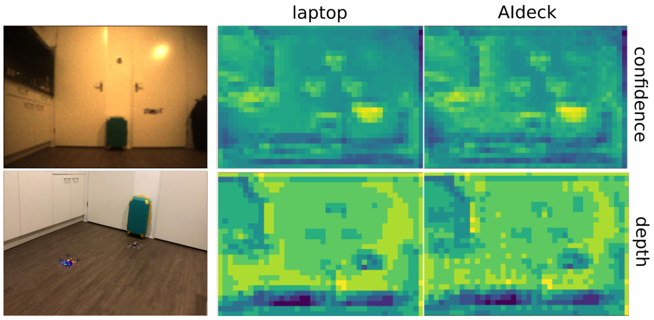


Fig. 10: Comparison of the deep visual relative localization between running on the laptop and the onboard AI chip. Top left: an example captured image during flight; bottom left: the experimental flight of two quadrotors; top middle: confidence channel predicted on laptop; bottom middle: depth channel predicted on laptop; top right: confidence channel predicted on AIdeck; bottom right: depth channel predicted on AIdeck.

Fig. 10 compares the deep inference results on both laptop (the middle column) and the edge AI chip (the right column) for the same flight image (top left). The network outputs on PC multiply the quantization scale calculated by the GAP8 tools. Two right images on first row shows the confidence prediction, where both laptop and AIdeck have similar inference of drone localization in the image. Though they have slight differences in depth prediction as seen in the right two images on second row, the depth values with respect to the robot center area are similar.

## VI. CONCLUSIONS

This paper proposes a novel monocular multi-robot relative localization framework which are self-supervised, autonomous, low-cost, and scalable, without using any external positioning system. In addition, this paper solves a challenging problem, i.e., implementation of deep learning into a tiny AI chip for relative localization of tiny flying robots.

Future work could include real-world dataset collection in more scenes, and control of a large number of flying robots with the proposed localization method.

# REFERENCES

[1] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[2] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," *Proc. Robotics: Science & Systems VII*, 2011.

[3] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, 2018.

[4] M. Basiri, F. Schill, D. Floreano, and P. U. Lima, "Audio-based localization for swarms of micro air vehicles," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 4729–4734.

[5] 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. 1-2, pp. 5–20, 2012.

[6] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, and G. C. H. E. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, 2019.

[7] S. Li, M. Coppola, C. De Wagter, and G. C. H. E. de Croon, "An autonomous swarm of micro flying robots with range-based relative localization," *arXiv preprint arXiv:2003.05853*, 2020.

[8] 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.

[9] I. Rekleitis, P. Babin, A. DePriest, S. Das, O. Falardeau, O. Dugas, and P. Giguere, "Experiments in quadrotor formation flying using on-board relative localization," in *Vision-based Control and Navigation of Small, Lightweight UAVs Workshop, IROS*, 2015.

[10] S. Vemprala and S. Saripalli, "Monocular vision based collaborative localization for micro aerial vehicle swarms," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 315–323.

[11] R. T. Rodrigues, P. Miraldo, D. V. Dimarogonas, and A. P. Aguiar, "A framework for depth estimation and relative localization of ground robots using computer vision," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3719–3724.

[12] E. Price, G. Lawless, R. Ludwig, I. Martinovic, H. H. Bülthoff, M. J. Black, and A. Ahmad, "Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3193–3200, 2018.

[13] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1801–1807, 2018.

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

[15] 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)*. IEEE, 2018, pp. 1034–1037.

[16] T. Hinzmann and R. Siegwart, "Deep uav localization with reference view rendering," *arXiv preprint arXiv:2008.04619*, 2020.

[17] B. Joshi, M. Modasshir, T. Manderson, H. Damron, M. Xanthidis, A. Q. Li, I. Rekleitis, and G. Dudek, "Deepurl: Deep pose estimation framework for underwater relative localization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1777–1784.

[18] 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.

[19] V. Walter, M. Vrba, and M. Saska, "On training datasets for machine learning-based visual relative localization of micro-scale uavs," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 674–10 680.

[20] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1521–1529.

[21] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[22] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[23] T. Hodan, D. Barath, and J. Matas, "Epos: estimating 6d pose of objects with symmetries," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 703–11 712.

[24] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-robot pose estimation from a single image," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9426–9432.

[25] K. Wada, E. Sucar, S. James, D. Lenton, and A. J. Davison, "Morefusion: multi-object reasoning for 6d pose estimation from volumetric fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14 540–14 549.

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

[27] J. Hur and S. Roth, "Self-supervised monocular scene flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7396–7405.

[28] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[29] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.