Learning Vision-based Flight in Drone Swarms by Imitation

Fabian Schilling, Julien Lecoeur, Fabrizio Schiano, and Dario Floreano

***Abstract*—Decentralized drone swarms deployed today either rely on sharing of positions among agents or detecting swarm members with the help of visual markers. This work proposes an entirely visual approach to coordinate markerless drone swarms based on imitation learning. Each agent is controlled by a small and efficient convolutional neural network that takes raw omnidirectional images as inputs and predicts 3D velocity commands that match those computed by a flocking algorithm. We start training in simulation and propose a simple yet effective unsupervised domain adaptation approach to transfer the learned controller to the real world. We further train the controller with data collected in our motion capture hall. We show that the convolutional neural network trained on the visual inputs of the drone can learn not only robust inter-agent collision avoidance but also cohesion of the swarm in a sample-efficient manner. The neural controller effectively learns to localize other agents in the visual input, which we show by visualizing the regions with the most influence on the motion of an agent. We remove the dependence on sharing positions among swarm members by taking only local visual information into account for control. Our work can therefore be seen as the first step towards a fully decentralized, vision-based swarm without the need for communication or visual markers.**

***Index Terms*—Aerial Systems: Perception and Autonomy, Swarms, Visual Learning, Sensor-based Control**

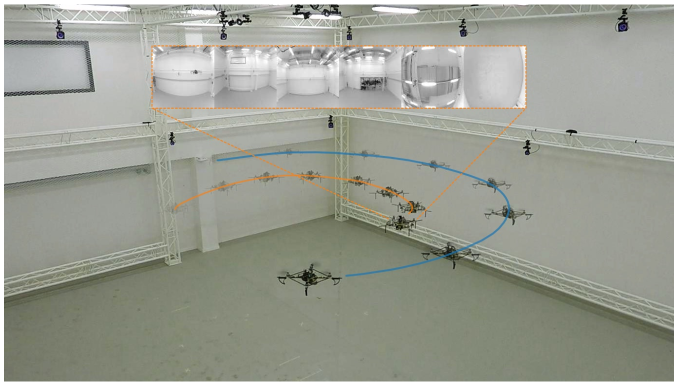


Figure 1: Vision-based multi-agent experiment in our motion tracking hall. Our proposed visual controller operates fully decentralized and provides collision-free, coherent collective motion without the need to share positions among agents. The behavior of an agent only depends on its omnidirectional visual inputs (see orange rectangle). Collision avoidance and cohesion between agents are learned entirely from visual inputs (see supplementary video).

Supplementary Material

Supplementary video: https://youtu.be/I9vFvPphfpU.

I. Introduction

COLLECTIVE motion of animal groups such as flocks of birds is an awe-inspiring natural phenomenon that has profound implications for the field of aerial swarm robotics [1], [2]. Animal groups in nature operate in a completely self-organized manner since the interactions between them are purely local. By taking inspiration from decentralization in biological systems, we can develop powerful robotic swarms that are 1) robust to failure, and 2) highly scalable since the number of agents can be increased or decreased dynamically depending on the workload of the task.

One of the most appealing characteristics of collective animal behavior for robotics is that decisions are made based on local information. Thus, the behavior of animal groups does not require extensive knowledge of the swarm state or a central coordinator. As of today, however, most multi-agent robotic systems rely on entirely centralized control [3]–[5] or wireless communication of positions [6]–[8], which are obtained either from a motion capture system or global navigation satellite system (GNSS). The main drawback of these approaches is the introduction of a single point of failure, as well as the use of unreliable data links, respectively. Relying on centralized control bears a significant risk since the agents lack the autonomy to make their own decisions in failure cases such as a communication outage. The possibility of failure is even higher in dense urban environments, where GNSS measurements are often unreliable and imprecise.

Vision is arguably the most promising sensory modality to achieve a maximum level of autonomy for robotic systems, particularly considering the recent advances in computer vision and deep learning [9]. Apart from being light-weight and having relatively low power consumption, even cheap commodity cameras provide an unparalleled information density with respect to sensors of similar cost. Their characteristics are specifically desirable for the deployment of an aerial multi-robot system. The difficulty when using cameras for robot control is the processing of the visual information which this paper addresses directly.

Manuscript received: May 27, 2019; Accepted July 28, 2019.

This paper was recommended for publication by Editor Jonathan Roberts upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Swiss National Science Foundation (SNSF) with grant number 200021-155907 and the European project RoboCom++.

The authors are with the Laboratory of Intelligent Systems (LIS), École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland. `https://lis.epfl.ch`

Corresponding author: `fabian.schilling@epfl.ch`

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

In this work, we propose a reactive control strategy based entirely on local visual information. We formulate the swarm interactions as a regression problem in which we predict control commands as a nonlinear function of the visual input of a single agent. To the best of our knowledge, this is the first successful attempt to learn vision-based swarm behaviors such as collision-free navigation in an end-to-end manner directly from raw images.

Our contributions can be summarized as follows:

- We propose a data-efficient imitation learning approach to solve the problem of vision-based coordination of a swarm of drones. Our control policy is trained incrementally by following the previous best policy and thus collecting relevant data from its failure cases. Our proposed system generates high-level control commands from raw images in the form of velocity setpoints, whereas a classical cascaded feedback control architecture handles low-level control.
- We present a remarkably simple and effective task-specific unsupervised domain adaptation approach to transfer the image data obtained from simulation to the real world. To this end, we collect a dataset of unlabeled images from our target environment to serve as backgrounds for images generated in simulation.
- We implement our algorithm on a physical quadrotor platform and show that all computations (policy evaluation, state estimation, and control) can be run entirely onboard in real-time.
- We provide an evaluation of our system in simulation and experimental validation in a motion tracking hall to show that our control policy generalizes to coordinated multi-agent flights in the real world.

II. Related Work

Decentralized swarms of drones such as quadrotors and fixed-wings are the focus of recent research in swarm robotics. Early work presents ten fixed-wing drones deployed in an outdoor environment [10]. Their collective motion is based on Reynolds flocking [11] with a migration term that allows the swarm to navigate towards the desired goal. Thus far, the largest decentralized quadrotor swarm consisted of 30 autonomous agents flying in an outdoor environment [8]. The underlying algorithm has many free parameters which are optimized using an evolutionary algorithm that relies on a fitness function which incorporates several swarm order parameters. The commonality of the mentioned approaches and others, for example, [6], [12], is the ability to share GNSS positions wirelessly among swarm members. However, there are many situations in which wireless communication is unreliable, or GNSS positions are too imprecise. We may not be able to tolerate position imprecisions in situations where the environment requires a small inter-agent distance, for example when traversing narrow passages in urban environments. In these situations, tall buildings may deflect the signal and communication outages occur due to the wireless bands being over-utilized.

Recent advances in the field of machine learning facilitate vision-based control of flying robots. In particular, the controllers are based on three types of learning methods: imitation learning, supervised learning, and reinforcement learning. Imitation learning is used in [13] to control a drone in a forest environment based on human pilot demonstrations. The authors motivate the importance of following suboptimal control policies in order to cover more of the state space. A supervised learning approach [14] features a convolutional network that is used to predict a steering angle and a collision probability for drone navigation in urban environments. In contrast with the previous methods based only on supervised learning, an approach based on reinforcement learning [15] shows that a neural network trained entirely in a simulated environment can generalize to flights in the real world. The work described above and other similar methods, for instance, [16], [17], use a data-driven approach to control a flying robot in real-world environments. The probability of collision is learned by minimizing the binary cross-entropy of labeled images collected while riding a bicycle through urban environments. A shortcoming of these methods is that the learned controllers operate only in two-dimensional space which bears similar characteristics to navigation with ground robots. Moreover, the approaches do not show the ability of the controllers to coordinate a multi-agent system.

The control of multiple agents based on visual inputs is achieved with relative localization techniques [18] for a group of three quadrotors. Each agent is equipped with a camera and a circular marker that enables the detection of other agents and the estimation of relative distance. The system relies only on local information obtained from the onboard cameras in near real-time. Thus far, decentralized vision-based drone control has been realized by mounting visual markers on the drones [19]. Although this simplifies the relative localization problem significantly, the marker-based approach would not be desirable for real-world deployment of flying robots. The used visual markers are relatively large and bulky which unnecessarily adds weight and drag to the platform; this is especially detrimental in real-world conditions. Another recently proposed approach is the use of active ultraviolet markers to identify the relative range and bearing to other agents [20]. However, the markers have to be placed in carefully chosen pre-defined locations, and the system is thus unable to detect markerless drones that do not precisely conform to these specifications.

III. Method

At the core of our method lies the prediction of a velocity command for each agent that matches the velocity command computed by a flocking algorithm. We consider the velocity command from the flocking algorithm as the target for a supervised imitation learning problem. The main idea is to eliminate the dependence on the knowledge of the positions of other agents by taking only local visual information into account for control. Imitation learning presents a practical alternative to the approach in which separate modules are responsible for object detection, multi-object target tracking, and control, respectively. The modular approach would require the manual labeling of prohibitively large amounts of images with precise bounding box annotations. By using direct imitation, the control inputs can be calculated directly from the relative positions of other agents obtained either from simulation or a motion capture system.

A. Flocking algorithm

We use an adaptation of Reynolds flocking [11] to generate targets for our learning algorithm. In particular, we only consider the *collision avoidance* and *flock centering* terms from the original formulation since they only depend on relative positions. We omit the *velocity matching* term since estimating the velocities of other agents is a challenging task given only a single snapshot in time. One would have to rely on either estimating velocities from several consecutive images or estimating the orientation and heading with relatively high precision in order to infer velocities from a single image.

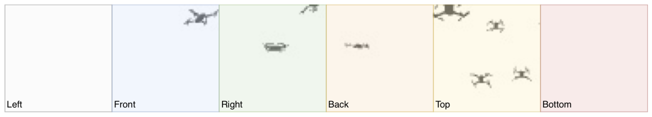
In our formulation of the flocking algorithm, we use the terms *separation* and *cohesion* to denote collision avoidance and flock centering, respectively [21]. We further add an optional *migration* term that enables the agents to navigate towards a goal. An important consideration when modeling the desired behavior of the swarm is the notion of neighbor selection. It is reasonable to assume that each agent can only perceive its neighbors in a limited range. We therefore only consider agents as neighbors if they are closer than the desired cutoff distance $r^{\max}$ which corresponds to only selecting agents in a sphere with a given radius. We do not make any restrictions on the field of view of the agents since limiting perception, specifically in the lateral direction, has been shown to have adverse effects on the flocking performance [22]. Therefore, we denote the set of neighbors of an agent i as the set $\mathcal{N}_i = \{\text{agents } j : j \neq i \wedge \|\mathbf{r}_{ij}\| < r^{\max}\}$ where $\mathbf{r}_{ij} \in \mathbb{R}^3$ denotes the relative position of agent j with respect to agent i and $\|\cdot\|$ the Euclidean norm. We compute $\mathbf{r}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ where $\mathbf{p}_i \in \mathbb{R}^3$ denotes the absolute position of agent i.

The separation term steers an agent away from its neighbors in order to avoid collisions, whereas the cohesion term can be seen as the antagonistic inverse since its purpose is to steer an agent towards its neighbors to provide coherence to the group. We can formalize the respective separation and cohesion velocity command for the ith agent as

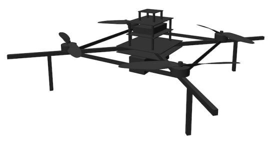
$$\mathbf{v}_i^{\text{sep}} = -\frac{k^{\text{sep}}}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|^2} \tag{1}$$

$$\mathbf{v}_i^{\text{coh}} = \frac{k^{\text{coh}}}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{r}_{ij} \tag{2}$$

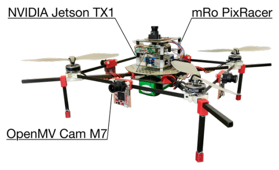
where k^{sep} is the separation gain which modulates the strength of the separation between agents and the cohesion gain k^{coh} modulates the tendency for the agents to be drawn towards the center of the neighboring agents. For our implementation, the separation and cohesion terms are sufficient to generate a collision-free swarm in which agents remain together, given that the separation and cohesion gains are chosen carefully. We denote the combination of the two terms as the Reynolds velocity command $\mathbf{v}_i^{\text{rey}} = \mathbf{v}_i^{\text{sep}} + \mathbf{v}_i^{\text{coh}}$ which is later predicted by the neural network.



(a) Visual input of an agent: concatenation of six camera images



(b) Simulated drone model



(c) Physical drone hardware

Figure 2: Camera configuration and resulting visual input for a simulated agent. The cameras are positioned such that the visual field of an agent corresponds to an image cube map, i.e., each camera is pointing at a different face of a cube as seen from within the cube itself. (2a) The concatenation of the six camera images into the full visual field (color-coded by camera). (2b) Simulated model of the drone built using mainly geometric primitives. (2c) Hardware implementation of the drone based on the design in [12]. It uses six OpenMV Cam M7 with ultra-wide angle lenses for image acquisition, an NVIDIA Jetson TX1 for image processing, and a Pixracer autopilot for state estimation and control.

Moreover, the addition of the migration term provides the possibility to give a uniform navigation goal to all agents. The corresponding migration velocity command is given by

$$\mathbf{v}_i^{\text{mig}} = k^{\text{mig}} \frac{\mathbf{r}_i^{\text{mig}}}{\|\mathbf{r}_i^{\text{mig}}\|} \tag{3}$$

where k^{mig} denotes the migration gain and $\mathbf{r}_i^{\text{mig}} \in \mathbb{R}^3$ denotes the relative position of the migration point with respect to agent i. We compute $\mathbf{r}_i^{\text{mig}} = \mathbf{p}^{\text{mig}} - \mathbf{p}_i$ where $\mathbf{p}^{\text{mig}} \in \mathbb{R}^3$ is the absolute position of the migration point.

The velocity command for an agent i is computed as a sum of the Reynolds terms, which is a combination of separation and cohesion, as well as the migration term, as $\tilde{\mathbf{v}}_i = \mathbf{v}_i^{\text{rey}} + \mathbf{v}_i^{\text{mig}}$. In general, we assume a homogeneous swarm, which means that all agents are given the same gains for separation, cohesion, and migration.

A final parameter to adjust the behavior of the swarm is the cutoff of the maximum speed. The final velocity command that steers an agent is given by $\mathbf{v}_i = \tilde{\mathbf{v}}_i / \|\tilde{\mathbf{v}}_i\| \min\left(\|\tilde{\mathbf{v}}_i\|, v^{\max}\right)$ where $v^{\max}$ denotes the desired maximum speed of an agent during flocking.

B. Drone model

We perform simulation in Gazebo with a group of nine quadrotor drones, each equipped with six simulated cameras to provide omnidirectional vision. The cameras are positioned away from the center of gravity of the drone in order to have an unobstructed view of the surrounding environment, including the propellers (see Fig. 2a). Each camera has a $135 \times 90°$ horizontal and vertical field of view and takes a grayscale

Algorithm 1: Multi-agent dataset aggregation.

```
Initialize empty dataset $\mathcal{D} \leftarrow \emptyset$.
Initialize parameters of learned policy $\hat{\pi}_1$.
for $i \leftarrow 1$ to $N$ do
    Sample trajectories from learned policy $\hat{\pi}_i$
      simultaneously for all agents.
    Collect dataset $\mathcal{D}_i = \{(\mathbf{o}_t, \pi^*(\mathbf{s}_t))\}_{t=1}^T$ of
      observations from the learned policy $\hat{\pi}_i$ and
      actions given by expert $\pi^*$ for all agents.
    Aggregate datasets $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$.
    Train new policy $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end
return Best policy $\hat{\pi}_i$ on hold-out validation set $\mathcal{D}_{\text{val}}$.
```

image of 128×128 pixels with a refresh rate of 10 Hz. We concatenate the images from all six cameras along the horizontal axis to form a 128×768 pixels grayscale image.

C. Imitation learning

We use an on-policy imitation learning approach to synthesize a purely vision-based control policy, denoted by $\hat{\pi}$, that matches the behavior of the position-based flocking policy, denoted by π^*, as closely as possible. More formally, we denote the learned policy $\hat{\pi}(\mathbf{o}_t) = \mathbf{a}_t$ as a mapping from observations to actions, where the observations $\mathbf{o}_t \in \mathbb{R}^{128 \times 768}$ are grayscale images and the actions $\mathbf{a}_t \in \mathbb{R}^3$ velocity commands for each time step $t \in T$. The expert policy $\pi^*(\mathbf{s}_t) = \mathbf{a}_t$, on the other hand, computes velocity commands from the state $\mathbf{s}_t$ of the system, which is represented by known relative positions $\mathbf{r}_{ij}$ to other agents as described in Sec. III-A. The image observations can be seen as a lossy representation of the underlying system state (e.g., the relative positions of other agents) because of adverse factors such as limited resolution, occlusions, lens distortions, and inherent noise in the system. To learn the vision-based policy $\hat{\pi}$, we use the DAGGER imitation learning algorithm [23] (see Alg. 1).

We collect data and train our policy in an iterative fashion, first in simulation and then in our motion tracking hall. We use a roughly $80\%/20\%$ split between training $\mathcal{D}_{\text{train}}$ and validation data $\mathcal{D}_{\text{val}}$. In simulation, each iteration of the imitation learning algorithm proceeds as follows. The drones take off and assume random positions within a cube of side length 4m, and with a minimum inter-agent distance of 1.5m. The side length and minimum distance were chosen to resemble a plausible real-world deployment scenario in a confined environment such as our motion tracking hall. All agents then switch to vision-based control and use raw velocity commands generated by the learned policy (which is randomly initialized at first) from the visual inputs sampled at 10Hz. Simultaneously, ground truth control commands are computed from the flocking algorithm and stored for post-processing. The iteration is considered complete as soon as 1) any two drones collide, 2) any two drones are too far away from each other, or 3) 200 observation-action samples are generated. We consider two agents too close if any pair of drones falls below a collision threshold of 1m; similarly, we consider two agents as too far away when the distance between them exceeds a threshold of 7m. The collision threshold follows the constraints of the drone model, and the dispersion threshold stems from the diminishing size of other agents in the field of view. For data collection in the real world, we relax the above requirements and stop an iteration as soon as the situation becomes subjectively too dangerous, for instance when the inter-agent distance becomes too small, or the drone starts to move to close to the walls of the motion tracking hall. A new policy is then trained using the collected image samples, the control commands generated by the learned policy, and the expert control commands computed from the flocking algorithm rules. Finally, the data collection process is repeated with the new policy.

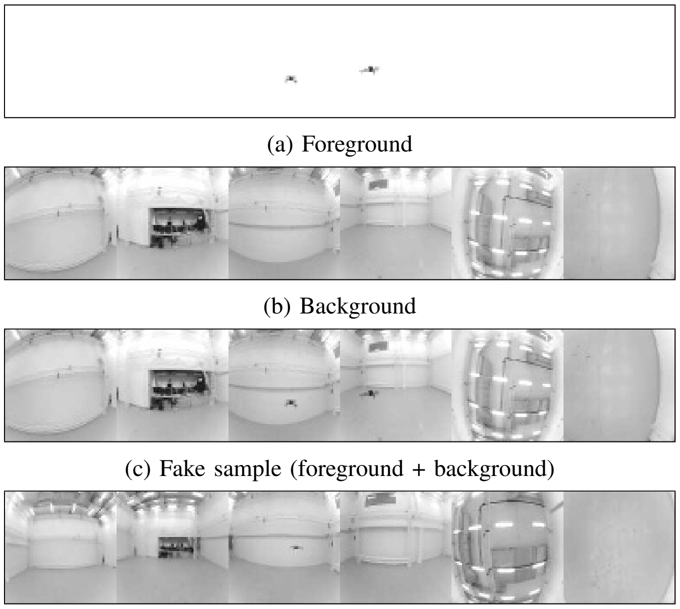
D. Domain adaptation

One fundamental problem with the on-policy imitation learning approach outlined in Alg. 1 is that the policy needs to be executed in the real world in order to collect samples. Executing an untrained policy in a multi-agent setting with quadcopters in a confined space such as a motion tracking hall can be dangerous. A possible solution is to set an initial policy $\pi_i(\mathbf{o}_t) = \beta_i \pi^*(\mathbf{s}_t) + (1 - \beta_i)\hat{\pi}_i(\mathbf{o}_t)$, i.e. a linear combination of the expert and learner's action and let the factor β_i decay from one to zero over time. While this approach would work, data generation in the real world is error-prone and collecting large datasets would require significant amounts of time.

To avoid the collection of a large real-world dataset, we propose a simple yet effective task-specific domain adaptation method to learn an initial vision-based policy from simulated and unsupervised images. To this end, we construct a simulated environment in which there is no visual clutter such that the drones appear in front of a uniform white background (see Fig. 3a). Next, we collect a $20k$-sample image background dataset from our six onboard cameras during a single-agent flight in our motion tracking hall (see Fig. 3b). During the flight, we tried to rotate the drone in all possible orientations and to cover as much of the space in the hall as possible to increase the variability in the image data. As a final step, we add the simulated drones onto the background in order to create a dataset that resembles actual drones flying in the motion tracking hall. The real and domain-adapted images are almost indistinguishable to the human eye at the resolution used by the control policy (see Fig. 3c and 3d). The control actions corresponding to the images from the simulated dataset remain unchanged during this process.

E. Visual policy

We formulate the vision-based imitation of the flocking algorithm as a regression problem which takes an image (see Fig. 2a) as an input and predicts a velocity command which matches the ground truth velocity command as closely as possible. To produce the desired velocities, we consider a small and efficient convolutional neural network [14] that is geared towards drone navigation. However, unlike [14], we opt for a single-head regression architecture to avoid


(a) Foreground

(b) Background

(c) Fake sample (foreground + background)

(d) Real sample

Figure 3: Example of unsupervised domain adaptation method in which simulated foreground images (3a) and real background images from our motion tracking hall (3b) are combined into domain-adapted images (3c). For visual comparison, we also show a real sample from a two-agent flight in the motion tracking hall (3d).

convergence problems caused by different gradient magnitudes from an additional classification objective during training. This simplifies the optimization problem and the model architecture and thus the resulting controller.

We use mini-batch stochastic gradient descent to minimize the regularized mean squared error loss between predicted and target velocity commands. We employ variance-preserving parameter initialization by drawing the initial weights from a truncated normal distribution according to [24]. The biases of the model are initialized to zero. The objective function is minimized using the Adam optimizer [25] and an initial learning rate of 10^{-3} which is decayed by a factor of 0.5 after 10 consecutive epochs without improvement on the hold-out validation set. We train the network using a mini-batch size of 128, a weight decay factor of $5{\cdot}10^{-4}$, and a dropout probability of 0.5. We stop the training process as soon as the validation loss plateaus for more than ten consecutive epochs.

The raw images and velocity targets are pre-processed using feature standardization such that each input batch has a mean of zero and a standard deviation of one. For the velocity targets from the flocking algorithm, we perform a frame transformation from the world frame $\mathcal{W}$ into the drone's body frame $\mathcal{B}$ as $\mathbf{v}_i = \mathbf{R}^{\mathcal{B}}_{\mathcal{W}i}\mathbf{v}^{\text{rey}}_i$ where $\mathbf{R}^{\mathcal{B}}_{\mathcal{W}i} \in \text{SO}(3)$ denotes the rotation matrix from world to body frame for robot i and $\mathbf{v}^{\text{rey}}_i$ corresponds to the target velocity command. We perform the inverse rotation to transform the predicted velocity commands from the neural network back into the world frame. In terms of data augmentation, we randomly adjust the image brightness and contrast of each mini-batch by $\pm 25\%$. Furthermore, we randomly rotate the image cube map and the control command in $90°$ increments around the body frame z-axis (yaw) such that the vision-based controller becomes invariant to the direction in which agents are predominantly present in the data. In practice, this is equivalent to shifting and wrapping around the first four images (left, front, right, and back) in 128-pixel increments, as well as rotating the last two images (top and bottom) by $90°$ increments.

IV. Simulation Results

This section presents an evaluation of the learned controller as a comparison to the target flocking algorithm. We refer to the swarm operating on the learned controller (which relies on visual inputs) as *vision-based*. We refer to the swarm operating on the flocking algorithm (which relies on shared agent positions) as *position-based*. The results show that the proposed controller represents a robust alternative to communication-based systems in which the positions of other agents are shared with other members of the group.

The experiments are performed using the Gazebo simulator in combination with the PX4 autopilot [26] for state estimation and control. The neural network is implemented in PyTorch. We employ the same set of flocking parameters used during the training phase throughout the following experiments. We set the number of agents $N = 9$, the maximum perception radius $r^{\text{max}} = 7\text{m}$, and the maximum speed $v^{\text{max}} = 2\,\text{m}\,\text{s}^{-1}$. We set the separation, cohesion, and migration gain to $k^{\text{sep}} = 7$, $k^{\text{coh}} = 1$, and $k^{\text{mig}} = 1$, respectively.

We report our results in terms of minimum and maximum inter-agent distances, two complementary metrics that describe the state of the swarm at a given time step. The minimum and maximum inter-agent distance are direct indicators for successful collision avoidance, as well as general segregation of the swarm, respectively. Two conditions are tested: a first one in which all agents share a common migration goal, and a second one in which a subset of the agents have an opposing migration goal.

A. Common migration goal experiment

In the first experiment, we give all agents the same migration goal and show that the swarm remains collision-free during navigation. The *vision-based* and the *position-based* swarm exhibit remarkably similar behavior while migrating (see Figs. 4a and 4c). For the *vision-based* controller, one should notice that the velocity commands predicted by the neural network are sent to the agents in their raw form without any further processing. The *vision-based* swarm matches the *position-based* one very well since the inter-agent distances do not deviate significantly over the course of the entire trajectory (see Fig. 4e). The minimum inter-agent distance remains larger than the collision threshold of 1m, which indicates that the neural controller has learned to keep a minimum inter-agent distance and thus to avoid collisions.

B. Opposing migration goals experiment

In this experiment, we assign different migration goals to two subsets of agents. The first group, consisting of five agents,

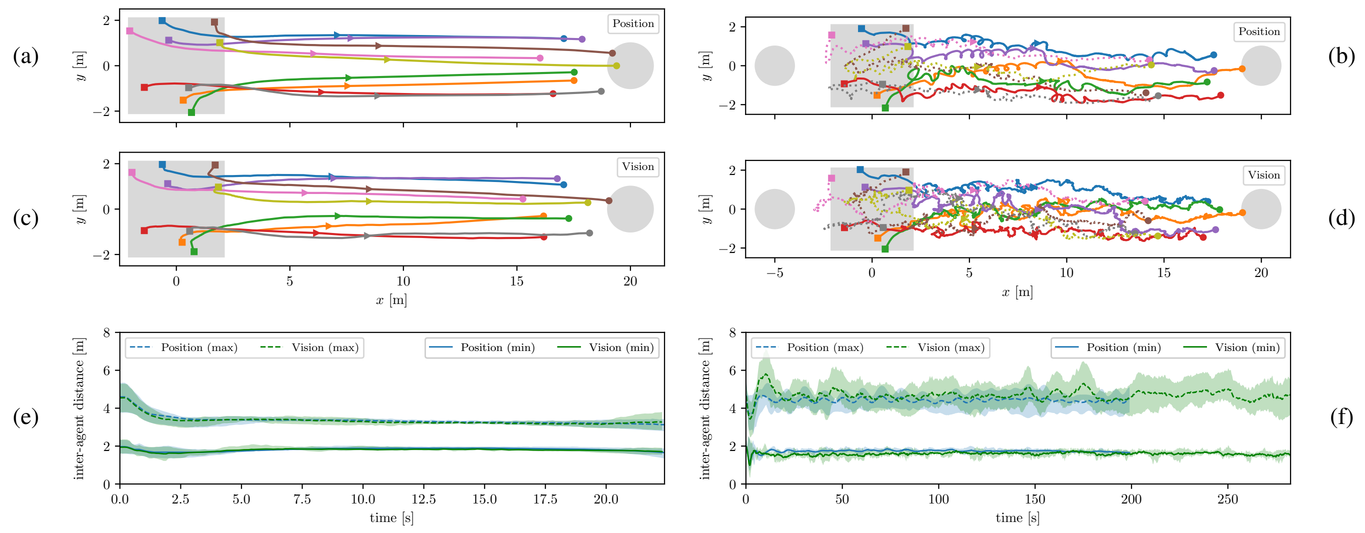


Figure 4: Flocking with common migration goal (left column) and opposing migration goals (right column). **First two rows**: Top view of a swarm migrating using the *position-based* (4a and 4b) and *vision-based* (4c and 4d) controller. The trajectory of each agent is shown in a different color. The colored squares, triangles, and circles show the agent positions during the first, middle, and last time step, respectively. The gray square and gray circle denote the spawn area and the migration point, respectively. For the swarm with opposing migration goal (4b and 4d), the waypoint on the right is given to a subset of five agents (solid lines), whereas the waypoint on the left is given to a subset of four agents (dotted lines). **Third row**: Inter-agent minimum and maximum distances over time (4e and 4f) while using the *position-based* and *vision-based* controller. The mean minimum distance between any pair of agents is denoted by a solid line, whereas mean maximum distances are shown as a dashed line. The colored shaded regions show the minimum and maximum distance between any pair of agents. Note that the plot for the opposing goal swarm does not continue until the last time step since the *vision-based* swarm takes longer than the *position-based* swarm to reach the migration point.

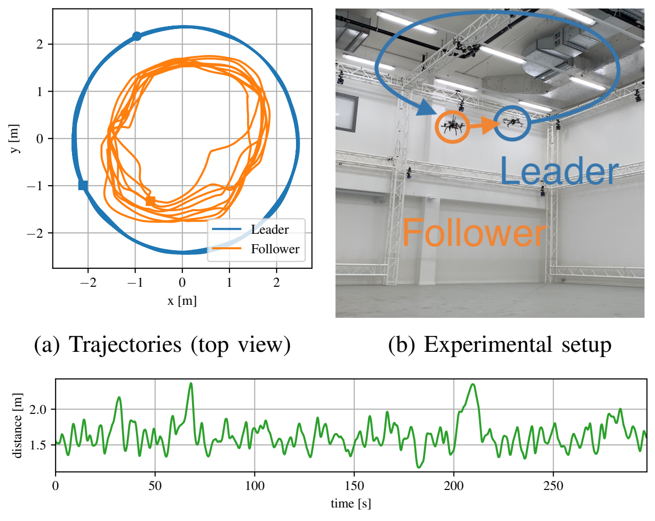
is assigned the same waypoint as in Sec. IV-A. The second group, consisting of the remaining four agents, is assigned a migration point on the opposite side with respect to the first group. The *position-based* and *vision-based* swarm exhibit very similar migration behaviors (see Figs. 4b and 4d). In both cases, the swarm cohesion is strong enough to keep the agents together despite the diverging migration goals. Note that the *vision-based* swarm reaches its migration goal far later than the *position-based* swarm.

V. Real-world Results

We propose three experiments involving two quadrotors to show that the learned controller can perform vision-based markerless flight in the real world. We conclude with an attribution study that visualizes the regions of the visual input that contribute the most to the neural network's predictions. All flights are performed in our motion tracking hall that is equipped with 26 OptiTrack cameras. Each drone receives its ground truth pose via Wi-Fi at a frequency of $100\,\mathrm{Hz}$.

A. Circle experiment

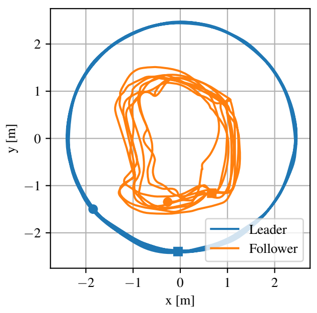
The circle experiment showcases the ability of the learned policy to maintain cohesion with another agent. To this end, the leader drone is given a circular trajectory, whereas the vision-based follower uses raw velocity commands generated onboard by the neural network. We set the radius of the circle as $2.5\,\mathrm{m}$ and the angular velocity along the circular trajectory to $10\,^\circ\,\mathrm{s}^{-1}$. The follower drone keeps a stable distance between itself and the leader drone during a representative $6\,\mathrm{min}$ flight (see Fig. 5). One can observe that the visual policy can recover from small mistakes reliably, most notably after the $70\,\mathrm{s}$ and $210\,\mathrm{s}$ marks (see Fig. 5c).

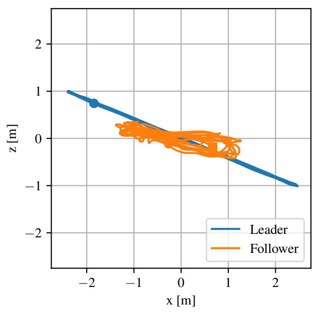


(a) Trajectories (top view) (b) Experimental setup

(c) Inter-agent distance over time

Figure 5: Ground truth trajectories and inter-agent distance during the circle experiment.


(a) Trajectories (top view)


(b) Trajectories (side view)

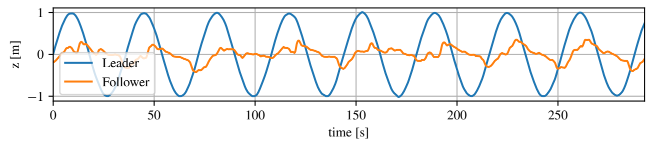
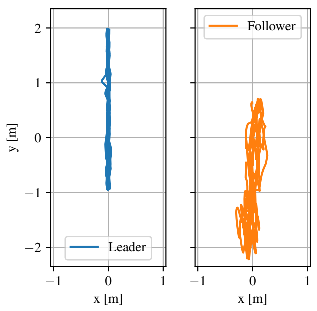
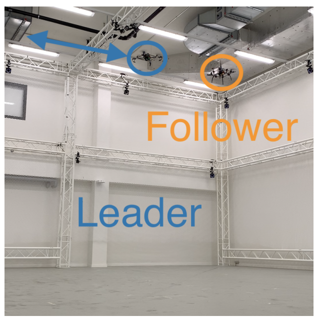

(c) Altitude over time

Figure 6: Ground truth trajectories and altitude during the carousel experiment.


(a) Trajectories (top view)



(b) Experimental setup

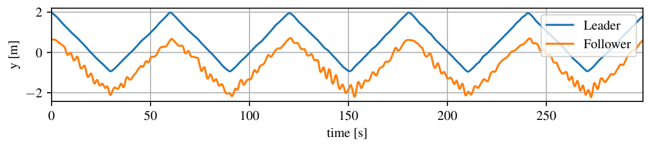

(c) Positions over time

Figure 7: Ground truth trajectories and positions during the push-pull experiment.

B. Carousel experiment

The carousel experiment can be seen as an extension of the circle scenario with the added difficulty that the altitude of the leader is now modulated by a sinusoid as well. The parameters of the circle trajectory remain the same, but we add a sinusoid component to the altitude tracked by the leader drone. The altitude component has an amplitude of $1\,\mathrm{m}$ and the same frequency as the horizontal components, which leads to a tilted circular trajectory (see Fig. 6b). The vision-based follower thus needs the ability to operate in full 3D space in order to stay cohesive with the leader. The inter-agent distance in the carousel experiment increases slightly compared to the circle experiment, especially when the leader agent deviates the most from the average flight altitude. Nevertheless, the vision-based drone can maintain a steady cohesion with the leader (see Fig. 6). Upon closer examination of the altitude of both agents over time, it is clear that the follower can modulate its altitude as a reaction to the leader. The extreme points in altitude of the follower are indeed somewhat aligned with the intersection points of the two curves (see Fig. 6c).

C. Push-pull experiment

The motivation for the push-pull experiment is the validation of the separation ability of the vision-based flocking policy. In both the circle and carousel experiments, the follower drone is never on a direct collision course with the leader. In order to encourage collisions, we let the leader navigate between two waypoints and position the follower in the middle. We further set both the x and the z-component of the velocity command computed by the neural network to zero in order to fix the follower's degrees of freedom to a line defined by the two waypoints. This adjustment is necessary to show collision avoidance since leaving the control input unrestricted would degenerate into a cohesion-like scenario where collisions are not explicitly encouraged. Moreover, since the drones may get into situations where they are on top of each other, downwash may blur the lines between the separation due to the learned controller and the physical repulsion due to the airflow. The vision-based follower avoids collisions and maintains a constant equilibrium distance to the leader drone (see Fig. 7). This behavior results directly from the spring-like dynamics of two agents in which one is following the flocking algorithm. The oscillations in the position may occur because distances smaller than equilibrium are penalized quadratically, while distances larger than equilibrium are penalized linearly by the flocking rules (see Eq. 1). In addition, noise in the raw control command leads to small and sudden repulsive maneuvers that are quickly compensated (see Fig. 7c).

D. Attribution study

Since the *vision-based* controller provides a very tight coupling between perception and control, the need for interpretation of the learned behavior arises. To this end, we employ a state-of-the-art attribution method [27], which shows how much influence each pixel in the input image has on the predicted velocity command (see Fig. 8). More specifically, we compute the gradients for the heat map with respect to the last convolutional layer of the neural network in which the individual feature maps have a spatial size of only 8×48 pixels. We then employ bilinear upsampling to increase the resolution of the resulting saliency map before we blend it with the original input image using a jet colormap for visualization purposes. The attribution map can be generated very efficiently using one forward and backward pass and could therefore serve as a valuable attention-like input for further real-time processing.

One can observe that the network is effectively localizing the other agent spatially in the visual input. However, the network is putting non-zero importance on regions with more visual clutter such as the control room in the backward-facing

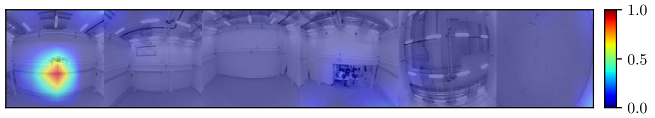


Figure 8: Heat map visualization of the relative importance of each pixel in the visual input of the drone towards its velocity command. Red regions have the most influence on the control command, whereas blue regions contribute the least. Best viewed in color.

camera (see Fig. 8). One may note that the most salient region is not perfectly matching the location of the visible agent, which can be attributed to the low spatial resolution of the activations generated at the last convolutional layer.

VI. Conclusions and Future Work

This paper presented a machine learning approach to the problem of collision-free and coherent motion of a dense swarm of quadcopters. The agents learn to coordinate themselves entirely via visual inputs in 3D space by mimicking a flocking algorithm. The learned controller removes the need for communication of positions among agents and thus presents the first step towards a fully decentralized vision-based swarm of drones. The trajectories of the swarm are relatively smooth even though the controller is based on raw neural network predictions. Our algorithm naturally handles navigation tasks by adding a migration term to the predicted velocity of the neural controller.

Regarding future work, a natural subsequent step will be to scale up the real-world experiments with more vision-based drones, as well as the transfer of the learned controller to outdoor scenarios where ground truth positions will be obtained using RTK-capable GNSS receivers. To reduce the need for large amounts of labeled data, we are exploring recent advances in unsupervised domain adaptation to aid generalization of the neural controller to environments with background clutter. Another challenge is the addition of obstacles to the environment in which the agents operate.

Acknowledgments

We thank Olexandr Gudozhnik, Vivek Ramachandran, and Enrica Soria for their valuable contributions.

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] J.-C. Zufferey, Hauert, Sabine, Stirling, Timothy, Leven, Severin, Roberts, James, and Floreano, Dario, "Aerial Collective Systems," in *Handbook of Collective Robotics*, S. Kernbach, Ed. Pan Stanford, 2011, pp. 609–660.

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

[4] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *Int Conf Rob Autom (ICRA)*, 2017, pp. 3299–3304.

[5] A. Weinstein, A. Cho, G. Loianno, and V. Kumar, "Visual Inertial Odometry Swarm: An Autonomous Swarm of Vision-Based Quadrotors," *IEEE Robot Autom Lett (RA-L)*, vol. 3, no. 3, pp. 1801–1807, 2018.

[6] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *Int Conf Intel Rob Sys (IROS)*. IEEE/RSJ, 2014, pp. 3866–3873.

[7] C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. Somorjai, T. Nepusz, and T. Vicsek, "Flocking algorithm for autonomous flying robots," *Bioinspir Biomim*, vol. 9, no. 2, p. 025012, 2014.

[8] 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.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] 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.

[11] 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.

[12] N. Dousse, G. Heitz, F. Schill, and D. Floreano, "Human-Comfortable Collision-Free Navigation for Personal Aerial Vehicles," *IEEE Robot Autom Lett (RA-L)*, vol. 2, no. 1, pp. 358–365, 2017.

[13] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning Monocular Reactive UAV Control in Cluttered Natural Environments," in *Int Conf Rob Autom (ICRA)*, 2013, pp. 1765–1772.

[14] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *IEEE Robot Autom Lett (RA-L)*, vol. 3, no. 2, pp. 1088–1095, 2018.

[15] F. Sadeghi and S. Levine, "CAD2RL: Real Single-Image Flight without a Single Real Image," in *Rob: Sci Sys (RSS)*, vol. 13, 2017.

[16] A. Giusti, J. Guzzi, D. C. Cireşan, F. L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robot Autom Lett (RA-L)*, vol. 1, no. 2, pp. 661–667, 2016.

[17] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness," in *Int Conf Intel Rob Sys (IROS)*, 2017, pp. 4241–4247.

[18] 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," *Auton Robots*, vol. 41, no. 4, pp. 919–944, 2017.

[19] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, and M. Mejail, "A Practical Multirobot Localization System," *J Intell Robotic Syst*, vol. 76, no. 3-4, pp. 539–562, 2014.

[20] 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.

[21] M. Saska, J. Vakula, and L. Přeućil, "Swarms of micro aerial vehicles stabilized under a visual relative localization," in *Int Conf Rob Autom (ICRA)*, 2014, pp. 3570–3575.

[22] 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.

[23] S. Ross, G. Gordon, and D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *Int Conf Artif Intel Stat (AISTATS)*, vol. 14. JMLR.org, 2011, pp. 627–635.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Int Conf Comp Vis (ICCV)*, 2015, pp. 1026–1034.

[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Int Conf Learn Repr (ICLR)*, 2014.

[26] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *Int Conf Rob Autom (ICRA)*, 2015, pp. 6235–6240.

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *Int Conf Comp Vis (ICCV)*, 2017, pp. 618–626.