I2SL: Learn How to Swarm Autonomous Quadrotors Using Iterative Imitation Supervised Learning

Omar Shrit, Michèle Sebag

I2SL: learn how to swarm autonomous quadrotors using Iterative Imitation Supervised Learning.

Omar Shrit and Michèle Sebag

Université Paris Saclay
Rue Joliot Curie, 91190 Gif-sur-Yvette, France
firstname.lastname@universite-paris-saclay.fr

Abstract In this paper, a decentralized controller for a quadrotor swarm is presented following the leader-follower principle. The quadrotors embedding the decentralized controller follow a remotely controlled leader. The controller, governing the behavior of a set of followers is learned using an Iterative Imitation supervised learning approach. The novelty of this approach is to build complex policies supporting the flocking behavior for a set of quadrotors while requiring only COTS (Commercial Off The Shelf) wireless sensors. In the first iteration, a set of trajectories is generated using the well-known Reynolds flocking model (adapted by Schilling et al, 2018, to add a migration term); the logs are exploited to enable the follower quadrotor controller to achieve the migration function. In the further iterations, the learned controller is exploited in combination with the Reynolds model; the logs generated are then exploited to learn a follower quadrotor controller achieving both the migration and the flocking functions, as robust as the Reynolds model. The validation of the approach using a Software In The Loop (SITL) environment relying on the Gazebo simulator, confirms that the learned controller enables the followers to accurately follow the leader while collectively satisfying the swarm properties.

Simulation videos

Available at https://tinyurl.com/7b2f7mcz

1 Introduction

It is well known that animals, e.g., insects, birds or mammals [1], can be organized in various ways, having either one leader [2], a hierarchy of leaders [3], or no leader at all [4]. The objective of this organization is to create emergent behaviors. By monitoring these behaviors, an observer can deduce that these animals are trying to achieve a synchronous movement also known as collective motion in order to force back the danger from a predator [5] or to migrate from one zone to another [6]. The collective motion itself is a result of a collective behavior [7], in which this complex behavior is composed of simple interactions between the agents. While collective behavior aims to

fulfil a common goal, this goal depends on the context. For example, a school of fishes can change their direction immediately if they face an instant danger; a group of bees can attack directly the predators in order to save the nest and the group. Flocking and swarming behaviors are not limited to mammals and birds. It is well observed at the micro-biological level, as it is the case for cells [8] and bacteria [9]. Overall, a swarm can be defined as a group of similar agents interacting with each other in order to create emergent behavior.

Related works. The above observation is at the root of analytical models, enabling us to understand and mimic the behaviors of animals and insects. Several flocking models have been developed in the last decades such as the Reynolds model [10], the Vicsek model [11] or the Olfati-Saber model [12]. When tuned correctly, these models can achieve the flocking behavior perfectly. The issue with these models arises when deploying them directly on a multi-agent system such as quadrotors. These models rely on precise localization which is rarely possible in real-life scenarios. Therefore, several adaptations are required. Viragh et al [13] adapt the Viscek model by adding several parameters such as the inner noise of sensors, inertia, time delay, and communication constrained in order to enhance the behavioral law. They test this model on a set of quadrotors while using GNSS as a localization system [14]. To improve their outdoor swarms, they use an evolutionary algorithm to find the best flocking parameters for an outdoor swarm of 30 quadrotors [15].

Another approach relies on the use of vision onboard sensors instead of GNSS systems. Vision-based swarms can follow a true decentralized controller for a set of quadrotors, that is not based on external localization systems. Schilling et al [16] embed six cameras on each quadrotor in order to provide 360 degrees vision of the environment. They adapt the Reynolds flocking model by adding a migration term. Their method, deployed on real quadrotors in indoor [17] and outdoor [18] environments, relies on the prediction of the velocity command using supervised learning.

However, it is challenging to deploy these methods on small and nano quadrotors [19] because 1) they require the use of heavy sensors, such as a complete set of cameras, or expensive GNSS; 2) the addition of these sensors will require a considerable amount of onboard computation which is rarely available on nano quadrotors; 3) the outdoor performance is related to the weather condition, since GNSS improves performance on sunny days, and similarly, vision sensors perceive better their neighbors with good lighting conditions.

Contributions. In this paper, we present I2SL (*Iterative Imitation Supervised Learning*), a supervised iterative imitation learning method that is inspired from [20,16,21]. The contributions of this method can be described in two perspectives. From the control perspective: this method addresses the challenge of the design of a decentralized swarm controller for small quadrotors using imitation learning. The use of imitation learning is favored in this case since the existence of an oracle demonstrator (flocking model), eliminating the need for reinforcement learning since it requires the design of a specific reward function that can used for all agents. In addition, several researchers [22,23] have demonstrated that imitation learning can be combined successfully in the case of multi-agent system in order to learn a specific policy. From the perception perspective:

our aim is to learn a decentralized controller using only one wireless sensor on each quadrotor, with very limited computation on-board. The approach has been validated on MagicFlock [1], a home-made SITL framework based on RotorS [24] and extended to model a swarm (as RotorS supports only one quadrotor). The choice of developing this framework is related to the limited access to real quadrotors hardware.

The presented work extends the IL4MRC method [21], which likewise aims to achieve a decentralized controller with cheap embedded wireless sensors. The contribution is based on the combination of IL4MRC with Dagger, along with an iterative approach, gradually refining the controller learned in the former iteration, thereby exploring only the necessary information rather than exploring the entire environment. In the first iteration, the leader is assigned a random model to simulate a human pilot, while the followers are using a flocking model, and each quadrotor generates logs describing its state as a vector of sensor values. The controller learned from these logs immediately enforces the following of the leader, i.e. it supports the *migration* function. but does not enforce the *Cohesion* and *Separation* rules in order to avoid one another. In the second iteration, the former controller is used in alternation with the flocking policy following the Dagger approach. The obtained trajectories thus alternate between avoiding any possible collision and following the leader. After several iterations, while each quadrotor is controlled from its embedded controller, they collectively 'swarm' around the (remotely controlled) leader, i.e. they satisfy the main swarm properties: i) following the leader, ii) avoiding collision and preventing separation of the neighbor followers.

Formally, the presented approach makes the following contributions:

- The learning procedure combines agile iterative imitation learning with a multi-agent system in order to create a decentralized swarm controller for quadrotors;
- It learns a decentralized controller, that requires only one cheap wireless sensor and very limited computational on-board resources.

This paper is organized as follows. Section 2 introduces the formal definition of the flocking model and presents an overview of I2SL. Section 3 gives a proof of concept for the accuracy of the proposed method. Sections 4 and 5 respectively present the experimental setting and the experimental validation of I2SL. Section 6 concludes the paper with some perspectives for further research.

2 Methodology

This section introduces the flocking model used in the paper for the sake of self-contentedness and presents the I2SL approach.

2.1 Flocking algorithm

This subsection presents the adaptation of the Reynolds flocking model, as described in [16], except for its ability to handle the leader-follower mechanism. Specifically,

[1] https://github.com/shrit/MagicFlock

the considered swarm comprises a *piloted* leader and a set of followers that follow the leader. The leader is remotely controlled and is not aware of the followers. The followers' objective is to migrate toward this leader. Similar to [16], we omit the velocity matching term since the quadrotors do not communicate with one another, and do not have additional sensors to estimate neighbors' velocity reactively. Most generally the flocking model involves three terms, known as *cohesion*, *separation*, and *migration*. Let $\mathcal{N}_i$ denote the set of neighbor follower quadrotors of the quadrotor i, with

$$\mathcal{N}_i = (follower\ j : j \neq i \wedge \|\mathbf{r_{ij}}\| < r^{max}) \tag{1}$$

where $\|.\|$ is the euclidean norm. $\mathbf{r_{ij}} \in \mathbb{R}^3$, $\mathbf{r_{ij}} = \mathbf{p}_j - \mathbf{p}_i$ denotes to the relative position of quadrotor j with respect to follower i. Only one set of quadrotors is considered at a time; they can not divide themselves into several sets as they all need to stay close to the leader. Formally, the swarm is said to be valid as long as $d_{ij} < 30$ meters. The three terms cohesion, separation, and migration work together to produce the flocking behavior; the separation term pushes away agents that are close to each other to avoid a collision; inversely the cohesion term moves far away quadrotors toward their nearest neighbors. Both terms work together in order to provide a consistent swarm behavior.

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

$$\mathbf{v_i}^{coh} = \frac{k^{coh}}{\mathcal{N}_i} \sum_{j \in \mathcal{N}_i} \mathbf{r}_{ij} \tag{3}$$

Where k^{sep} is the separation gain, k^{coh} is the cohesion gain. The sum of two velocities (Cohesion and Separation) produce the Reynolds velocity $\mathbf{v_i}^{rey} = \mathbf{v_i}^{sep} + \mathbf{v_i}^{coh}$. In addition to the above terms, the migration terms allows the followers quadrotors to move toward the leader quadrotor permanently. The migration point is not fixed: it is the position of the leader itself. The migration term is given by:

$$\mathbf{v_i}^{mig} = k^{mig} \frac{\mathbf{r}_{ij}}{\|\mathbf{r_{ij}}\|} \tag{4}$$

Where k^{mig} is the migration gain and $\mathbf{r}_{ij} \in \mathbb{R}^3$ is the relative position of the migration point w.r.t. the i-the quadrotor, with $\mathbf{r}_i^{mig} = \mathbf{p}^{leader} - \mathbf{p}_i^{follower}$. In order to achieve the flocking behavior, the controller embedded on each follower uses the sum of the tree velocity commands $\mathbf{v_i} = \mathbf{v_i}^{sep} + \mathbf{v_i}^{coh} + \mathbf{v_i}^{mig}$. The leader is normally operated by a human pilot, with a limited velocity (racing tasks are not considered in the following). In simulation, the maximum speed of the flocking model is bounded to a maximum final velocity command, set to $v_{max} = 2m/s$, and the velocity of each follower $\mathbf{v}_i$ is accordingly bounded as:

$$\mathbf{v}_i = \min(\|\mathbf{v}_i\|, v_{max}) \tag{5}$$

2.2 Iterative imitation supervised learning

The proposed approach takes inspiration from [21]. The extension aims to relieve the simplifying assumptions of discrete action space and a good initial condition of the swarm. We show that using the flocking policy to generate the logs that will serve to the imitation-based controller learning relieves the need for such simplifying assumptions. Formally, the proposed iterative imitation approach uses a 3-step process inspired from the Dagger algorithm [20].
At the iteration i:

- The trajectory executed by each quadrotor is logged, defined as a sequence of states s_t and actions a_t. This trajectory is generated after controller π_i using the Dagger mechanism:
$$\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_{i-1} \tag{6}$$
Where π^* is the flocking policy, and $\hat{\pi}_{i-1}$ is the policy learned in the last iteration. β is decayed exponentially from 1 to 0 over time as $\beta = e^{-\lambda i}$ where $\lambda = 0.69314$ is a constant.
- The trajectories generated in the above are logged to form a training dataset $\mathcal{E}_i$:
$$\mathcal{E}_i = \{(s_t, a_t), t = 1, \dots T\} \tag{7}$$
and the datasets are stacked:
$$\mathcal{E} \leftarrow \mathcal{E}_i + \mathcal{E}_{i-1} \tag{8}$$
- The model $\mathcal{F}_i$ is trained from $\mathcal{E}$ to learn the best action to execute based on the sequence of the last states:
$$\mathcal{A} = \mathcal{F}_i(\mathcal{S}) \tag{9}$$

The state of each quadrotor is defined as a vector of the signal strength the azimuth and the elevation angles perceived from neighbors $i = 1, ..., n$. Therefore, at each time step t the state of the quadrotor j is given as $s_t^j = (rss_1, \phi_1, \theta_1, rss_2, \phi_2, \theta_2, ..., rss_n, \phi_n, \theta_n)$. While the action a_t^j is the velocity vector $\mathbf{v}$.

3 A proof of principle of I2SL: Application to quadrotors control

This section presents a proof of principle of I2SL,[2] applied to the control of a set of quadrotors. After describing the position of the problem, the algorithmic pipeline (data acquisition phase, training of the model, exploitation of the model) is detailed.

3.1 Position of the problem

Considering a set of quadrotors with one leader and several followers, the goal is to gradually build an independent controller for each follower, knowing that each follower has very minimal sensing capabilities, such as measuring the distance to its neighbors, and receive the azimuth and elevation angles of its neighbors. The objective of this

[2] https://github.com/shrit/MagicFlock

controller is to achieve the flocking behavior known as swarming for the followers' quadrotors, where the leader quadrotor is manually controlled by a human pilot.

Following [21], the main goal of the proposed approach is to achieve some trade-off between efficiency and computational and other resources. On the one hand, the cost of sophisticated sensors is an issue. On the other hand, quadrotors consume a considerable amount of energy when carrying heavy sensors, not to mention the algorithmic complexity to analyze the perceived data from the environment.

In the simulation, Gazebo provides a generic wireless sensor that can be added to each quadrotor, we have integrated 3 antennas on each robot. The wireless sensor considers obstacles in the nearby of each robot which affect the value of the Received Signal Strength (RSS) according to the number and density of the obstacles. The proposed method does not require sharing information, which means there are no communications between the agents. Therefore, the wireless channel is not used only RSS values are perceived from neighbors. In addition, for the only sake of simulation, we add a ray sensor on each quadrotor to provide angle estimation (azimuth and elevation) to neighbors. During simulation as provided by the link in the abstract to experiment videos, we have turned off ray visualization to remove heavy computations from GPU. However, in a real-life scenario, one can use COTS quadrotors that have an embedded WiFi card such as Intel 5300 with 3 antennas allowing to estimate Angle of Arrival (AoA) of signals and the RSS values from neighbors, more details are discussed in section 6.

Each robot is capable of mapping distances and angles to its neighbors. In this case, one might argue that robots can create a polar coordinate system thus constructing gradually a relative localization system. Indeed, this system can be used directly by the flocking model and remove the need for imitation learning. Arguably this is true. However, there are two disadvantages to this method. First, this will require more computation from each robot, since it needs to calculate the relative position of neighbors in each time step and then apply the calculations related flocking model. Second, the sensor noise needs to be estimated before the flight, and proper modifications have to be applied to the flocking model accordingly. However, imitation learning alleviates the need for such a calibration as the noise is embedded inside the data. The learned controller has a better estimation of its neighbors allowing it to perform as well as the oracle flocking model as demonstrated in section 5.

Quadrotor settings: Formally we have defined one setting of quadrotors to train and test the models on the quadrotors. We have a set of 7 quadrotors, in which there are one leader and 6 followers. The goal is to embed the **same** trained controller on all the followers to *follow* the leader without having any collision with the neighbor quadrotors.

3.2 Data acquisition

During the data acquisition, the state of each follower is recorded along with a set of episodes. Each follower registers two data sets simultaneously. The first data set registers the states of the leaders along with the migration velocity as given by the flocking model. The second data set registers the state of the other followers' neighbors along with the Reynolds velocity as given by the flocking model.

First iteration Each episode starts with quadrotors taking off. Once the taking off has finished, the leader chooses a direction randomly and moves in this direction for 80 seconds. The follower quadrotors use the flocking model and start following the leader. The max velocity of the leader is equal to 0.7 meters per second which are slightly lower than the followers equal to 1.0 meters per second. This small variation allows the followers to catch up with the leader. The episode ends once the followers are close to the leader and there is no longer any change in their distances. To ensure the diversity of the collected data set, all the quadrotors are reset into a new position at the end of each episode, allowing each quadrotor to have a different set of neighbors.

Second iteration Similar to the first episode, the episode starts by taking off, and then the leader moves before the other quadrotors. The main difference is that the value of β is reduced from 1 to 0.5 in this iteration allowing alteration between the flocking model and model trained in the first iteration.

Third and further iterations The following iterations follow the same principle in the second iteration while continuing to reduce β as described in 2.

3.3 Forward model

The data set is exploited to learn a forward model. The forward model uses the last states in order to predict the action to execute at this time step. Formally, the data set is decomposed as a set of the last five states, that are trained to predict the action a_t.

$$(X = (s_{t-4}, s_{t-3}, s_{t-2}, s_{t-1}, s_t); Z = a_t) \tag{10}$$

We use a mainstream supervised learning algorithm to train a function $\mathcal{F}$ such that $\mathcal{F}(X) = Z$ from 80% of the data.

During the training, the quality of the model $\mathcal{F}$ is estimated by applying the model on the validation set which comprises 20% of the data set.

3.4 I2SL controller

At production time, model $\mathcal{F}$ is used as the decentralized controller that is embedded on each quadrotor. The model is composed of two submodels, the first model is trained on the data set that is based on sensor value received from the leader while the second model is trained on data set received from the neighbors followers.

$$a_t^{(1)} = \mathcal{F}^1(s_{t-4}, s_{t-3}, s_{t-2}, s_{t-1}, s_t) \tag{11}$$

$$a_t^{(2)} = \mathcal{F}^2(s_{t-4}, s_{t-3}, s_{t-2}, s_{t-1}, s_t) \tag{12}$$

$$a_t^* = a_t^{(1)} + a_t^{(2)} \tag{13}$$

The quadrotors are operated at production time very similarly as in the data acquisition phase. In each episode, the quadrotors take off, the leader is randomly operated. The leader starts moving 5 seconds before the followers, in order to allow for the followers

to accumulate a decent amount of states from the sensor in order to predict the good action. The episode runs as long as the flocking behavior is maintained and neither collision nor separation is noticed. As said, we consider that the flocking is maintained as long as the distance between two quadrotors $d_{ij} < 30$ meters.

4 Experimental setting

This section describes the goal of experiments and the experimental setting used to validate the I2SL approach.

4.1 Goals of experiments

As said, an episode starts with the swarm taking off. Once the taking off has finished the leader starts moving 5 seconds before the followers. We create a ZigZag experiment that allows testing if the learned models are capable of imitating the flocking behavior, and how the followers are behaving when the leader changes its direction from time to time.

Such experiments aim to simulate a human pilot flying the leader quadrotor through a specific trajectory while the followers keep appropriate distances among all of them.

The straightforward performance indicator is to measure the minimum and the maximum distance between the follower quadrotors during the flight; these indicators reflect the consistency of the learned flocking behavior. The distance metric is given by:

$$d^{min} = \min_{i,j \in \mathcal{N}} \|\mathbf{r}_{ij}\| \tag{14}$$

$$d^{max} = \max_{i,j \in \mathcal{N}} \|\mathbf{r}_{ij}\| \tag{15}$$

Where $\mathcal{N}$ if the set of follower quadrotors and d^{min}, d^{max} is respectively the minimum and the maximum distance observed in the follower quadrotors swarm. In addition to the indicators, we show the trajectory executed by the leader and the followers for each iteration and each experiment.

4.2 Baseline

To assess the performance of I2SL we used the flocking model that uses the absolute positioning system. The flocking model (oracle) delivers the perfect flocking behavior when knowing the exact position of all the neighbor followers. The gain of the flocking model have been chosen as described in [16], since they modulate the strength of the cohesion and the separation of the swarm.

The behavior in each iteration is compared with the behavior obtained in the former iteration, and with the flocking model.

4.3 Simulation platform

In the experiments, the system includes seven robots of the same type, the IRIS quadrotor designed by 3DR[3], with height 0.11m, width 0.47m, and weight 1.5 kg. The quadrotors are simulated using the software in the loop simulation (SITL) [24] integrating the Gazebo simulator [4]. Each quadrotor uses PX4 [5] as an autopilot software.

The maximum velocity that a quadrotor can reach is 1m/s, The take-off altitude is 45 m.

4.4 Learning of the flocking model

The total training data set for all the iterations records at least 24h of flying time. The flocking model is implemented as a neural net, using mlpack [25], while the linear algebra library is Armadillo [26]. The neural architecture is a 2-hidden layers, with 256 neurons on each layer and Sigmoid as activation function. The training uses Glorot initialization [27], with .5 Dropout and batch size 32; the hyper-parameters are adjusted using Adam [28] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and initial learning rate $\alpha = 0.001$.

5 Empirical validation

5.1 Zigzag experiment

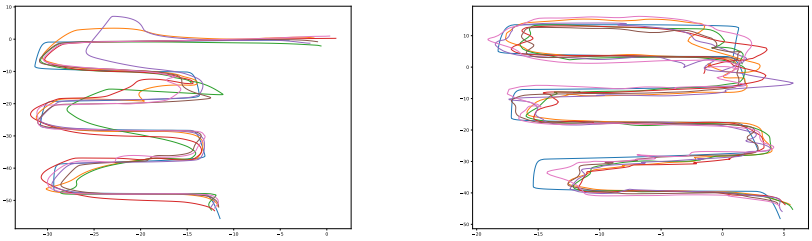
The zigzag experiment aims to assess whether the followers are capable to imitate the flocking behavior. The experiment runs as follows: all the quadrotors take off, the leader follows a random trajectory for 5 seconds and then follows the zigzag trajectory that is already embedded on the leader. The followers do not know about their leader's trajectory, their objective is to follow the leader and avoid collision and dispersion. The same experiment is executed on each iteration, allowing us to validate the learned model in each iteration.

We compare the first and the second iteration in figures 1 2. In both iterations, the trained controller uses the data perceived by the wireless sensor. The first iteration uses the classic imitation learning algorithm, while the second iteration applies the Dagger approach. The result 1 (left) shows the trajectory executed by each quadrotor, while the inter-quadrotor distances is shown in figure 2 (left). In the first iteration, we observe that the quadrotors learn how to follow the leader. However, they do not learn how to respect distances among them, resulting in several minor collisions between the quadrotors and distortion in the executed trajectory; this is confirmed in figure 2 for this experiment. In the second iteration (right) quadrotors start to learn how to avoid each other, but their behavior is very aggressive, and not refined yet to be similar to the flocking model. In addition, in both iterations, no collision was observed between the followers.

We continue to train the controller iteratively, resulting in a third iteration with a performance similar to the flocking model in figure 3. The controller uses the wireless

[3] https://3dr.com/
[4] http://gazebosim.org/
[5] https://px4.io/

sensor data, while the flocking model using the absolute position acting as ground truth for swarming behavior. We observe that the followers' quadrotors respect distances among each other in a similar manner compared to the flocking model in figure 4.



(a) Iteration 1 (sensor based): trajectories executed by all quadrotors

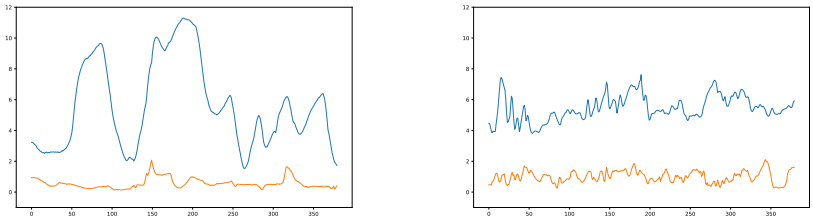
(b) Iteration 2 (sensor based): trajectories executed by all quadrotors

Figure 1. This figure compares the trajectories executed during the zigzag experiment by the quadrotors in both the first and the second iteration. All the quadrotors take off from the (0,0) coordination, and they land at (11, -55) in the first iteration, and at (5, -44) in the second iteration. The leader quadrotors labeled in blue have an integrated embedded trajectory to simulate a human pilot, while all the followers use the learned controller based on a cheap wireless sensor. This figure shows an improved trajectory in the second iteration. This is due to the usage of the iterative learning technique over the basic imitation supervised learning represented in the first iteration.

6 Discussion and future work

In this paper, we presented I2SL, an Iterative Imitation Supervised Learning method used in order to resolve the challenge of decentralized controller design for a set of quadrotors with no computational power and endowed with a single wireless sensor. The objective of this method is to resolve the optimization issue offline rather than during the flight and to learn the flocking behavior for follower quadrotors while following the remotely controlled leader. This approach demonstrated the feasibility of a leader-followers swarm using MagicFlock, a Software In the Loop (SITL) simulation framework that is based on RotorS.

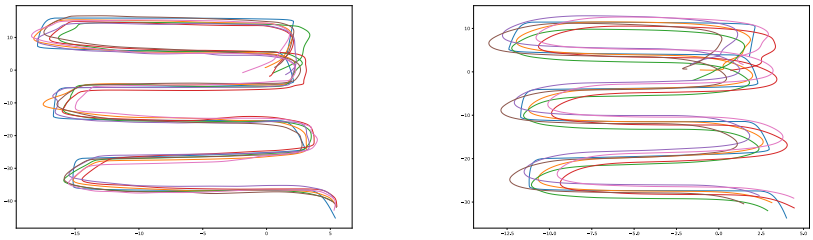
Wireless antennas are often intended to be used as a communication tool using the radio channel. One might consider this usage in order to share state information and leader commands between quadrotors. This method might work when the number of quadrotors is small. However, when the number of the agent increases, the communication channel tends to saturate, with a high loss of the packets sent between the transmitter and receiver. A possible solution would be is a re-transmission, but this might increase the communication delay. In both cases, the power consumption increases due to the frequent treatment of the sent or the received packets. Therefore, it is easier to



(a) Iteration 1 (sensors based): max and min inter-quadrotor distances

(b) Iteration 2 (sensor based): max and min inter-quadrotor distances

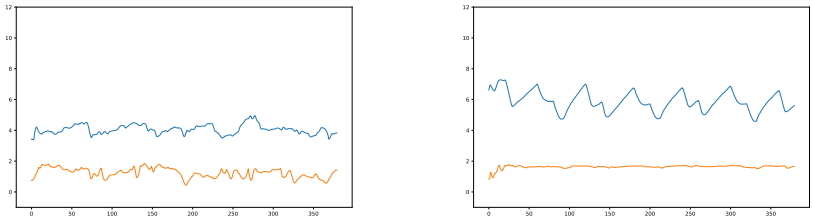
Figure 2. This figure shows the inter-quadrotors distances among the followers for both trajectories executed in iteration 1 (left) and iteration 2 (right). The blue line shows the maximum inter-quadrotor distances while the orange one shows the minimum distance. We observe a considerable improvement in the second iteration comparing it the first one, as the decentralized controller has learned the cohesion and separation policy in the second iteration. The quadrotors remain collision-free in the second iteration and do not disperse. Knowing that in the first iteration we observe minor collisions but non of these collisions were not critical allowing us to complete the experiment.



(a) Iteration 3 (sensor based): trajectories executed by all quadrotors

(b) Flocking model (position-based): trajectories executed by all quadrotors

Figure 3. This figure shows the trajectory executed by the quadrotor in the third iteration (left) compared to the adapted Reynolds flocking model (right). The quadrotors start their trajectory at coordination (0,0) and end at (5, -44). The leader is labeled in blue in both cases. By comparing the two trajectories, we can observe a similar performance between the flocking model (position-based) and the third iteration of the learned controller (wireless sensors-based).



(a) Iteration 3 (sensor based): max and min inter-quadrotor distances

(b) Flocking model (position based): max and min inter-quadrotor distances

Figure 4. Results of inter-agent distances when executing the zigzag trajectory by the quadrotors. The controller from the third iteration shows a similar performance compared to the flocking model. The quadrotors do not disperse nor collide with one another. These results show that the controller can be improved iteratively in order to achieve a performance compared to the flocking model.

have only one pilot that communicates with only the leader quadrotor, while the followers use the trained embedded controller and the wireless sensor to swarm around the leader.

When reading this work, one might question the type of wireless sensor that can be used with these robots. To this end, any available wireless communication tools that allow the estimation of Angle of Arrival (AoA) and signal strength can be used, such as Bluetooth, Zigbee, Ultra-Wideband (UWB). Among these wireless tools, we have considered using WiFi for several reasons, First, most commercial quadrotors are already embedded WiFi antennas, removing the cost for additional sensors that need to be embedded on each robot. Second, most integrated WiFi cards have several antennas installed on these robots allowing to estimate AoA of the received radio signal and therefore the direction of the emitter [29,30]. Most generally, the method for AoA estimation on COTS WiFi cards is well detailed. First, we need to extract the Channel State Information (CSI) [31] since it contains the phase information and the signal strength for all OFDM subcarriers. Second, we need to analyze the signal phase since it suffers a shift and attenuation when the signal propagates in the environment. Finally, by analyzing this shift over all of these subcarriers using for instance the MUSIC algorithm [32], one can easily deduce the AoA of the signal. This method can be used with a commodity WiFi card such as Intel 5300 since it has several antennas that can be arranged to a uniform antennas array.

To demonstrate the capacity of precise AoA estimation. Several researchers went even further by analyzing and smoothing the CSI values to create a decimeter localization systems [33,34] or even a system that has a similar performance to the ground truth [35]. However, the goal of future work is not to create a localization system for quadrotors nor to use fixed Access Points (AP) that act as beacons but instead is to use AoA estimation techniques and integrate them directly on real quadrotor platforms that are equipped with such antennas.

Finally, our further research will focus on transfer learning, from simulation to real quadrotors. Besides, we will adapt the designed method and find a suitable AoA estimation technique for real pico-quadrotors. In the medium term, our goal is to achieve outdoor swarms entirely based on wireless antennas acting as bearing and heading sensors.

References

1. I. Lebar Bajec and F. Heppner, "Organized flight in birds," *Animal Behaviour*, vol. 78, pp. 777–789, 11 2012.
2. I. Couzin, J. Krause, N. Franks, and S. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, pp. 513–6, 03 2005.
3. M. Nagy, Z. Akos, D. Biro, and T. Vicsek, "Hierarchical group dynamics in pigeon flocks." *Nature*, vol. 464, no. 7290, pp. 890–893, 2010-04-08 00:00:00.0.
4. I. AOKI, "A simulation study on the schooling mechanism in fish," *NIPPON SUISAN GAKKAISHI*, vol. 48, no. 8, pp. 1081–1088, 1982.
5. G. Beauchamp, "Group-size effects on vigilance: a search for mechanisms," *Behavioural processes*, vol. 63, pp. 141–145, 08 2003.
6. D. Grossman, I. S. Aranson, and E. B. Jacob, "Emergence of agent swarm migration and vortex formation through inelastic collisions," *New Journal of Physics*, vol. 10, no. 2, p. 023036, feb 2008. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2F10%2F2%2F023036
7. T. Vicsek and A. Zafeiris, "Collective motion," *Physics Reports*, vol. 517, no. 3, pp. 71–140, 2012.
8. X. Trepat, M. Wasserman, T. Angelini, E. Millet, D. Weitz, J. Butler, and J. Fredberg, "Physical forces during collective cell migration," *Nature Physics*, vol. 5, pp. 426–430, 05 2009.
9. Y. Wu, A. D. Kaiser, Y. Jiang, and M. S. Alber, "Periodic reversal of direction allows myxobacteria to swarm," *Proceedings of the National Academy of Sciences*, vol. 106, no. 4, pp. 1222–1227, 2009.
10. C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Aug. 1987. [Online]. Available: http://doi.acm.org/10.1145/37402.37406
11. T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical Review Letters*, vol. 75, no. 6, p. 1226, 1995.
12. R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.
13. C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szörényi, G. Somorjai, T. Nepusz, and T. Vicsek, "Flocking algorithm for autonomous flying robots," vol. 9, 10 2013.
14. G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," *CoRR*, vol. abs/1402.3588, 2014.
15. G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, 2018.
16. F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based cohesive flight in drone swarms," *CoRR*, vol. abs/1809.00543, 2018. [Online]. Available: http://arxiv.org/abs/1809.00543
17. ——, "Learning vision-based flight in drone swarms by imitation," *CoRR*, vol. abs/1908.02999, 2019. [Online]. Available: http://arxiv.org/abs/1908.02999

18. F. Schilling, F. Schiano, and D. Floreano, "Vision-based flocking in outdoor environments," 2020.
19. Y. Mulgaonkar and V. Kumar, "Towards open-source, printable pico-quadrotors," 2014.
20. S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," 2011.
21. O. Shrit, D. Filliat, and M. Sebag, "Iterative Learning for Model Reactive Control: Application to autonomous multi-agent control," in *ICARA*, Prague, Czech Republic, Feb. 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03133162
22. R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1534–1539.
23. H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1995–2003.
24. F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *RotorS—A Modular Gazebo MAV Simulator Framework*. Cham: Springer International Publishing, 2016, pp. 595–625.
25. R. R. Curtin, J. R. Cline, N. P. Slagle, W. B. March, P. Ram, N. A. Mehta, and A. G. Gray, "Mlpack: A scalable c++ machine learning library," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 801–805, Mar. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2502581.2502606
26. C. Sanderson, "Armadillo: C++ template metaprogramming for compile-time optimization of linear algebra," *Computational Statistics and Data Analysis*, vol. 71, 2014.
27. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.
28. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980
29. A. Paulraj, V. U. Reddy, T. J. Shan, and T. Kailath, "Performance analysis of the music algorithm with spatial smoothing in the presence of coherent sources," in *MILCOM 1986 - IEEE Military Communications Conference: Communications-Computers: Teamed for the 90's*, vol. 3, 1986, pp. 41.5.1–41.5.5.
30. D. Niculescu and B. Nath, "Vor base stations for indoor 802.11 positioning," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 58–69. [Online]. Available: https://doi.org/10.1145/1023720.1023727
31. D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release: Gathering 802.11n traces with channel state information," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, p. 53, Jan. 2011. [Online]. Available: https://doi.org/10.1145/1925861.1925870
32. R. Schmidt, "Multiple emitter location and signal parameter estimation," 1979.
33. M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "Spotfi: Decimeter level localization using wifi," ser. SIGCOMM '15, New York, NY, USA, 2015. [Online]. Available: https://doi.org/10.1145/2785956.2787487
34. J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 71–84. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xiong
35. M. Kotaru and S. Katti, "Position tracking for virtual reality using commodity wifi," 2017.