Vision-based Drone Flocking in Outdoor Environments

Fabian Schilling, Fabrizio Schiano, and Dario Floreano

***Abstract*—Decentralized deployment of drone swarms usually relies on inter-agent communication or visual markers that are mounted on the vehicles to simplify their mutual detection. This letter proposes a vision-based detection and tracking algorithm that enables groups of drones to navigate without communication or visual markers. We employ a convolutional neural network to detect and localize nearby agents onboard the quadcopters in real-time. Rather than manually labeling a dataset, we automatically annotate images to train the neural network using background subtraction by systematically flying a quadcopter in front of a static camera. We use a multi-agent state tracker to estimate the relative positions and velocities of nearby agents, which are subsequently fed to a flocking algorithm for high-level control. The drones are equipped with multiple cameras to provide omnidirectional visual inputs. The camera setup ensures the safety of the flock by avoiding blind spots regardless of the agent configuration. We evaluate the approach with a group of three real quadcopters that are controlled using the proposed vision-based flocking algorithm. The results show that the drones can safely navigate in an outdoor environment despite substantial background clutter and difficult lighting conditions. The source code, image dataset, and trained detection model are available at `https://github.com/lis-epfl/vswarm`.**

***Index Terms*—Aerial Systems: Perception and Autonomy, Multi-Robot Systems, Sensor-based Control**

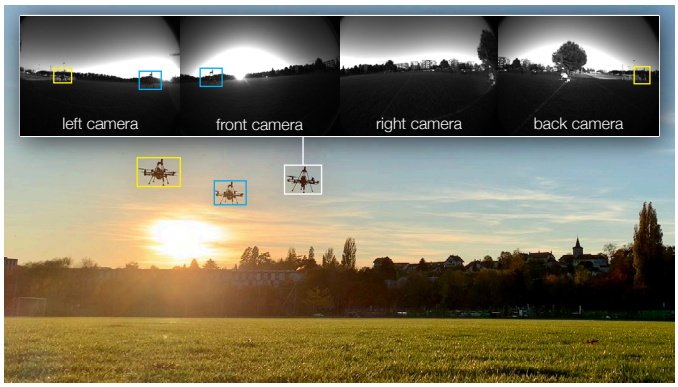


Figure 1: Photo taken during outdoor experiments with detection annotations. The quadcopters avoid collisions with each other and remain cohesive as a group while performing a variety of navigation tasks. Each agent detects its neighbors in real-time from omnidirectional images. There is no communication of state information between agents, and relative positions and velocities are estimated onboard using local visual inputs.

I. Introduction

DRONE SWARMS have a large socio-economic potential and can serve in a variety of real-world applications [1]. For example, drones can be leveraged to automatically monitor crops, safely inspect confined spaces, or quickly deliver medicine to inaccessible locations. Operating these vehicles in swarms could bring increased robustness to failures, larger area coverage, and faster task completion times [2].

Despite this potential, decentralized control has been a limiting factor in the deployment of drone swarms. For instance, large groups of quadcopters can be used to perform awe-inspiring aerial choreographies in the night sky. These robotic light shows are a true feat of engineering but individual drones are far from autonomous: their motion is centrally controlled by a ground computer that precomputes their trajectories and continuously monitors their positions. Hence, the ground computer represents a single point of failure. Researchers attempted to remove the central computer by equipping drones with hardware that allows them to wirelessly communicate with each other. Notable examples of these decentralized swarms feature bearing-only formation control [3], exploration of unknown environments [4], as well as flocking with ten fixed-wing drones [5] or thirty quadcopters [6].

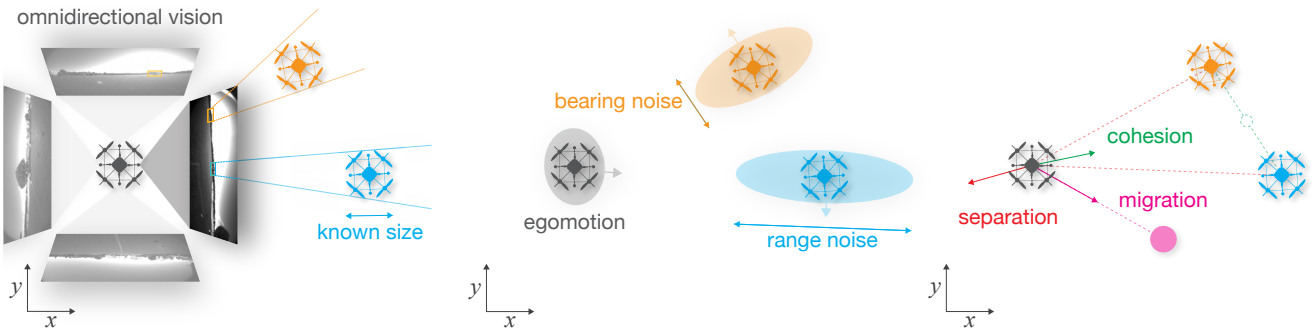
Scaling up decentralized drone swarms is complicated by the limitations of wireless communication. As the number of robots increases, the communication channels may become saturated and possibly jammed since the data transfer volume scales quadratically with the robot count [7]. Frequent retransmissions of messages can lead to delays that render the control of each drone extremely difficult. Researchers have experimented with different sensory modalities such as sound [8], but vision seems to be the most scalable approach to address the relative localization problem. Indeed, there is evidence to support that vision is the primary sensory modality that enables collective motion in animal groups [9].

However, visual detection of outdoor flying drones is challenging because they are relatively small and can fly in environments with large amounts of background clutter and difficult lighting conditions [10]. Additionally, the visual detection algorithm must run onboard the drones, whose own motion may generate motion blur and whose small dimensions and lightweight mass can restrict computational capabilities. To simplify the relative localization problem, researchers have mounted different types of easily detectable visual markers on the drones [11], [12]. However, this approach is less general since specialized hardware has to be mounted on each drone, which can result in increased weight and drag, thus reducing the energetic autonomy of the drones.

In recent years, markerless detection of drones has become

Manuscript received: December 10, 2020; accepted February 2, 2021. This letter was recommended for publication by Editor P. Pounds upon evaluation of the reviewer comments. This work was supported by the Swiss National Science Foundation with grant number 200021-155907 (*Corresponding author: Fabian Schilling*).

The authors are with the Laboratory of Intelligent Systems, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland (e-mail: fabian.schilling@epfl.ch; fabrizio.schiano@epfl.ch; dario.floreano@epfl.ch).



(a) Real-time drone detection (b) Multi-agent state tracking (c) Potential-field-based control

Figure 2: Overview of the processing steps of the vision-based flocking algorithm for a single time step: a) *detection*, b) *tracking*, and c) *control*. Step 1 (*detection*): we detect neighboring agents from omnidirectional images to estimate their relative range and bearing from the camera intrinsics and the drone's known physical size. Step 2 (*tracking*): we track the positions and velocities of the detected agents using a linearized observation model of range and bearing, as well as a state transition model that takes the focal agent's egomotion into account. Step 3 (*control*): we control the focal agent using a Reynolds-rules-based flocking algorithm that keeps the swarm collision-free and cohesive while following a navigation goal.

an active research topic. In [13], the authors use a boosted cascade of classifiers in combination with visual tracking and finite set filtering to estimate the positions and velocities of markerless drones from a mostly static observer. However, the method is applied in post-processing and not validated in a sense-and-avoid setting. In [14], the authors propose a combination of stereo vision and convolutional detection to enable leader-follower flight. However, the to-be-localized agent is always visible in front of the clear sky which simplifies the detection problem and would be impossible to guarantee in a self-organized flock. Moreover, the limited field of view and processing delays in the proposed system are known to be problematic when flying in dense swarms [6], [15]. Other notable examples of markerless detection include approaches based on template matching and morphological filtering [16], as well as convolutional neural networks [17].

Our previous work [18] proposes a fundamentally different approach to visual flocking based on imitation learning. Rather than detecting neighboring agents, we predict flocking algorithm commands directly from omnidirectional visual inputs which allow the drones to remain collision-free and cohesive. The approach is validated with leader-follower experiments in an indoor environment but its reliance on end-to-end learning means that the entire monolithic neural network has to be retrained each time the task and/or visual appearance of the drones change. Adopting a more modular approach can be beneficial since the flocking algorithm may easily be exchanged for another task-dependent controller, and only the detector would have to be retrained for different drone appearances or environmental conditions. In all of the above cases, the algorithms are validated on a single agent and not in a multi-robot control setting.

Here, we propose a modular detection and tracking algorithm that enables collision-free and cohesive navigation for drone swarms. We automatically label images of drones using background subtraction to generate a dataset for the drone detector. We show that the detector can localize other drones in the presence of background clutter from onboard a flying drone despite being trained on images from a static camera. We assess the method with a dense group of three real quadcopters that flock in planar configurations in an outdoor environment with difficult lighting conditions. The omnidirectional camera configuration of the drone is specifically designed to enable safe operation in swarms regardless of the agent configuration. The overall proposed flocking algorithm is modular since each component, i.e. detection, tracking, and control, is self-contained and can be evaluated independently. To the best of our knowledge, this is the first entirely vision-based flock that does not depend on visual markers to simplify mutual detections.

II. Method

The proposed approach to vision-based flocking can be divided into the following steps: *detection*, *tracking*, and *control* (Fig. 2). Firstly, the *detection* module (Fig. 2a) takes grayscale images from an omnidirectional camera setup as inputs and outputs bounding boxes of nearby drones in real-time (Sec. II-C). Secondly, the *tracking* module (Fig. 2b) transforms the bounding boxes into range and bearing measurements using the known dimensions of the drones. Their relative positions and velocities are then estimated from the noisy measurements using a multi-agent state tracker (Sec. II-B). Finally, the *control* module (Fig. 2c) applies a flocking algorithm to the relative positions to obtain high-level control commands that keep the drones collision-free and cohesive (Sec. II-C). In the tracking and control steps, we assume that the agents are moving on a horizontal plane. We mainly introduce this constraint to be able to attribute their mutual repulsion to the flocking algorithm and avoid the effects of physical downwash that may be caused by nearby agents.

A. Real-time monocular drone detection

1) Automatic drone labeling with background subtraction: We use background subtraction to automatically generate a

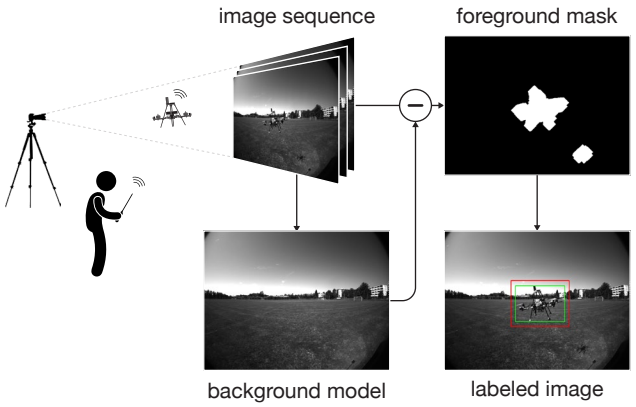


Figure 3: Schematic overview of the automatic dataset generation process. The foreground mask is generated using background subtraction. We find that the most precise bounding box labels are obtained by dilating the foreground mask since it eliminates discontinuities that occur due to the mechanical design of the drone. Enclosing the dilated mask with a bounding box (red rectangle) overestimates the size of the drone. We therefore scale the bounding box down (green rectangle) to obtain a precise label. We record six flights of roughly one-minute duration in the above target environment.

Figure 4: Example images with bounding box annotations from the dataset generated using background subtraction. We record image data from a static camera in both indoor (top row) and outdoor (bottom row) environments. The environments are selected to maximize the variety of background clutter and lighting conditions. We record three flights of roughly one-minute duration in each of the above environments.

labeled image dataset for the object detector (Fig. 3). We record images from a stationary camera mounted on a tripod and manually fly a quadcopter within its field of view. The image data is recorded under varying lighting conditions in both indoor and outdoor environments that contain large amounts of background clutter (Fig. 4). For each scene, the camera location and orientation are carefully chosen such that the quadcopter is the dominant source of motion. The final dataset consists of 9 891 training and 1 931 testing examples.

In post-processing, we apply a nearest neighbor background subtraction algorithm [19] to the sequence of images to learn a background model of the scene. We extract a foreground mask of the moving parts of the image (and therefore the quadcopter) by computing the element-wise difference of the input image and the background, followed by a thresholding step. The ground truth label is obtained by filtering out the largest contour present in the foreground mask and enclosing it with an axis-aligned rectangular bounding box.

2) Training the real-time drone detector: We train a single-stage convolutional object detector on the automatically annotated image dataset to obtain drone detections. We opt for the YOLOv3-tiny architecture [20] due to its favorable tradeoff between detection accuracy and inference speed on embedded devices. The network architecture is comprised of a total of 13 convolutional layers that are interspersed with max-pooling operations and leaky rectified linear units (ReLUs). The final detections are computed independently at two different scales and subsequently filtered by non-maximum suppression to account for bounding box overlaps.

We make a few notable modifications to the training procedure of the original publication [20]. Firstly, we replace the mean-squared error localization loss with an objective that is based on the generalized intersection over union (GIoU) [21] since it provides scale invariance. Secondly, we employ two recently popularized data augmentation techniques to improve detection accuracy: 1) multi-scale training, and 2) mosaic augmentation [22]. In multi-scale training, each training batch is scaled at random by up to $\pm 50\%$ of its side length to make the detector invariant to object scales. Mosaic training refers to concatenating four random training samples along their spatial dimensions to obtain an image collage. The resulting four-image mosaic is subsequently cropped randomly in the center to obtain a new training sample. We use holdout cross-validation to find suitable values for the most critical hyperparameters.

The parameters of the network are initialized with weights that are pre-trained on the COCO [23] dataset. The detector is then fine-tuned on a single *drone* class with stochastic gradient descent and Nesterov momentum of $\mu = 0.937$. We modulate the learning rate using a cosine annealing schedule with an initial learning rate $\eta = 0.01$ and a final learning rate $\eta_f = 0.0005$. We also employ a weight decay term of $\lambda = 0.0005$. The total loss is computed as

$$\mathcal{L} = k^{\text{bal}} \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{obj}} \tag{1}$$

where $\mathcal{L}_{\text{loc}}$ denotes the GIoU-based localization loss and $\mathcal{L}_{\text{obj}}$ refers to the binary cross-entropy objectness loss. We set the hyperparameter $k^{\text{bal}} = 0.055$ to balance the losses and to account for their different magnitudes during training. Note that unlike the original paper, we omit the classification loss term since we are training on a single class.

Our highest-scoring model achieves an average precision (AP@0.5) of 98.9% [24] at a confidence threshold of $p^{\text{conf}} = 0.001\%$ on the hold-out test set after 77 epochs of training.

The model can be trained in less than $1\,\mathrm{h}$ using a batch size of 64 on a recent GPU such as the Nvidia GeForce RTX 2080Ti. We also evaluate the model at different image scales on the onboard computer (Sec. III-A) to determine a reasonable speed/accuracy tradeoff of the detector. We find that performing inference at a resolution of 512×384 pixels (in batches of four, one image per camera) provides accurate detections at a frequency of around $5\,\mathrm{Hz}$. For the experiments, we set the confidence threshold to $p^{\text{conf}} = 50\%$ and use a non-maximum suppression threshold of $p^{\text{nms}} = 60\%$.

B. Multi-agent localization and tracking

1) Relative localization based on known physical size: We compute the relative location of the drone detections using their apparent size in the field of view and the camera parameters. The camera parameters are obtained using the Kalibr visual-inertial calibration toolbox [25]. We use the equidistant camera model for its compatibility with fisheye lenses and its resulting sub-pixel reprojection errors.

To obtain the relative position estimate from the detection, we first compute the unit-norm bearing vector to the center $\boldsymbol{\beta}^{\text{ctr}}$ and one of the extreme points $\boldsymbol{\beta}^{\text{ext}}$ of the image bounding box from the intrinsic camera parameters. We assume the drone can be enclosed by a bounding cube with side length l which is reasonable given its mechanical design (Fig. 5). We can then compute the approximate distance to the three-dimensional center of the detected object as

$$d = \frac{l/2}{\tan(\alpha)} + l/2 \tag{2}$$

where $\alpha = \cos^{-1}(\boldsymbol{\beta}^{\text{ctr}} \cdot \boldsymbol{\beta}^{\text{ext}})$ denotes the angle between the unit-norm bearing vectors. Note that the second term in the above equation accounts for the depth of the object.

2) Multi-agent state estimation using random finite sets: We use the Gaussian mixture probability hypothesis density (GM-PHD) filter [26] to filter out spurious false-positive detections and to estimate the positions and velocities of nearby agents over time. We briefly describe the workings of the filter but refer the reader to [26] for more details. We omit the subscript i to denote the dependence on the focal agent for notational brevity. The following steps are computed independently for each agent in a decentralized fashion.

a) Theory: The GM-PHD filter takes a set of relative localization measurements $\mathcal{Z}_k = \{\mathbf{z}_{k,1}, \ldots, \mathbf{z}_{k,M_k}\}$ as input and computes an output set of agent states $\mathcal{X}_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,J_k}\}$ for each discrete time step k. The states can be described as a single intensity that consists of a weighted sum of Gaussian components of the form

$$v_k(\mathbf{x}_k) = \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}(\mathbf{x}_k; \mathbf{m}_k^{(i)}, \mathbf{P}_k^{(i)}) \tag{3}$$

where $w_k^{(i)}$ denotes the weight associated with each of the J_k Gaussian components which are described by their mean $\mathbf{m}_k^{(i)}$ and covariance $\mathbf{P}_k^{(i)}$. Each of the Gaussian components is then propagated with a *prediction* and *update* step, similar to the Kalman filter.

The *prediction* step can be formalized as

$$v_{k|k-1}(\mathbf{x}_k) = \sum_{i=1}^{J_k} w_{k|k-1} \mathcal{N}(\mathbf{x}_k; \mathbf{m}_{k|k-1}^{(i)}, \mathbf{P}_{k|k-1}^{(i)}) + \gamma(\mathbf{z}_k) \tag{4}$$

with respective weight, mean, and covariance

$$w_{k|k-1}^{(i)} = p_{s,k} w_{k-1}^{(i)} \tag{5}$$

$$\mathbf{m}_{k|k-1}^{(i)} = \mathbf{F}_{k-1} \mathbf{m}_{k-1}^{(i)} + \mathbf{B}_{k-1} \mathbf{u}_{k-1} \tag{6}$$

$$\mathbf{P}_{k|k-1}^{(i)} = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^{(i)} \mathbf{F}_{k-1}^{\top} + \mathbf{Q}_{k-1} \tag{7}$$

where $\mathbf{F}_{k-1}$ is the state transition matrix and $\mathbf{Q}_{k-1}$ the process noise covariance. To account for the egomotion of the observing drone, we include a control input matrix $\mathbf{B}_{k-1}$ and its control input $\mathbf{u}_{k-1}$. We let $p_{s,k}$ denote the probability that a Gaussian component survives the prediction step. We assume an adaptive agent birth model $\gamma_k(\mathbf{z}_k)$ in which each observation generates a new Gaussian component with weight w_γ, mean $\mathbf{m}_\gamma$, and covariance $\mathbf{P}_\gamma$. We further assume that new agents cannot be spawned from existing ones and that there are no spontaneous births without associated measurement.

The *update* step can be formalized as

$$v_k(\mathbf{x}_k) = (1 - p_{d,k}) v_{k|k-1}(\mathbf{x}_k) \tag{8}$$

$$+ \sum_{\mathbf{z}_k \in Z_k} \sum_{i=1}^{J_k} w_k^{(i)}(\mathbf{z}_k) \mathcal{N}(\mathbf{x}_k; \mathbf{m}_{k|k}^{(i)}(\mathbf{z}_k), \mathbf{P}_{k|k}^{(i)}) \tag{9}$$

with respective weight, mean, and covariance

$$w_k^{(i)}(\mathbf{z}_k) = \frac{p_{d,k} w_{k|k-1}^{(i)} q_k^{(i)}(\mathbf{z}_k)}{\kappa_k(\mathbf{z}_k) + \sum_{j=1}^{J_{k|k-1}} p_{d,k} w_{k|k-1}^{(j)} q_k^{(j)}(\mathbf{z}_k)} \tag{10}$$

$$\mathbf{m}_{k|k}^{(i)}(\mathbf{z}_k) = \mathbf{m}_{k|k-1}^{(i)} + \mathbf{K}_k^{(i)}(\mathbf{z}_k - \mathbf{H}_k \mathbf{m}_{k|k-1}^{(i)}) \tag{11}$$

$$\mathbf{P}_{k|k}^{(i)} = (\mathbf{I} - \mathbf{K}_k^{(i)} \mathbf{H}_k) \mathbf{P}_{k|k-1}^{(i)} \tag{12}$$

and

$$q_k^{(i)}(\mathbf{z}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{H}_k \mathbf{m}_{k|k-1}^{(i)}, \mathbf{H}_k \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^{\top} + \mathbf{R}_k) \tag{13}$$

$$\mathbf{K}_k^{(i)} = \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^{\top} (\mathbf{H}_k \mathbf{P}_{k|k-1}^{(i)} \mathbf{H}_k^{\top} + \mathbf{R}_k)^{-1} \tag{14}$$

where $\mathbf{H}_k$ is the measurement matrix and $\mathbf{R}_k$ the measurement noise covariance. We let $p_{d,k}$ denote the probability that an agent is detected during the update step. We model false positive detections as clutter κ_k.

Since the number of Gaussian components increases at each filter iteration, the intensity quickly becomes computationally intractable. Therefore, we prune the components according to the following three conditions to guarantee fast tracking performance. Firstly, we discard components with a weight of less than the truncation threshold of T. Secondly, we merge components with Mahalanobis distance less than the merging threshold of U. Finally, we retain only the $J_{\max}$ components with the largest weights.

b) Implementation: We model the state of each agent as $\mathbf{x}_k = [p_{x,k}, p_{y,k}, v_{x,k}, v_{y,k}]^{\top}$ which consists of relative position $(p_{x,k}, p_{y,k})$ and velocity $(v_{x,k}, v_{y,k})$. The position and velocity components of the state are two-dimensional since the agents are assumed to fly in a planar configuration

at approximately the same altitude. The control input $\mathbf{u}_k$ is defined as the linear velocity of the drone in the body frame which we obtain from the internal state estimate of the autopilot. Finally, the measurements $\mathbf{z}_k = [d_k, \beta_k]^\top$ consist of range and bearing, respectively.

The process and measurement noise covariances are modeled as

$$\mathbf{Q}_k = \sigma_v^2 \begin{bmatrix} \frac{\Delta_k^4}{4}\mathbf{I}_2 & \frac{\Delta_k^3}{2}\mathbf{I}_2 \\ \frac{\Delta_k^3}{2}\mathbf{I}_2 & \Delta_k^2\mathbf{I}_2 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_k = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\beta^2 \end{bmatrix} \tag{15}$$

where Δ_k denotes the time elapsed since the last measurement and is computed as the difference between consecutive timestamps $\Delta_k = t_k - t_{k-1}$. We further let σ_v, σ_d and σ_β denote the standard deviation of the process, range, and bearing noise, respectively.

The state transition function follows a linear Gaussian model and is defined as

$$f(\mathbf{x}_{k-1}, \mathbf{u}_k) = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k \tag{16}$$

where

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{I}_2 & \Delta_k \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix} \qquad \text{and} \qquad \mathbf{B}_k = \Delta_k \mathbf{I}_2. \tag{17}$$

The observation model is nonlinear and consists of measurements of range and bearing

$$h(\mathbf{x}_k) = \begin{bmatrix} d_k \\ \beta_k \end{bmatrix} = \begin{bmatrix} \sqrt{p_{x,k}^2 + p_{y,k}^2} \\ \operatorname{atan2}(p_{y,k}, p_{x,k}) \end{bmatrix} \tag{18}$$

where we use the two-argument function $\operatorname{atan2}$ to avoid ambiguities in the conversion from cartesian to polar coordinates. We linearize the measurement model by computing the Jacobian with respect to the state as

$$\mathbf{H}_k = \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k} = \begin{bmatrix} \frac{p_{x,k}}{\sqrt{p_{x,k}^2+p_{y,k}^2}} & \frac{p_{y,k}}{\sqrt{p_{x,k}^2+p_{y,k}^2}} & \mathbf{0}_2 \\ -\frac{p_{x,k}}{p_{x,k}^2+p_{y,k}^2} & \frac{p_{y,k}}{p_{x,k}^2+p_{y,k}^2} & \end{bmatrix}. \tag{19}$$

We set the probability of detection to $p_d = 90\%$ to provide a slightly more conservative estimate of the detection performance during real-time inference than the results on our test dataset suggest (Sec. II-A2). We assume a probability of survival of $p_s = 100\%$ since agents that are detected once should not disappear. New Gaussian components are initialized directly from the measurements using an adaptive birth model with weight, mean, and covariance

$$w_\gamma = 10^{-5} \tag{20}$$

$$\mathbf{m}_\gamma = [d_k \cos(\beta_k), d_k \sin(\beta_k), 0, 0]^\top \tag{21}$$

$$\mathbf{P}_\gamma = \operatorname{diag}\left(\left[\sigma_p^2, \sigma_p^2, \sigma_v^2, \sigma_v^2\right]\right) \tag{22}$$

where σ_p and σ_v are the standard deviation of the position and velocity which we set to $1\,\mathrm{m}$ and $10\,\mathrm{m\,s^{-1}}$, respectively. The mean is computed by converting the raw measurement from polar to cartesian coordinates, assuming zero initial velocity. We model false positive detections by assuming that we observe one clutter return per time step and therefore set $\kappa_k = 1/a^2$ where $a = 10\,\mathrm{m}$ is the side length of the virtual arena. Although the bearing measurements are precise to a single degree, we set its standard deviation to a slightly larger value of $\sigma_\beta = 3^\circ$ to account for factors such as calibration inaccuracies or imprecisions in the detections due to background clutter. Since we experimentally determined that the range noise varies with the distance to the observer, we model its standard deviation as a function of the measured distance itself (Sec. IV-A). Finally, we model the truncation threshold as $T = 10^{-5}$, the merging threshold as $U = 0.5$, and the maximum number of components to $J_{\max} = 100$.

C. Flocking algorithm

The method described so far could be leveraged by any flocking algorithm. In this work, we use a control algorithm based on the Reynolds flocking rules [27] to compute high-level velocity commands from the relative position estimates of nearby drones [18]. The weighted velocity commands are: 1) a repulsive *separation* term to steer nearby drones away from each other, 2) a *cohesion* term to keep the drones close to each other, and 3) a *migration* term that provides a navigation goal to the swarm. The respective velocity terms for separation, cohesion, and migration can be formalized as

$$\mathbf{v}_i^{\text{sep}} = k^{\text{sep}} \frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} \frac{-\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|} \tag{23}$$

$$\mathbf{v}_i^{\text{coh}} = k^{\text{coh}} \frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} \mathbf{r}_{ij} \tag{24}$$

$$\mathbf{v}_i^{\text{mig}} = k^{\text{mig}} \frac{\mathbf{r}^{\text{mig}}}{\|\mathbf{r}^{\text{mig}}\|} \tag{25}$$

where $\mathbf{r}_{ij}$ denotes the relative position of the j-th agent with respect to agent i, and $\mathcal{A}_i$ the set of neighbors of the i-th agent. The migration point is denoted by $\mathbf{r}^{\text{mig}}$ and expressed relative to the body frame. We further let k^{sep}, k^{coh}, and k^{mig} denote the gains that modulate the strength of the separation, cohesion, and migration terms, respectively. The final velocity command is then given by $\mathbf{v}_i = \tilde{\mathbf{v}}_i / \|\tilde{\mathbf{v}}_i\| \min(\|\tilde{\mathbf{v}}_i\|, v^{\max})$ where $\tilde{\mathbf{v}}_i = \mathbf{v}_i^{\text{sep}} + \mathbf{v}_i^{\text{coh}} + \mathbf{v}_i^{\text{mig}}$ is the sum of all velocity terms and $v^{\max}$ is the desired maximum speed cutoff.

During the experiments, we set the maximum speed to $v^{\max} = 0.5\,\mathrm{m\,s^{-1}}$, and the separation, cohesion, and migration gains to $k^{\text{sep}} = 7$, $k^{\text{coh}} = 1$, and $k^{\text{mig}} = 1$, respectively. The gains are chosen such that the agents converge to an equilibrium distance of approximately $2\,\mathrm{m}$ during migration.

III. Experimental setup

A. Hardware

We use a custom-built quadcopter named LeQuad for all experiments (Fig. 5). Each quadcopter features four FLIR Firefly S global-shutter cameras mounted at a right angle from each other to obtain omnidirectional visual inputs. Each camera is equipped with an OpenMV ultra-wide angle lens which provides a horizontal and vertical field of view of 166° and 116°, respectively. We operate the cameras over a powered USB 2 hub at a binned resolution of 720×540 to obtain grayscale images at a frequency of $10\,\mathrm{Hz}$. We refrain from

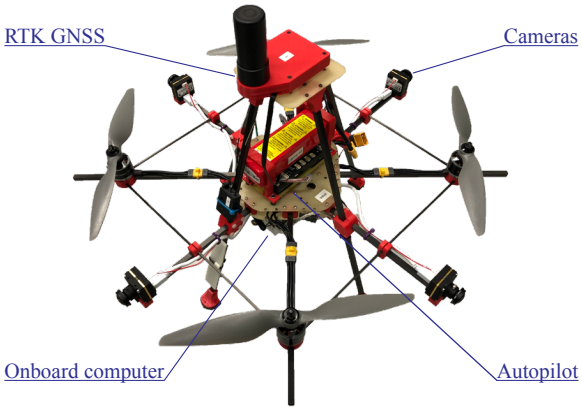


Figure 5: The LeQuad drone features omnidirectional vision via four cameras, a high performance onboard computer with embedded GPU for real-time inference, and a RTK-enabled GNSS to obtain centimeter-accurate ground-truth positions.

using USB 3 to avoid electromagnetic interference with the Drotek F9P RTK-GNSS receiver, which provides centimeter-accurate absolute positions. We use the Nvidia Jetson TX2 mounted on a ConnectTech Orbitty carrier board as an onboard computer and the Holybro Pixhawk 4 as an autopilot.

B. Software

The onboard computer runs Ubuntu 18.04 bundled with the Linux4Tegra (L4T) distribution, and we use ROS Melodic as a robotics middleware. The autopilot runs PX4 and is responsible for hardware-triggering the cameras to provide the resulting images with IMU-synchronized timestamps. The neural networks are trained and evaluated using PyTorch.

IV. Results

We first report the accuracy of the visual relative localization of drones in a controlled indoor environment (Sec. IV-A) and then show vision-based flocking with three real quadcopters performing several navigation tasks outdoors (Sec. IV-B).

The proposed approach has also been extensively tested using the Gazebo simulator with up to ten vision-based agents. For instance, we simulated noisy detections with different false positive and negative rates, as well as processing delays, to find suitable parameter values and operational requirements for the detection, tracking, and control modules. However, we find that it is difficult to realistically model misdetections in simulation since their distribution highly depends on environmental conditions such as visual clutter.

A. Visual relative localization

We show results on the theoretical performance of the visual relative localization system to test its operational bounds and to find suitable values for the range and bearing noise parameters σ_d and σ_β (Eq. 15). To this end, we employ a setup similar to the one used for automatic labeling (Sec. II-A1) except that we additionally obtain ground truth poses of the observing camera and drone from a motion capture system. After transforming

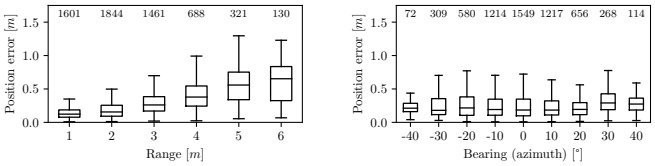


Figure 6: Comparison of vision-based relative localization errors by *range* (left) and *bearing* (right). The bearing estimates are near-constant over the field of view, whereas the range errors increase with distance from the observer. Millimeter-accurate ground-truth positions are obtained at 100 Hz using a motion capture system. The counts above the boxes indicate the number of measurements used to calculate their statitics.

the visual detections into the frame of the motion capture system, we can directly compare the drone's true position with its estimate obtained using vision in metric space. We find that the relative localization error varies considerably as a function of the distance to the drone, whereas the error caused by bearing variations is negligible (Fig. II-B1).

B. Collective outdoor navigation

We report results for three different navigation scenarios: *linear*, *rectangular*, and *circular* migration. Before each flight, we place the drones at roughly 2.5 m distance from each other and wait for their RTK-GNSS receivers to converge to a fixed solution which provides centimeter-accurate absolute positions at 10 Hz. These measurements are only used to provide a reliable ground-truth for the evaluation of the experiments.

After the RTK fix is obtained, we let all agents take off simultaneously and reach an altitude of 2 m before we let the vision-based flocking algorithm take over the control of their motion. The agents are given the same list of migration points depending on the type of navigation scenario. We switch from one migration point to the next as soon as an agent enters an acceptance radius of $r^{\text{acc}} = 3\,\text{m}$. As the list of migration points is exhausted, we repeat the procedure from the first waypoint. We stop the experiment as soon as the battery level of one of the agents reaches a critical capacity of 15%.

The agents' altitude is individually regulated using a proportional controller to constrain their motion to a horizontal plane. However, the planar constraint may be lifted by mounting additional cameras that point to the top and bottom, or by equipping the existing cameras with lenses that provide a larger field of view.

During the linear migration experiment, the agents fly between two waypoints that are located 10 m apart from each other (Fig. 7a). Over a total flight duration of around 2.5 min, the minimum inter-agent distance the agents reach is 1.82 m and the overall mean 2.42 m (Fig. 7d).

The rectangular migration experiment defines four waypoints that are located at the corners of a square with side length 10 m (Fig. 7b). The total flight time is around 3.3 min and the overall minimum and mean inter-agent distances are 1.45 m and 2.36 m, respectively (Fig. 7e).

Finally, the circular migration experiment leads the agents through a series of twelve waypoints that are linearly spaced

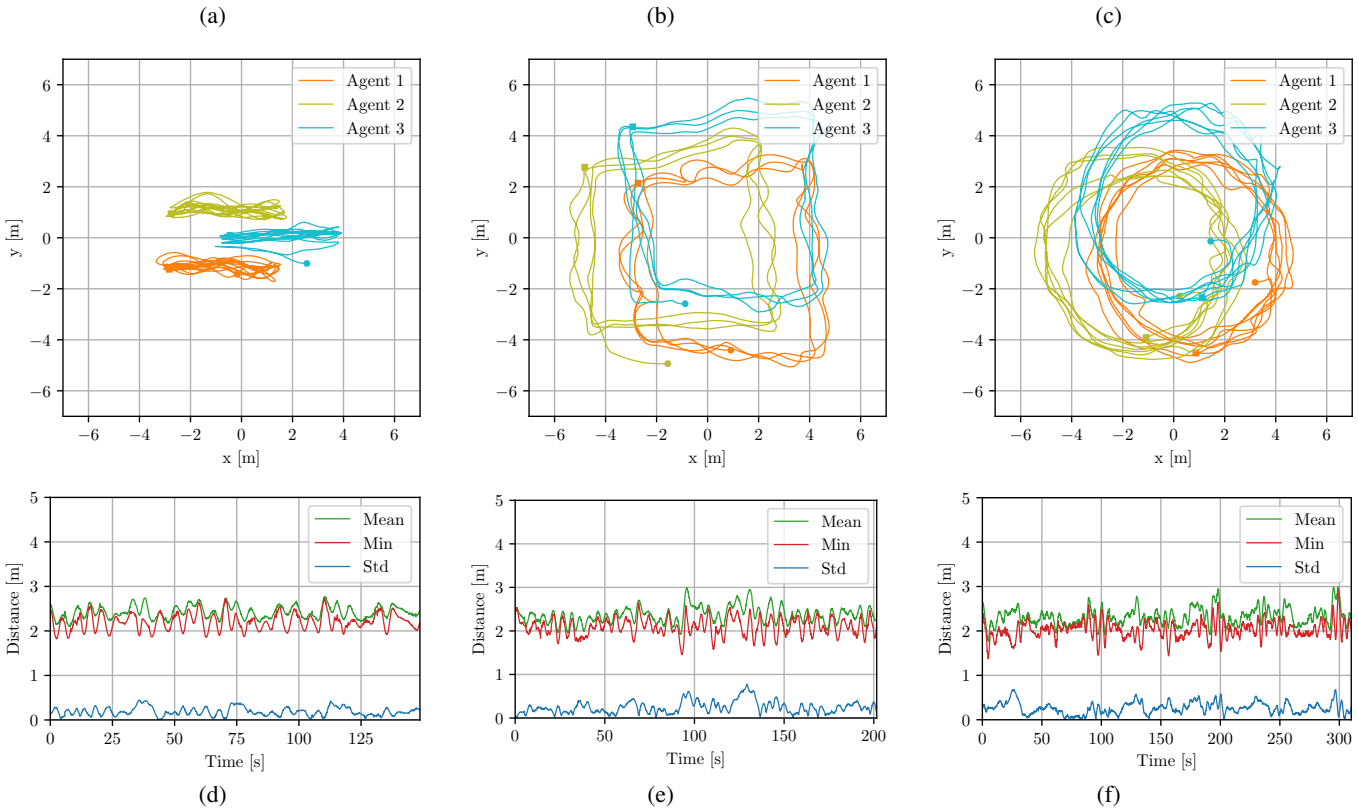


Figure 7: Results of vision-based flocking in outdoor environments. We show *trajectories* (top) and inter-agent *distances* (bottom) for three different migration scenarios: *linear* (left), *rectangular* (middle) and *circular* (right) migration. The drones remain collision-free and cohesive using only local visual information, which is processed onboard in real-time. Centimeter-accurate ground-truth positions are obtained at 10 Hz using RTK-enabled GNSS receivers mounted on the drones. These positions are not shared between the drones during the experiments and only serve to analyze the inter-agent distances.

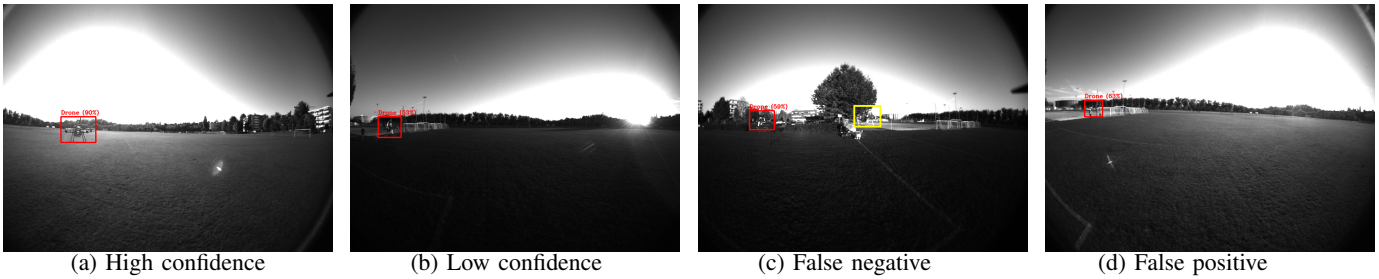
around a circle with 10 m diameter (Fig. 7c). During an overall flight time of around 5 min, the agents remain collision-free while reaching a minimum inter-agent distance of 1.37 m and an overall mean distance of 2.32 m (Fig. 7f).

The above experiments are three representative flights taken from a total of 30 min of collision-free experimental recordings. The progressively lower inter-agent distances across linear, rectangular, and circular migration experiments can be explained by examining the distribution and/or density of waypoints. During the rectangular migration experiment, the lowest inter-agent distances are reached close to the corners where directional changes occur. In the case of the circular migration experiment, the larger number and density of waypoints have a cohesive effect since the agents simultaneously approach points that are more densely spaced.

During the experiments, a variety of objects that can potentially confuse the detector are present in the background (Fig. 8). In descending order of frequency, the most common objects aside from drones are trees, buildings, cars, people, fences, traffic signs, tables, and even dogs. The other drones are generally detected despite background clutter and adverse lighting conditions (Fig. 8a and 8b). False-negative detections occur most frequently when another drone is flying far away and/or in front of the ground control station, i.e. a camping table with the experimental equipment covered by a sun umbrella (Fig. 8c). False-positive detections appear most often in image regions that contain many line-like features since they resemble the mechanical design of the drone (Fig. 8d). They are mostly present for the duration of a single frame and the tracker can reliably reject them. False tracks are occasionally created if false-positive detections are present for more than one frame. However, the false tracks do not cause instabilities that lead to collisions.

V. Conclusions

We presented a vision-based detection and tracking algorithm that enables dense groups of drones to fly cohesively and without mutual collisions. The proposed approach does not depend on visual markers or inter-agent communication and is thus suitable for flocking operation in GNSS-denied environments or in situations where wireless links are unreliable. The approach is fully decentralized since each agent relies exclusively on onboard processing of local visual information to estimate the positions and velocities of neighboring drones. The outdoor navigation experiments show that the system is robust to background clutter and enables collision-free flight even in demanding lighting conditions. These experiments should be considered as minimal validation conditions of



(a) High confidence (b) Low confidence (c) False negative (d) False positive

Figure 8: Qualitative examples of detection results categorized by confidence score (high and low) and type of error (false negative and false positive). High confidence (8a): the drone is easily detected in front of the grass texture despite direct sunlight. Low confidence (8b): the drone blends in with the background due to the line features and lighting conditions. False negative (8c): the right drone (yellow, manually labeled) can not be reliably distinguished from the background clutter. False positive (8d): this type of spurious detection occurs relatively frequently but is filtered out by the multi-agent state tracker.

vision-based flocking algorithms without explicit communication or localization infrastructure. Further experiments are needed to ensure the scalability of the proposed system to larger numbers of vision-based agents.

ACKNOWLEDGMENTS

We thank Olexandr Gudozhnik for the contributions to the drone hardware, Vivek Ramachandran and Enrico Ajanic for the help with conducting outdoor experiments, as well as Enrica Soria, Valentin Wüest, and Davide Zambrano for the helpful discussions.

REFERENCES

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[2] M. Coppola, K. N. McGuire, C. De Wagter, and G. C. H. E. de Croon, "A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints," *Front Robot AI*, vol. 7, p. 18, 2020.

[3] F. Schiano and P. Robuffo Giordano, "Bearing rigidity maintenance for formations of quadrotor UAVs," in *Int Conf Rob Autom (ICRA)*, Singapore, Singapore, May 2017, pp. 1467–1474.

[4] K. N. McGuire, C. D. Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robot*, vol. 4, no. 35, p. eaaw9710, Oct. 2019.

[5] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, and D. Floreano, "Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate," in *Int Conf Intel Rob Sys (IROS)*. IEEE/RSJ, 2011, pp. 5015–5020.

[6] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robot*, vol. 3, no. 20, p. eaat3536, 2018.

[7] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-Efficient Decentralized Visual SLAM," in *Int Conf Rob Autom (ICRA)*. IEEE, pp. 2466–2473.

[8] M. Basiri, F. Schill, P. Lima, and D. Floreano, "Onboard Relative Bearing Estimation for Teams of Drones Using Sound," *IEEE Robot Autom Lett (RA-L)*, vol. 1, no. 2, pp. 820–827, Jul. 2016.

[9] A. Strandburg-Peshkin, C. R. Twomey, N. W. F. Bode, A. B. Kao, Y. Katz, C. C. Ioannou, S. B. Rosenthal, C. J. Torney, H. S. Wu, S. A. Levin, and I. D. Couzin, "Visual sensory networks and effective information transfer in animal groups," *Curr Biol*, vol. 23, no. 17, pp. R709–R711, Sep. 2013.

[10] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting Flying Objects Using a Single Moving Camera," *IEEE Trans Pattern Anal Mach Intell (TPAMI)*, vol. 39, no. 5, pp. 879–892, May 2017.

[11] S. Roelofsen, D. Gillet, and A. Martinoli, "Reciprocal Collision Avoidance For Quadrotors Using On-board Visual Detection," in *Int Conf Intel Rob Sys (IROS)*. Hamburg, Germany: IEEE/RSJ, Sep. 2015, pp. 4810–4817.

[12] V. Walter, N. Staub, A. Franchi, and M. Saska, "UVDAR System for Visual Relative Localization with application to Leader-Follower Formations of Multirotor UAVs," *IEEE Robot Autom Lett (RA-L)*, pp. 1–1, 2019.

[13] K. R. Sapkota, S. Roelofsen, A. Rozantsev, V. Lepetit, D. Gillet, P. Fua, and A. Martinoli, "Vision-based Unmanned Aerial Vehicle Detection and Tracking for Sense and Avoid Systems," in *Int Conf Intel Rob Sys (IROS)*. Daejeon, South Korea: IEEE/RSJ, Oct. 2016, pp. 1556–1561.

[14] M. Vrba and M. Saska, "Marker-Less Micro Aerial Vehicle Detection and Localization Using Convolutional Neural Networks," *IEEE Robot Autom Lett (RA-L)*, vol. 5, no. 2, pp. 2459–2466, Apr. 2020.

[15] E. Soria, F. Schiano, and D. Floreano, "The Influence of Limited Visual Sensing on the Reynolds Flocking Algorithm," in *Int Conf Rob Comp (IRC)*, 2019, pp. 138–145.

[16] R. Opromolla, G. Fasano, and D. Accardo, "A Vision-Based Approach to UAV Detection and Tracking in Cooperative Applications," *Sensors*, vol. 18, no. 10, Oct. 2018.

[17] P. M. Wyder, Y.-S. Chen, A. J. Lasrado, R. J. Pelles, R. Kwiatkowski, E. O. A. Comas, R. Kennedy, A. Mangla, Z. Huang, X. Hu, Z. Xiong, T. Aharoni, T.-C. Chuang, and H. Lipson, "Autonomous drone hunter operating by deep learning and all-onboard computations in GPS-denied environments," *PLOS One*, vol. 14, no. 11, p. e0225092, Nov. 2019.

[18] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning Vision-based Flight in Drone Swarms by Imitation," *IEEE Robot Autom Lett (RA-L)*, vol. 4, no. 4, p. 8, 2019.

[19] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit Lett*, vol. 27, no. 7, pp. 773–780, May 2006.

[20] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv e-prints*, p. arXiv:1804.02767, Apr. 2018.

[21] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression," in *Conf Comp Vis Pat Rec (CVPR)*, Jun. 2019, pp. 658–666.

[22] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv e-prints*, p. arXiv:2004.10934, Apr. 2020.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Europ Conf Comp Vis (ECCV)*. Springer, Sep. 2014, pp. 740–755.

[24] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int J Comput Vis (IJCV)*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[25] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Int Conf Intel Rob Sys (IROS)*, Nov. 2013, pp. 1280–1286.

[26] B. Vo and W. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter," *IEEE Trans Signal Process*, vol. 54, no. 11, pp. 4091–4104, Nov. 2006.

[27] C. W. Reynolds, "Flocks, Herds and Schools: A Distributed Behavioral Model," in *Annual Conf Comp Graph Interactive Technol (SIGGRAPH)*, vol. 14, 1987, pp. 25–34.