Towards Drone Flocking using Relative Distance Measurements

Andreas Brandstätter[1][0000−0003−2820−4446], Scott A. Smolka[2], Scott D. Stoller[2][0000−0002−8824−6835], Ashish Tiwari[3][0000−0002−5153−2686], and Radu Grosu[1][0000−0001−5715−2142]

[1] Research Division for Cyber-Physical Systems, TU Wien, Austria
andreas.brandstaetter@tuwien.ac.at
[2] Department of Computer Science, Stony Brook University, USA
[3] Microsoft, USA

Abstract. We introduce a method to form and maintain a flock of drones only based on relative distance measurements. This means our approach is able to work in GPS-denied environments. It is fully distributed and therefore does not need any information exchange between the individual drones. Relative distance measurements to other drones and information about its own relative movement are used to estimate the current state of the environment. This makes it possible to perform lookahead and estimate the next state for any potential next movement. A distributed cost function is then used to determine the best next action in every time step. Using a high-fidelity simulation environment, we show that our approach is able to form and maintain a flock for a set of drones.

Keywords: Drones · Quadcopters · Flock · Swarm · Distributed controller.

1 Introduction

Flocking is a fundamental flight-formation problem. Birds flock for a variety of reasons, including foraging for food, protection from predators, communal warmth, and for mating purposes. Starling flocks can also perform high-speed pinpoint maneuvers, such as a 180° turn [1]. Some types of flocks in nature have distinct leaders, such as queen bees, and queen ants. Other swarms are formed by animals that do not have a permanently defined leadership, such as starlings or herrings. Although flocking is a well-studied problem mathematically [19, 15, 8, 7], its realization using actual drones is not nearly as mature (but see [26, 23]).

Drone swarms, a quintessential example of a multi-agent system, can carry out tasks that cannot be accomplished by individual drones alone [5]. They can, for example, collectively carry a heavy load while still being much more agile than a single larger drone [16, 13]. In search-and-rescue applications, a swarm can explore unknown terrain by covering individual paths that jointly cover the entire area [9, 3, 17]. While flocking provides a number of advantages over

individual flight, it also poses a significant challenge: the need for a distributed control mechanism that can maintain flock formation and its stability [18]. These collective maneuvers can be expressed as the problem of minimizing a *positional cost function*, i.e., a cost function that depends on the positions of the drones (and possibly information about their environment). In our formulation, every agent is identical, which means there is no designated leader.

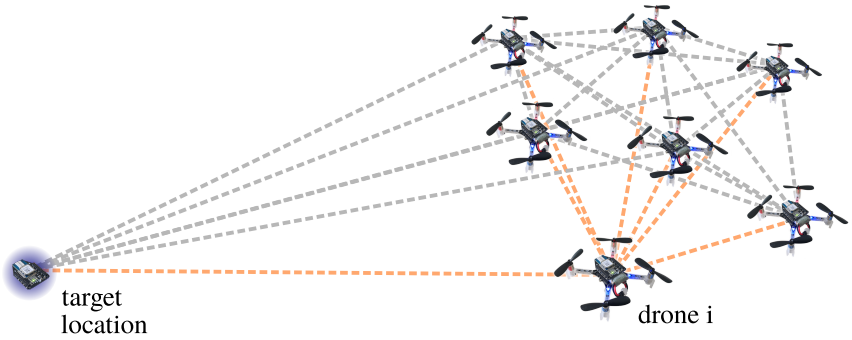

Fig. 1. Our distributed controller forms and maintains a flock based on relative distance measurements to other agents of the flock. The target location is shown in blue. Distance measurements for drone i to other drones and to the target location are shown in orange.

To work with such a positional cost function, an absolute localization system is needed. This can be an optical or radio-based system for indoor applications or GPS-based localization for outdoor scenarios. In this work, we study the problem for scenarios that lack an absolute localization system (GPS-denied environments). We only have the ability to measure the distance to other drones and to measure the acceleration and rotational velocity of the own drone using an onboard Inertial Measurement Unit (IMU). For flock formation, we observe that the positional cost function can be replaced by a function based solely on relative distances. This obviates the need for absolute localization. We propose a method to simultaneously learn properties of the environment (inter-agent distance changes), while at the same time maintaining the flock formation solely on relative distance information.

In this paper, we address the following **Challenge Problem:** *Design a distributed controller that forms and maintains a flock based solely on inter-agent distance measurements.*

To solve this problem, we introduce a method to estimate changes of the environment based on the observed changes for previous movements and thereafter use this information to minimize the cost-function over a set of candidate positions. We build upon our previous work that introduced Spatial Predictive

Control (SPC) [4] to select the best next action from the set of candidate positions. However we have a substantially different problem here, since we have limited observation capability: in the previous work [4], absolute positions of all the drones were available; whereas in this work we can only measure relative distances. This also changes our possibilities how to apply SPC: whereas in the previous work it was possible to optimize the direction based on the cost function's gradient, we need to do a search on possible candidate positions in all directions in this work.

Our agent's observations consist of its own acceleration in three-dimensional space, rotational velocity along three axes, and the relative distance to other agents, as well as the distance to a fixed target location (as shown in Figure 1). (The target location is currently only used to counteract drifting tendencies of the whole flock.) There is no communication or central coordination, which makes our approach fully distributed. Our flocking objective is formulated as a cost function (see Section 2.2) which is based on these distance measurements. The corresponding action of each agent is a relative spatial vector, to which the drone should move, to minimize its cost function's value.

Paper Outline: Section 2 describes our cost function for flocking with target seeking and related performance metrics. Section 3 introduces our method to represent environmental knowledge and thereafter describes our distributed flocking controller. Section 4 presents the results of our experimental evaluation. Section 5 considers related work. Section 6 offers our concluding remarks.

2 Drone Flocking

This section starts with background on flocking, introduces our cost function for flocking with target seeking, and then presents metrics to assess the quality of a flocking controller.

2.1 What is a Flock?

A set of agents, D, is in a flock formation if the distance between every pair of agents is range bounded; that is, the drones are neither too close to each other nor too far apart. Our approach to flock formation is based on defining a cost function such that the agents form a flock when the cost is minimized. The requirement that the inter-agent distance is range bounded is encoded as the first two terms of our cost function, namely the *cohesion* and *separation* terms shown in the next section. Note that the Reynolds rules for forming a flock [19] also include a term for aligning the drone's velocities, apart from the cohesion and separation terms. By not including velocity alignment term, we potentially allow a few more behaviors, such as circling drones, but some of those behaviors are eliminated by our third term, namely the *target seeking* term.

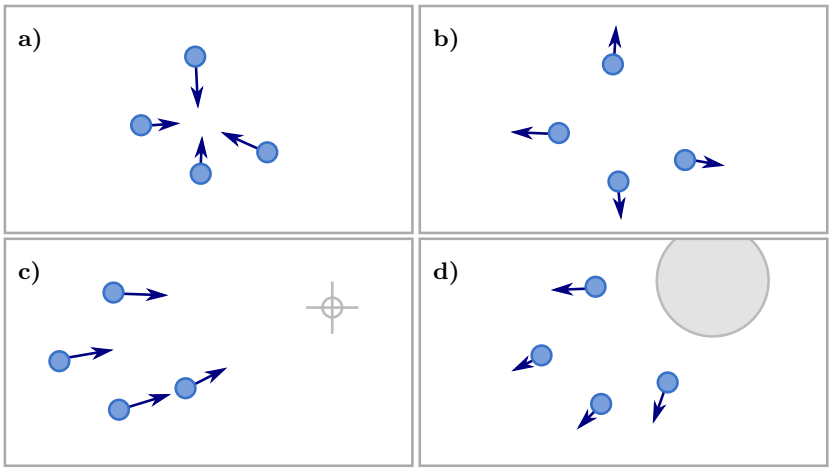

Fig. 2. Directional movements (indicated by arrows) induced by cost-function terms: **a**: *Cohesion*, **b**: *separation*, **c**: *target seeking*, and **d**: *obstacle avoidance* (not implemented in our method yet).

2.2 Cost Function

Consider drones i and j, where $i, j \in D$. Let d_{ij}, when it appears in the local cost function of drone i, denote the distance between drone i and drone j as it appears to drone i; this may differ from the actual distance due to sensing error. Similarly l_i denotes the distance between drone i and the fixed target location p_{tar}. In all cases, distances are measured from the drone's center of the mass. Let r_{drone} denote the radius of each drone (specifically the radius of the circumscribed sphere including propellers). In our formulation for the cost function, drone i has access to distances of only a subset $H_i \subseteq D$ of drones, namely its local neighbors. Hence, we define a local cost function, parameterized by i, which uses only the distances to drones in H_i. However for now we only consider the case for global neighborhood, which is $H_i = D$. We plan to extend our experiments also to local neighborhood as future work (see Section 6). Let d_{H_i} denote the tuple of distances from drone i to drones in H_i. The cost function c we use in this paper is defined for every drone $i \in D$ as in Equation (1).

$$c(d_{H_i}, l_i) = c_{coh}(d_{H_i}) + c_{sep}(d_{H_i}) + c_{tar}(l_i) \tag{1}$$

The value of the *cohesion* term increases as drones drift apart, and the *separation* term increases as drones get closer. Each term has a weight, denoted by a subscripted ω.

Cohesion term:

$$c_{coh}(d_{H_i}) = \omega_{coh} \cdot \frac{1}{|H_i|} \cdot \sum_{j \in H_i} {d_{ij}}^2 \tag{2}$$

Separation term:

$$c_{sep}(d_{H_i}) = \omega_{sep} \cdot \frac{1}{|H_i|} \cdot \sum_{j \in H_i} \frac{1}{max(d_{ij} - 2r_{drone}, \hat{0})^2} \tag{3}$$

Here $\hat{0}$ denotes a very small positive value. The function $max(., \hat{0})$ ensures the denominator remains nonzero, especially because sensor noise can cause distance measurements to have errors.

To prevent the flock from moving in random directions, we currently use a *target seeking* term with a fixed target location, denoted by p_{tar}, for the entire flock. Here l_i denotes the distance between the center of drone i and the fixed target location p_{tar}.

Target seeking term:

$$c_{tar}(l_i) = \omega_{tar} \cdot {l_i}^2 \tag{4}$$

With only *cohesion* and *separation*, the whole flock would form and move in random directions and random locations in absolute world coordinates. This would make it of limited use in any real-world scenario. Our *target seeking* term avoids this behaviour. All drones use the same target location; thus, this last term assumes shared global knowledge of the target. The control algorithm will still be fully distributed. A way to avoid having a fixed target location would be to designate one of the drones as the leader of the flock. This leader could be equipped with additional sensors to get information about its absolute position, allowing it to employ a different control scheme. We leave that investigation for future work.

2.3 Flock-Formation Quality metrics

We define two quality metrics to assess the quality of the flock formation achieved by a flocking controller. To compute these quality metrics, we assume to have access to full ground truth information about the absolute positions of the drones. The position (center of mass) of drone i is denoted by p_i.

Collision avoidance: To avoid collisions, the distance between all pairs of drones $\texttt{distance}(D)$ must remain above a specified threshold $\texttt{distance}_{thr}$. We define the metric for the minimum distance between any pair of drones as follows:

$$\texttt{distance}(D) = \min_{i,j \in D; i \neq j} \|p_i - p_j\| \tag{5}$$

$$\texttt{distance}(D) \geq \texttt{distance}_{thr} \tag{6}$$

We set $\texttt{distance}_{thr} = 2 \cdot r_{drone} + r_{safety}$, where r_{safety} is a safety margin.

Compactness: Compactness of the flock is determined by the flock *radius*. Radius is defined as the maximum distance of any drone from the centroid of the flock:

$$\texttt{radius}(D) = \max_{i \in D} \left\| \frac{\sum_{j \in D} p_j}{|D|} - p_i \right\| \tag{7}$$

The drones are said to be in a compact flock formation if $\texttt{radius}(D)$ stays below some threshold $\texttt{radius}_{thr}$; otherwise the drones are too far apart, not forming a flock.

$$\texttt{radius}(D) \leq \texttt{radius}_{thr} \tag{8}$$

The value for $\texttt{radius}_{thr}$ is picked based on the drone model and other parameters governing the flock formation problem. We set it to $\texttt{radius}_{thr} = \frac{F \cdot r_{drone}}{\sqrt[3]{|D|}}$, where we use the drone radius r_{drone} to incorporate the physical size and multiply by a factor F.

3 Distributed Flocking Controller using Relative Distances

In our distributed approach to flock formation, each drone picks the best action at every time step. The action here is a target displacement vector. Each drone picks the optimal displacement vector for itself by looking ahead in different spatial directions and finding a location that would minimize the cost *if this drone moved there*. To perform this search, each drone needs capability to *estimate* the relative distances to other drones when it moves to different potential target locations. To perform this estimation, each drone stores some measurements from past time steps, which is described in Section 3.1. Thereafter, Section 3.2 shows how this stored knowledge is used by each drone to estimate relative distances of other drones for different possible displacements of itself.

3.1 Environmental knowledge representation

We describe the procedure from the perspective of Drone i. The "environment" for Drone i consists of the current distances to the neighboring drones (and the fixed target), as this is all the information Drone i needs to evaluate the cost function. To represent the knowledge of the environment, Drone i keeps two matrices, a $(k \times 3)$-matrix N and a $(k \times (|D|+1))$-matrix P for some $k \geq 3$. The j-th row of N is a displacement vector for Drone i. The j-th row of P is a vector of change in distances of every other drone and the target to Drone i (as seen by Drone i when it moved by displacement vector in j-th row of N). In particular, P_{lj} is the change in distance of Drone j (or target if $j = |D| + 1$) as seen by Drone i when it moved by the vector N_{l*}. The notation N_{l*} denotes the l-th row vector of matrix N. Let us see how the matrices N and P are generated.

Each drone is capable of measuring its own acceleration vector in three dimensions $\vec{a_i}$. By integration, the velocity vector $\vec{v_i}$ can be derived. Drone i constructs the matrices N and P as follows:

1. Save the observations of time instant t. Let $d_{ij,t}$ denote the distances to Drone j, and let $l_{i,t}$ denote the distance to the fixed target, at this time instant t (as obtained from the sensors).

$$d_{ij,t} = d_{ij} \,|\, j \in H_i, t \tag{9}$$

$$l_{i,t} = l_i \,|\, t \tag{10}$$

2. Integrate velocity vector to keep track of its own position changes, which gives the displacement vector $\vec{u_i}$:

$$\vec{u_i} = \int_{t-\Delta t}^{t} \vec{v_i}\, dt \tag{11}$$

3. If the norm of the change in position is larger than a threshold $||\vec{u_i}|| > s_{thr}$, calculate the changes in distances as follows:

$$\bar{d}_{ij} = d_{ij,t} - d_{ij,t-\Delta t} \tag{12}$$

$$\bar{l}_i = d_{i,t} - d_{i,t-\Delta t} \tag{13}$$

   Here $d_{ij,t-\Delta t}$ denotes the observed distance of Drone j at the previous time instant $t - \Delta t$. If the length of the displacement vector is smaller than the threshold, we go back to Step (1).

4. Add the displacement vector $\vec{u_i}$ of Drone i as a row vector in matrix N and add the vector $\langle \bar{d}_{i1}, \ldots, \bar{d}_{i|D|}, \bar{l}_i \rangle$ as a row vector in matrix P. Note that we have assumed here that the neighborhood H_i of Drone i is the full set D, but the details can be easily adapted to the case when $H_i \subset D$.

5. The process starts again at (1) and we thus keep adding rows to the matrices N and P.

In this way, the matrix P reflects the available knowledge of how the distances to other drones and to the target change when Drone i moves along vector $\vec{u_i}$. Note that this data gets stale as time progresses, and the newly added rows clearly have more relevant and current information compared to the rows added earlier. Furthermore note that $\vec{u_i}$ is obtained by double integration and therefore it is prone to acceleration sensing errors, and also numerical errors. This influence is however limited, since integration times Δt are also small.

When the procedure above is followed, the matrices N and P keep getting bigger. Let N_{l*} denote the l-th row vector of matrix N. Let N_{a*}, N_{b*}, N_{c*} denote three displacement (row) vectors taken from the (most recent rows in) matrix N such that they are linearly independent – that is, they are all different from each other ($N_{a*} \neq N_{b*} \neq N_{c*}$), nonzero ($N_{a*} \neq \vec{0}$, $N_{b*} \neq \vec{0}$, $N_{c*} \neq \vec{0}$), and not in a common plane ($(N_{a*} \times N_{b*}) \cdot N_{c*} \neq \vec{0}$). These three vectors form a basis in the three-dimensional space. Using a basis transform it is therefore possible to estimate the change for distances for any movement vector $\vec{u}$. Specifically, if

$$\vec{u} = \lambda_a \cdot N_{a*} + \lambda_b \cdot N_{b*} + \lambda_c \cdot N_{c*} \tag{14}$$

then we can compute the estimated change in distances of each of the other drones, $\bar{d}(\vec{u})$, and the target, $\bar{l}(\vec{u})$ for this displacement $\vec{u}$ as follows:

$$\langle \bar{d}(\vec{u}), \bar{l}(\vec{u}) \rangle = \lambda_a \cdot P_a + \lambda_b \cdot P_b + \lambda_c \cdot P_c \tag{15}$$

(addition and multiplication in Equation (15) are applied element-wise on the vectors).

We have shown how the three vectors N_{a*}, N_{b*}, N_{c*} can be used to infer the expected change for any displacement vector $\vec{u}$ for Drone i. To ensure that three different vectors with meaningful data are present, our controller employs some optimizations in addition to the procedure described above. A special startup procedure with random movements is used to collect initial data. The three vectors (N_{a*}, N_{b*}, N_{c*}) and their associated data in P are continuously updated to avoid outdated information. However, a vector is only considered if the threshold s_{thr} is exceeded within a certain time limit. This avoids updates when the drone is moving very slowly over longer time-periods. To get the best quality of the prediction for any displacement $\vec{u}$, it is desirable to have the vectors (N_{a*}, N_{b*}, N_{c*}) ideally, but not necessarily, orthogonal to each other. This also influences which row (vector) gets replaced in the matrices N and P. As soon as one of the vectors gets outdated, a random movement in an orthogonal direction might be triggered to enhance the knowledge representation.

3.2 Distributed flocking controller

We now describe our control approach based on the cost function introduced in Section 2.2 and on the environmental knowledge representation described in Section 3.1.
The set of candidate positions Q is defined as follows:

$$Q = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mid x \in \{-\epsilon_Q, 0, \epsilon_Q\}, y \in \{-\epsilon_Q, 0, \epsilon_Q\}, z \in \{-\epsilon_Q, 0, \epsilon_Q\} \right\} \tag{16}$$

This gives a set of 27 points on a equally spaced three dimensional grid. The spacing distance of this grid is ϵ_Q. Over this set Q the best action q_{next} is searched by minimizing the cost function c:

$$q_{next} = \underset{q \in Q}{\operatorname{argmin}} \{ c(\widehat{d}_i(q), \widehat{l}_i(q)) \} \tag{17}$$

If two candidate positions q_1 and q_2 both have the same minimum value for the cost function c, our implementation of argmin takes the last one based on the implementation of the enumeration. The function $\widehat{d}_i(q)$ estimates the distances to drones, where $\widehat{l}_i(q)$ estimates the distance to the target, if the action q is applied. For each $q \in Q$, the vector $\widehat{d}_i(q)$ (and the value $\widehat{l}_i(q)$) is calculated by first computing the estimates of the *change vector* $\bar{d}_i(q)$, and the change

$\bar{l}_i(q)$ using Equation 15. Now the distances can be estimated by just adding the estimated change to the currently measured distances d_{i*} and l_i:

$$\widehat{d}_i(q) = d_{i*} + \bar{d}_i(q) \tag{18}$$

$$\widehat{l}_i(q) = l_i + \bar{l}_i(q) \tag{19}$$

Each drone minimizes its local cost function (Eq. 17) in order to recompute the desired set-point at every time step. As we similarly did in [4], this set-point is then handed off to a low-level controller that controls the thrust of the drone's motors so that it steers towards this set-point.

4 Experiments

We evaluated our method using simulation experiments. The goal of the experiments was to investigate and demonstrate the ability to form and maintain a stable flock while holding position at a target location.

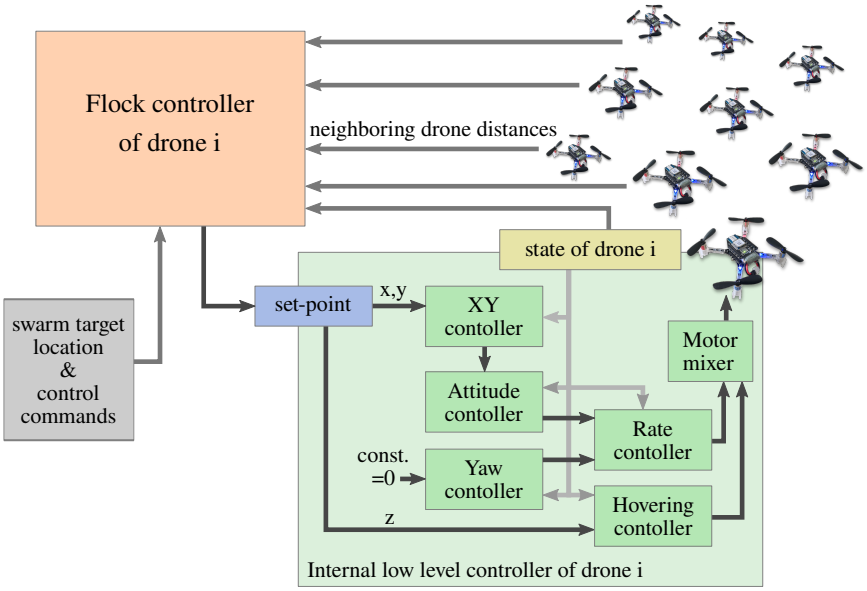

Fig. 3. The ROS-node of the SPC controller for drone i receives distance measurements to neighboring drones and control messages (e.g. swarm target location, start/stop command). It outputs the set-point for the internal low level controller.

4.1 Simulation Experiments

As a simulation framework, we use *crazys* [21], which is based on the *Gazebo* [12] physics and visualization engine and the Robotic Operating System (ROS) [24]. Our algorithm is implemented in C++ as a separate ROS node. As shown in Figure 3, it receives the measured distances to neighboring drones, and control messages, such as the target location or a stop command, from the human operator. It calculates the best next action according to Equations (16)-(19). The parameter ϵ_Q is determined empirically and fixed throughout the whole simulation. Auxiliary functions, like hovering at the starting position, and landing after the experiments are finished, are also implemented in this node.

In order to evaluate the control mechanism and its implementation, we fixed the target location, as described above. This avoids drifting behaviour of the whole flock, which could not be detected by relative distance measurements in any way. Simulations were done with flocks of size $|D| = 5$, 9, and 15. Figure 4 shows a screenshot of a simulation with 5 drones. All simulations use global neighborhood ($H_i = D$) for now.

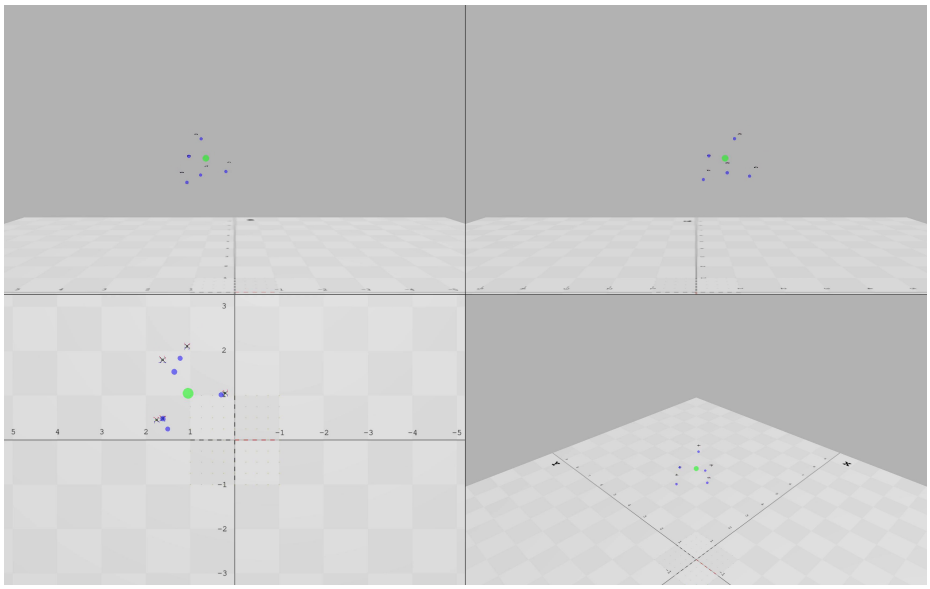

Fig. 4. Screenshot of the end of the simulation with 5 drones. Shown from four different camera views after the flock reached its target. The green dot indicates the target location. The blue dots visualize the next action which is supplied to the lower level controller.

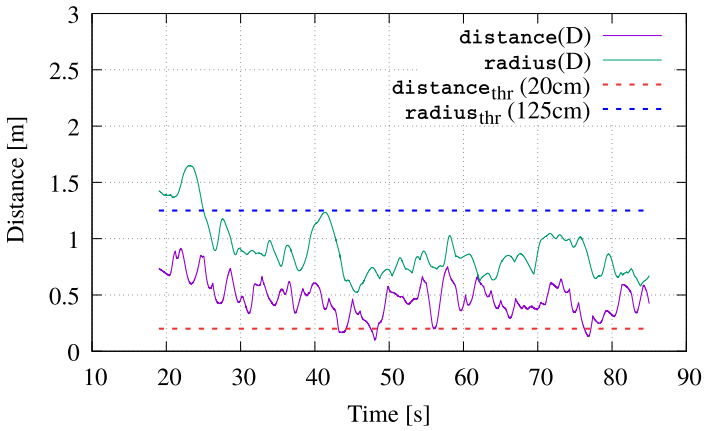

Fig. 5. Quality metrics for simulation with 5 drones. Threshold $\texttt{distance}_{thr}$ for collision avoidance is satisfied most of the time. After settling in, the swarm radius remains below the threshold $\texttt{radius}_{thr}$, thus showing the ability to form a compact flock in the simulation. (Quality metric recordings start at $t = 19\,s$ after initialization procedure.)

4.2 Results

Early results show that our approach is able to properly form and maintain a flock with only relative position measurements. Figure 5 shows performance metrics over time for a simulation with 5 drones. The analysis of the quality metrics for *collision avoidance*, and *compactness* show that our control approach successfully maintains a stable flock (threshold $\texttt{distance}_{thr}$ is only violated for very short moments). Note that these results are already obtained before extensive controller tuning. Using carefully adjusted values for ω_{coh} and ω_{sep} should lead to even better results and maintain the threshold throughout the whole simulation.

5 RELATED WORK

Reynolds [19] was the first to propose a flocking model, using cohesion, separation, and velocity alignment force terms to compute agent accelerations. Reynolds' model was extensively studied [10] and adapted for different application areas [6]. Alternative flocking models are considered in [18], [15], [14], [22], [25], and [20]. In all these approaches, absolute position measurements and/or inter-agent communication were available. In our work, we only work with relative distance measurements and a fully distributed formulation.

In addition to these largely theoretical approaches, in [26] and [23], flocking controllers are implemented and tested on real hardware. However, the approach of [23] involves the use of nonlinear model-predictive control (NMPC). In contrast to our work, [26] also requires the velocity of neighboring drones.

6 CONCLUSIONS

We introduced a method to control a flock only based on relative position measurements to neighboring drones, and demonstrated its utility on the drone flocking problem. We performed simulation experiments using a physics engine with a detailed drone model. Our results demonstrated the ability to form and maintain a flock, and hold its position on a target location.

Future work

As we currently have only intermediate results of the experiments with limited number of agents, we plan to do more extensive testing with a wide set of different scenarios, including larger number of drones, and local neighborhood ($H_i \subset D$). Neighborhood might be defined by euclidean distance, or alternatively by topological distance, as introduced in [2]. As further directions of future work, we plan to extend our approach with obstacle avoidance capabilities. We also plan to test it for moving target locations and various path tracking scenarios. To prepare for the transfer to real hardware we plan to introduce sensor noise in the simulation and test the robustness of our method to cope with such disturbances. As next goal it should then be implemented on real drones, specifically, *Crazyflie 2.1*-quadcopters [11].

ACKNOWLEDGMENTS

R.G. was partially supported by EU-H2020 Adaptness and AT-BMBWF DK-RES and CPS/IoT Ecosystem. This work was supported in part by NSF grants CCF-1954837, CCF-1918225, and CPS-1446832.

References

1. Attanasi, A., Cavagna, A., Castello, L.D., Giardina, I., Jelic, A., Melillo, S., Parisi, L., Pohl, O., Shen, E., Viale, M.: Emergence of collective changes in travel direction of starling flocks from individual birds' fluctuations. Journal of the Royal Society **12** (Jul 2015)
2. Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Lecomte, V., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., Zdravkovic, V.: Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. Proceedings of the National Academy of Sciences **105**(4), 1232–1237 (2008). https://doi.org/10.1073/pnas.0711437105, https://www.pnas.org/doi/abs/10.1073/pnas.0711437105
3. Boggio-Dandry, A., Soyata, T.: Perpetual flight for UAV drone swarms using continuous energy replenishment. In: 2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON). pp. 478–484 (2018). https://doi.org/10.1109/UEMCON.2018.8796684

4. Brandstätter, A., Smolka, S.A., Stoller, S.D., Tiwari, A., Grosu, R.: Multi-agent spatial predictive control with application to drone flocking (extended version) (2022). https://doi.org/10.48550/ARXIV.2203.16960, https://arxiv.org/abs/2203.16960
5. Chung, S.J., Paranjape, A.A., Dames, P., Shen, S., Kumar, V.: A survey on aerial swarm robotics. IEEE Transactions on Robotics **34**(4), 837–855 (2018). https://doi.org/10.1109/TRO.2018.2857475
6. Chung, S.J., Paranjape, A.A., Dames, P., Shen, S., Kumar, V.: A survey on aerial swarm robotics. IEEE Transactions on Robotics **34**(4), 837–855 (2018). https://doi.org/10.1109/TRO.2018.2857475
7. Cucker, F., Smale, S.: Emergent behavior in flocks. IEEE Transactions on Automatic Control **52**(5), 852–862 (2007). https://doi.org/10.1109/TAC.2007.895842
8. Cucker, F., Smale, S.: On the mathematics of emergence. Japanese Journal of Mathematics **2**(1), 197–227 (Mar 2007). https://doi.org/10.1007/s11537-007-0647-x, https://doi.org/10.1007/s11537-007-0647-x
9. Câmara, D.: Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios. In: 2014 IEEE Conference on Antenna Measurements Applications (CAMA). pp. 1–4 (2014). https://doi.org/10.1109/CAMA.2014.7003421
10. Eversham, J., Ruiz, V.F.: Parameter analysis of Reynolds flocking model. In: 2010 IEEE 9th International Conference on Cyberntic Intelligent Systems. pp. 1–7 (2010). https://doi.org/10.1109/UKRICIS.2010.5898089
11. Giernacki, W., Skwierczyński, M., Witwicki, W., Wroński, P., Kozierski, P.: Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In: 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR). pp. 37–42 (2017). https://doi.org/10.1109/MMAR.2017.8046794
12. Koenig, N., Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). vol. 3, pp. 2149–2154 vol.3 (2004). https://doi.org/10.1109/IROS.2004.1389727
13. Loianno, G., Kumar, V.: Cooperative transportation using small quadrotors using monocular vision and inertial sensing. IEEE Robotics and Automation Letters **3**(2), 680–687 (2018). https://doi.org/10.1109/LRA.2017.2778018
14. Martin, S., Girard, A., Fazeli, A., Jadbabaie, A.: Multiagent flocking under general communication rule. IEEE Transactions on Control of Network Systems **1**(2), 155–166 (2014). https://doi.org/10.1109/TCNS.2014.2316994
15. Mehmood, U., Paoletti, N., Phan, D., Grosu, R., Lin, S., Stoller, S.D., Tiwari, A., Yang, J., Smolka, S.A.: Declarative vs rule-based control for flocking dynamics. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. p. 816–823. SAC '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3167132.3167222
16. Michael, N., Fink, J., Kumar, V.: Cooperative manipulation and transportation with aerial robots. Autonomous Robots **30**(1), 73–86 (2011). https://doi.org/10.1007/s10514-010-9205-0
17. Michael, N., Shen, S., Mohta, K., Kumar, V., Nagatani, K., Okada, Y., Kiribayashi, S., Otake, K., Yoshida, K., Ohno, K., Takeuchi, E., Tadokoro, S.: Collaborative mapping of an earthquake damaged building via ground and aerial robots. In: Proceedings of 8th International Conference on Field and Service Robotics (FSR '12). pp. 33 – 47 (7 2012)

18. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Transactions on Automatic Control **51**(3), 401–420 (2006). https://doi.org/10.1109/TAC.2005.864190
19. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. p. 25–34. SIGGRAPH '87, Association for Computing Machinery, New York, NY, USA (1987). https://doi.org/10.1145/37401.37406
20. Schwager, M.: A Gradient Optimization Approach to Adaptive Multi-Robot Control. Ph.D. thesis, Massachusetts Institute of Technology (2009), https://dspace.mit.edu/handle/1721.1/55256
21. Silano, G., Aucone, E., Iannelli, L.: CrazyS: A software-in-the-loop platform for the Crazyflie 2.0 nano-quadcopter. In: 2018 26th Mediterranean Conference on Control and Automation (MED). pp. 1–6 (2018). https://doi.org/10.1109/MED.2018.8442759
22. Soria, E., Schiano, F., Floreano, D.: The influence of limited visual sensing on the Reynolds flocking algorithm. In: 2019 Third IEEE International Conference on Robotic Computing (IRC). pp. 138–145 (2019). https://doi.org/10.1109/IRC.2019.00028
23. Soria, E., Schiano, F., Floreano, D.: Predictive control of aerial swarms in cluttered environments. In: Nature Machine Intelligence (2021). https://doi.org/10.1038/s42256-021-00341-y
24. Stanford Artificial Intelligence Laboratory et al.: Robotic operating system (2018), https://www.ros.org
25. Tanner, H.G., Jadbabaie, A., Pappas, G.J.: Stability of flocking motion. Tech. rep., University of Pennsylvania (2003), https://www.georgejpappas.org/papers/boids03.pdf
26. Vásárhelyi, G., Virágh, C., Somorjai, G., Nepusz, T., Eiben, A.E., Vicsek, T.: Optimized flocking of autonomous drones in confined environments. Science Robotics **3**(20) (2018). https://doi.org/10.1126/scirobotics.aat3536