Self-organized Chain Formation of Nano-Drones in an Open Space

Barciś, Agata; Barciś, Michał; Natalizio, Enrico; Ferrante, Eliseo

published in
Swarm Intelligence
2022

DOI (link to publisher)
10.1007/978-3-031-20176-9_18

document version
Publisher's PDF, also known as Version of record

Self-organized Chain Formation of Nano-Drones in an Open Space

Agata Barciś[1(✉)], Michał Barciś[1], Enrico Natalizio[1,2], and Eliseo Ferrante[1,3]

[1] Technology Innovation Institute, Autonomous Robotics Research Center, Abu Dhabi, United Arab Emirates
{agata.barcis,michal.barcis,enrico.natalizio,eliseo.ferrante}@tii.ae
[2] Université de Lorraine, CNRS, Loria, Villers-lès-Nancy, France
[3] Vrije Universiteit Amsterdam, Amsterdam, Noord-Holland, The Netherlands

Abstract. We propose a method for the chain formation of multiple agents in an open space. Chaining can be considered as a building block for several application scenarios, including exploration, maintaining connectivity, or path formation. The proposed method was designed for a very sensing and computationally constrained robot platform, more specifically for nano-drones as they offer advantages in applications in tight spaces or in the proximity of people. To enable portability to a real platform, the method relies on a range and bearing sensing model with a limited field of view that is susceptible to occlusions, which was implemented both in simulation as well as on the real robot through a camera coupled with LEDs. We analyze the method in the simulation-based study. We show that the method works even in presence of noise in sensing and actuation, which rather than being harmful to the chaining performance has a positive effect. We analyze the performance in terms of quality of final chain formation, and speed of convergence, and how these two are affected by increasing swarm size. Finally, we present its practical feasibility in a robotic proof-of-concept featuring nano-drones.

1 Introduction

The problem of chain formation is widely studied in swarm robotics, and fosters many practical application scenarios. In fact, it enables path formation in which the agents act as landmarks for other agents in foraging [14,18] or task sequencing [6]. Chaining was also applied to navigation [5] and exploration of tunnel-like environments (e.g., caves) with the requirement to keep connectivity [8,10]. Despite the advancements in the development of both chaining methods and robot capabilities, most state-of-the-art solutions are designed for ground robots, and there are still no reliable solutions available for drones.

In this work, we focus on chaining methods that would be suitable for nano-drones. The main motivation for this choice is the capability of nano-drones to explore tight spaces, e.g., underground tunnels, ventilation ducts, or pipes. Additionally, thanks to their low cost, they facilitate the development and deployment

M. Dorigo et al. (Eds.): ANTS 2022, LNCS 13491, pp. 222–233, 2022.
https://doi.org/10.1007/978-3-031-20176-9_18

of large groups of robots. Finally, they are safer than standard-size drones and can work even in proximity of people, for example, in warehouse inventory or factory surveillance, without disturbing their normal operations.

However, these advantages of nano-drones come at the price of numerous limitations and challenges. The noise in sensing and actuation, present in any robotic or especially drone system, might be increased due to the minimization of sensors. In general, nano-drones used for swarm robotics have much more limited sensing abilities compared to the ground counter-part, and standard swarm models developed for ground robots are not directly portable to nano-drones [2,3]. The small payload that the nano-drones can carry leads to two consequences: (i) the size of the battery carried by the drone is highly limited and so is its flight time, and (ii) the choice of sensors is quite narrow compared to the possibilities of larger platforms. In particular, equipping a nano-drone with an omni-directional sensor (for neighbor sensing) as the one normally considered in ground robots is very difficult. Most of the state-of-the-art approaches to chaining are infeasible for nano-drones because they have too complex sensing demands, e.g., they require omni-directional sensing [5,6,12–14], global positioning [17], they take too much time to converge [18], or rely on the assumption that the actuation noise is negligible [7].

In this paper, we present a method for open-space chaining that allows formation of the chain that does not require global positioning and that works even with limited sensing field of view (Sect. 2). We focus on realistic sensing conditions, so the method is robust against sensing and actuating noise. Additionally, the sensor used in this work is susceptible to occlusions that are often neglected by other swarming algorithms [16].

To analyze the proposed method we perform simulation-based experiments (Sect. 3). We study the convergence time of the proposed method, the quality of the chaining obtained, under the effect of different swarm sizes and of realistic sensing and actuation noise. Finally, we present the practical feasibility of chaining in a robotic proof of concept with nano-drones, in which sensing is done fully on-board (Sect. 4).

2 Method

The system consists of N agents. The position of agent i is denoted as $\mathbf{x}_i$ and its orientation, i.e. yaw angle, is θ_i. We denote the displacement vector from agent i to j as $\mathbf{x}_{i,j} = \mathbf{x}_j - \mathbf{x}_i$ and distance between them as $\|\mathbf{x}_{i,j}\|$. Agent i is controlled by setting its velocity ($\mathbf{v}_i$) and yaw rate ($\dot{\theta}_i$) in its own reference frame.

All of the agents are equipped with a field-of-view (FOV) sensor. We assume such a sensor is able to measure the range (r) and bearing (ϕ) to all agents in front of it, within limited viewing angle and range. In practice, such a sensor could be realized in multiple ways: using a camera, radar, LiDAR, sonar, etc. Many of these technologies are susceptible to *occlusions*: if multiple objects share exactly the same bearing, but are at different distances, they will be perceived as a single object. Our real-life realization of the FOV sensor is based on a camera

(see Sect. 4.1 for details). It perceives multiple occluded objects as being closer than the closest of these objects. Our simulated setup models this behavior.

Note that the agents do not have access to the global positions of their peers or themselves. Therefore, whenever we mention the positions of other agents while calculating the update of agent i, we mean the relative positions with respect to the reference frame of agent i. Such positions are calculated based on the output of the FOV sensor $\mathbf{x}_j = \left[\cos\phi_j \cdot r_j, \sin\phi_j \cdot r_j\right]$.

The chain is parameterized with a desired distance d_0 that specifies the distance the agents should keep from their neighbors. The agents are deployed sequentially—as soon as agent i is positioned in the chain, agent $i+1$ takes off. Sequential deployment is implemented manually in this paper, by having the current agent notifying the simulator or the operator when it has completed joining the chain. This assumption can be relaxed in future work by either implementing a self-organized communication scheme that monitors the status of chain formation, or by removing the sequential deployment assumption altogether.

We distinguish two agents with special roles:

- One agent is used to emulate the *target* t. Its only task is to hover in the same spot throughout the whole chain forming process. It can be considered external to the swarm.
- Another special agent is the *seed* s. It follows a different logic than all the other agents. Its goal is to find the target and keep a constant distance to it (1), at the same time keeping it in the middle of its field of view (2):

$$\mathbf{v}_s = P_{\mathbf{x}} \left(\mathbf{x}_t - \frac{\mathbf{x}_{t,s}}{\|\mathbf{x}_{t,s}\|} \cdot d_0 \right), \tag{1}$$

$$\dot{\theta}_s = P_\theta \phi_t, \tag{2}$$

  where $P_{\mathbf{x}}$ and P_θ are the proportional controller settings for position and yaw, respectively. Since the agents have no sensor allowing them to distinguish the target, the seed agent assumes that the target is the closest agent out of the detected ones. This has no implications in the early phase of chain formation (when the seed is the deployed robot and only the target is hovering in the environment), while later in an experiment it can happen that the seed will seldom mistake another agent as the target.

In practical scenarios, the *target* agent could be replaced by some point of interest that the *seed* agent is able to detect (provided the target identification modules, out of scope for this paper, are implemented); or it could be used as a leader initiating the construction of a chain in a specific location determined by another criteria. The rest of agents are homogeneous and are guided by the procedure consisting of four main steps that are followed sequentially. The diagram explaining the behavior with possible transitions is depicted in Fig. 1. All controls described in the method are in the agent's relative frame, e.g., when the agent moves left, we mean that it moves to *its* left.

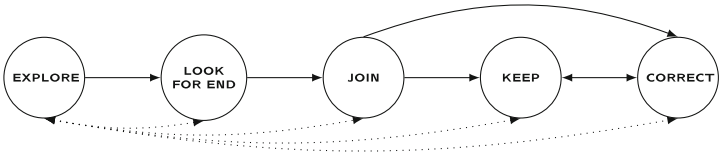

Fig. 1. Possible state transitions in the chaining algorithm.

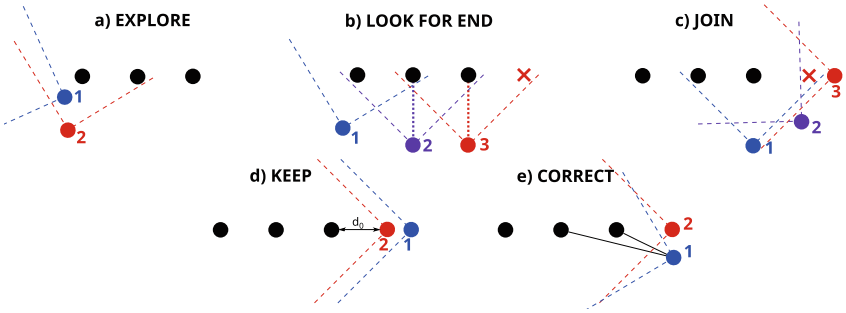

Fig. 2. An example of the chaining procedure. Black circles represent the part of the chain that is already formed. Dashed lines visualize the field of view of an agent, without showing the range limit. Blue circle represents an example of starting position of an agent in a given step and the red circle is the final position. In subfigures b) and c) an intermediate step is drawn with a purple color. (Color figure online)

Exploration (EXPLORE). At the beginning, just after being deployed, the agent i needs to find the chain. We assume, the chain is found if the agent detects at least two other agents in its field of view. In principle, any exploration strategy could be used for this step [11]. For the sake of simplicity, we assume the range of the FOV sensor is big enough to always detect a part of the chain from the place of the take-off, which is achieved by allowing the take off only within a specific area that has the target at its center. A possible take-off position and moment when the chain is detected are depicted in Fig. 2a as 1 and 2, respectively.

In the EXPLORE state, the agent just rotates around its center with yaw rate $\dot{\theta}_i = 0.2\,\mathrm{rad/s}$. When the agent detects another agent j in its field of view, it tries to keep it in the middle of its field of view, setting its yaw rate to $\dot{\theta}_i = P_\theta \cdot \phi_j$. Additionally, the agent moves left to go around the detected agent. It sets its velocity to $\mathbf{v}_i = [0, -0.2]$ m/s. If the detected agent is closer than d_c, the agent additionally moves back, so its velocity is $\mathbf{v}_i = [-0.1, -0.2]$ m/s. When the agent detects at least two agents in its field of view, it switches to the next step.

The agent can switch to the EXPLORE state from any of the other states if the required number of neighbors is not visible, and switches back to the state it comes from once enough neighbors are perceived. Effectively, this is used as a recovery strategy: The agent explores until it can see enough agents and continues with the execution of the previous state (transitions marked with dotted lines in Fig. 1).

Look for the End of the Chain (LOOK for END). We assume that agent k is the rightmost one in the FOV, and agent j is the second from the right in the FOV. The following equations describe movement of agent i:

$$\mathbf{v}_i = P_{\mathbf{x}} \mu_\beta \left[\min(r_j, r_k) - d_c, y_k \right] \tag{3}$$

$$\dot{\theta}_i = P_\theta \psi_\beta \cdot \arctan\left(\frac{y_{j,k}}{x_{j,k}} - \frac{\pi}{2} \right), \tag{4}$$

where y_k is a y coordinate of $\mathbf{x}_k$ and $x_{j,k}$ and $y_{j,k}$ are, respectively, x and y coordinates of $\mathbf{x}_{j,k}$ in the relative frame of reference, and β is the angle between agents i, j, k. The coefficients $\mu_\beta = \max(0.2, |\sin(\beta)|)$ and $\psi_\beta = \max(0.1, |\sin(\beta)|)$ are used to slow down the movement and rotation, respectively, if the agents i, j and k are close to being aligned.

Intuitively, the movement of the agent in this step can be split into three parts: it rotates to be perpendicular to the chain (4); it moves in the x direction to keep the constant distance d_c from the chain (3); and in the y direction to be directly in front of the agent k, so to reach its y coordinate (3). There is one corner case that we consider separately: if the agents j and k are almost aligned (difference of their bearings is smaller than 0.1 and hence it is likely there are occlusions), agent i moves with a constant speed to the left.

The distance d_c is chosen based on the FOV sensor viewing angle in such a way that the agent can see at least 3 agents that are in the chain (assuming they keep the desired distance d_0 from each other) with some safety margin for noise and disturbances caused by occlusions. The agent is moving to be directly in front of the rightmost visible agent (e.g., agent 2 in Fig. 2b). When there are no more agents on the right, it switches to the next step to join the chain. This final position is marked as 3 in Fig. 2b and the spot for the next agent in the chain is marked with a red cross.

Join the Chain (JOIN). To join the chain, the agent i is choosing the first two agents starting from the right of the FOV (the LOOK FOR END state guarantees that these are two agents at the end of the chain). We call these agents j and k, with k being the rightmost one (last in the chain). The agent rotates towards the point S in the middle between agents j and k

$$S = \frac{\mathbf{x}_j + \mathbf{x}_k}{2}, \tag{5}$$

$$\dot{\theta}_i = P_\theta \phi_S. \tag{6}$$

It calculates its desired position (marked with red cross in Fig. 2c) at the end of the chain:

$$\mathbf{x}_d = \mathbf{x}_k + d_0 \frac{\mathbf{x}_{j,k}}{\|\mathbf{x}_{j,k}\|} \tag{7}$$

and moves to the right with $\mathbf{v}_i = P_{\mathbf{x}} \left[0.25 \|\mathbf{x}_d\|, 0.5 \|\mathbf{x}_d\| \right]$. The movement in the x direction allows the agent to correct the distance from the chain on the way. The rotation combined with the motion result in the movement on an arc that

brings the agent i to the position aligned with agents j and k. In this state, we say i is *aligned* with agents j and k if:

- bearing to j and k is equal when occlusions are not taken into account,
- the agents occlude each other and only one agent is detected.

Once the two closest agents are aligned, the agent is already considered to be a part of the chain and needs to remain in it, the procedure for this is described in the next step.

Keep the Chain (KEEP). When all the visible agents are aligned, the agent keeps the desired distance d_0 with the closest neighbor (see Fig. 2d). The behavior is the same as the one of seed (see equations (1) and (2)). However, if there are some disturbances (caused by noise or new agents trying to join the chain) and the agent detects some unaligned agents it switches to the next step.

Correct the Chain (CORRECT). Let us assume that agents j and k are the closest neighbors (the agents connected with black lines with agent 1 in Fig. 2e). The agent rotates towards the middle between two closest agents:

$$\dot{\theta}_i = P_\theta \cdot \frac{\phi_j + \phi_k}{2}. \tag{8}$$

At the same time, it moves towards the corrected position in a straight line:

$$\mathbf{v}_i = P_{\mathbf{x}} \mathbf{x}_d, \tag{9}$$

where $\mathbf{x}_d$ is defined as in Eq. (7). Once all the visible agents are aligned again, the agent switches back to the previous step (**KEEP**), this situation corresponds to the position marked with 2 in Fig. 2e.

3 Simulation Results

For the simulation study we use a custom kinematic-based simulator implemented in Python, designed to bridge the simulation to reality gap. In fact, exactly the same logic, programmed in Python, is executed both in simulation and during robotic experiments. In our simulations, agents take off at random positions drawn uniformly from a deployment area 10 m × 10 m. The *target* agent is placed exactly in the middle of the deployment area. To avoid disturbing the chain in the early stage of forming, a square 2 m × 2 m in the middle of the arena is excluded. If an initial position is chosen in this part of the area, the initial position is redrawn. Each agent is equipped in a FOV sensor with a viewing angle 90° and range 6 m. Such configuration of the sensor is motivated by our hardware setup. Furthermore, we assume the agents are not detected by the FOV sensor before takeoff. For each experiment we use five different configurations:

Idealized configuration is without any noise and with a sensor without occlusions that can "see through" the agents. It measures exact range and bearing to all the agents in range.
Occlusions configuration has no noise, but occlusions are taken into account.
Sensing noise adds sensing noise to the *Occlusions* configuration. The noise has two components: range noise $\eta_r \sim N(0, 0.1)$ and bearing noise $\eta_\phi \sim N(0, 0.05)$, which are added to the FOV sensor measurements.
Actuating noise adds actuating noise to the *Occlusions* configuration. Actuating noise is added to the agent's velocity $\eta_{\mathbf{v}} \sim \left[N(0, 0.05), N(0, 0.05)\right]$ and its yaw rate $\eta_{\dot{\theta}} \sim N(0, 0.1)$.
Both noises adds sensing and actuating noise to the *occlusions* configuration.

3.1 Quality of the Chain Formation—Collinearity of Agents

We say that the chain formation is perfect if, at the end of an experiment, the agents are placed on a straight line. To measure how successful the agents are in forming the chain, we use the following order parameter.

For each triple of agents (including the ones that are still on the ground) we calculate the collinearity parameter.

$$C_{i,j,k} = \frac{\pi - \max(\alpha_{i,j,k},\ \alpha_{k,i,j},\ \alpha_{j,k,i})}{\pi}, \tag{10}$$

where $\alpha_{i,j,k}$ is the angle between the agents, assuming that j is the vertex of the angle. It is calculated as:

$$\alpha_{i,j,k} = \arccos \frac{\mathbf{x}_{j,i} \cdot \mathbf{x}_{j,k}}{\|\mathbf{x}_{j,i}\| \, \|\mathbf{x}_{j,k}\|}. \tag{11}$$

In the case when the agents are collinear, one of these angles is equal to π. Therefore, choosing the maximum of these angles allows us to determine how far are the agents from creating a straight line.

The collinearity of all the agents can be calculated as:

$$C = \frac{1}{\binom{N}{3}} \sum_{(i,j,k) \in \mathscr{C}(\{1,\ldots,N\})} C_{i,j,k}, \tag{12}$$

where $\mathscr{C}(\{1, \ldots, N\})$ is the set of all 3-combinations of agents.

In this experiment, we demonstrate how the metric C converges when the agents are creating the chain. We execute 50 runs for each configuration. The number of agents is constant and $N = 10$. The results are presented in Fig. 3. The line marks the median of the metric C for each simulation step from all 50 runs, whereas the colored area shows the range between first and third quartiles.

In all of the configurations the convergence curve of C has a similar shape and reaches a stable low value. This means that the agents manage to form the chain in a repeatable way. What is worth noting is that only in the setup with the idealized FOV sensor without occlusions the metric converges to 0, in the other plots it converges to a slightly greater value (around 0.03). The reason for that is that with occlusions the agents are not able to distinguish between an exactly straight and a slightly bent chain.

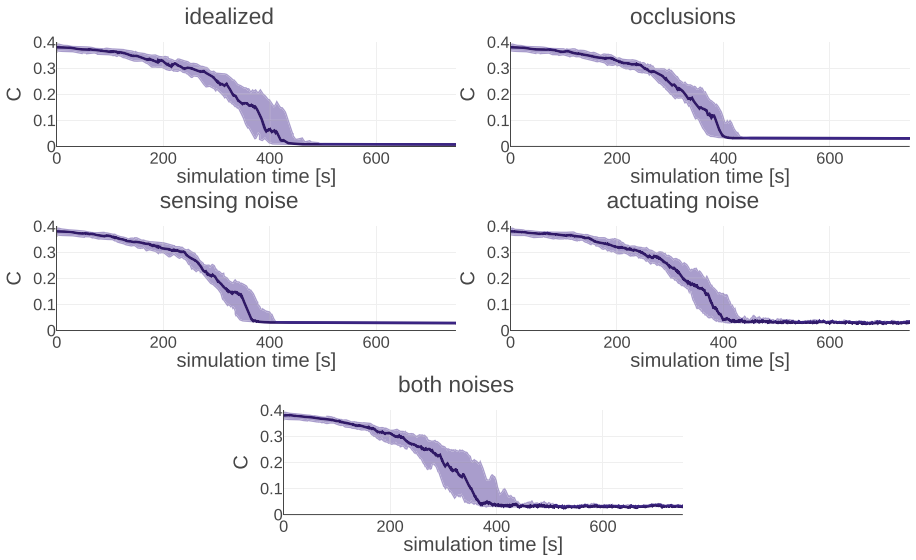

Fig. 3. The plot depicting convergence of order parameter C for five different experiment setups.

3.2 Convergence Time

In this experiment, we measured the time it takes for all the agents to detect they are in a chain (they are either in a KEEP or CORRECT state), and we use this also as a criteria to terminate the experiment. To avoid deadlocks, we assume that the maximum duration of an experiment is $2000 \cdot N$ simulation steps, after this time the experiment is stopped. By combining these two stopping criteria, we can easily distinguish the cases in which the chain converged to a low value of collinearity from the cases in which this did not happen (the experiment was stopped after $2000 \cdot N$, which corresponds to the dashed line in Fig. 4).

The experiment is run for different number of agents from $N = 10$ to $N = 100$, with the increment of 10 agents. For each number of agents we execute 25 runs for each of the five configurations of sensing capabilities and noise.

Figure 4 presents the results obtained in the course of the experiments. Each color represents a different configuration. The semi-transparent bars display the first and third quartile, with a median marked with a line. The line bars mark maximum and minimum or upper and lower fence. Additional points, not covered by the bars, indicate the outliers.

The worst performance is achieved in the setup without noise. With the sensor without occlusions, above 70 agents the chain is never achieved. Additional analysis of the failed runs of simulations showed that the agents fail to form a chain if one of them takes off between the agents that are already in the chain or very close to them. In this case, the agents try to keep the chain also with the new joiner, which results in the part of the chain following the agent that

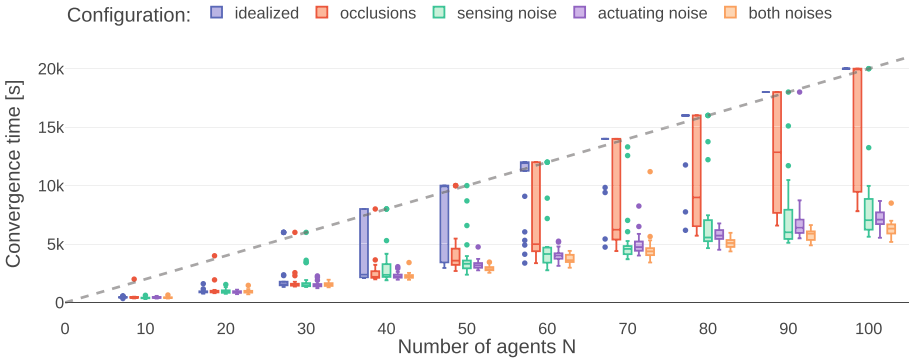

Fig. 4. The plot depicting the relationship between the convergence time and number of agents for five different experiment setups.

tries to join the chain. In such scenarios, the occlusions have a stabilizing effect: when the agent goes out of the line, the chain seems to be closer and the agents tend to stay with the chain rather than chase the new agent.

The setup with occlusions and no noise performs a bit better but, still, it is not perfect and in some runs the agents do not achieve the chain in the given time. The lack of convergence in this case is caused by deadlocks. A deadlock occurs when an agent takes off almost aligned with the chain. Depending on the direction it starts movement, the occlusions cause the other subset of agents to seem closer, which in turn causes the agent to move in the opposite direction. This procedure is continued until the end of the given simulation time.

Despite these shortcomings, in the setup with any kind of noise the agents almost always achieve chain, with only a few outliers. When the sensing and actuating noises are combined, the chain converges always and in the most efficient way. These results confirm that our solution can work in the realistic conditions and demonstrates its potential to be applied on nano-drones.

4 Proof of Concept

To show the practical applicability of the proposed method and that the simulated scenarios are realistic, we implemented a proof of concept on robots. We realized it using Crazyflie drones, a palm-sized, open-source flying robots.

To realize the proposed method in practice, we require a hardware implementation of the FOV sensor. We developed this sensor using a connected-component labeling algorithm implemented on AI-deck. AI-deck is an extension-board for Crazyflies that adds to it a HIMAX camera, GAP8 processor, and a WiFi module. For this work, we used the camera to capture pictures and the GAP8 processor to analyze them and estimate the range and bearing to all other agents in range. Then, this information is communicated to the main Crazyflie microcontroller as a sensor data. This sensor data is used in the control algorithm to coordinate the drone.

To facilitate the development and testing, the sensor data is communicated from the Crazyflie via radio to a laptop computer, where the control algorithm is running and sending the movement commands back to the Crazyflie. Such setup adds considerable latency to the control process, but allows for easier development. Furthermore, it shows the robustness of the proposed method. Even though we use a laptop computer, the logic is using only the information that would have been available on the drone. This, together with the fact that the method is not computationally intensive, allows to implement the whole solution fully on-board.

4.1 Camera-Based FOV Sensor

We develop a custom-made range and bearing sensor based on camera and LEDs. The drones are flying in the darkness with a 1W LED attached at the bottom to provide light for the optical flow sensor. The automatic exposure and gain (AEG) feature of the camera is turned off and the exposure and gain is tuned manually to achieve reproducible results for a fixed lighting conditions. In such setup, the only bright objects in a camera picture captured by a drone are the other drones.

To obtain range and bearing measurements to all visible drones from a camera picture, we use the following algorithm:

1. thresholding by discarding all pixels below a hand-tuned threshold,
2. expand the non-discarded areas by 4 pixels in all directions,
3. identify different drones by finding and labeling connected image components [15] using union-find data structure [4],
4. compute the centers and sizes of the connected components.

After that, we use the resulting centers of connected components to approximate the bearing ϕ to the other drone and the width of the component to approximate the distance d with the following equations:

$$d_i = \frac{1}{w_i} k, \tag{13}$$

$$\phi_i = \frac{x_i - \frac{1}{2}W}{W} A, \tag{14}$$

where w_i is the width of the i-th component and x_i is the center of this component in the horizontal direction. In our setup, $k = 15$ is a manually-tuned scaling factor dependent on the lighting conditions, $W = 255$ is the width of the camera image, and $A = 90°$ is the horizontal angle of view of the camera.

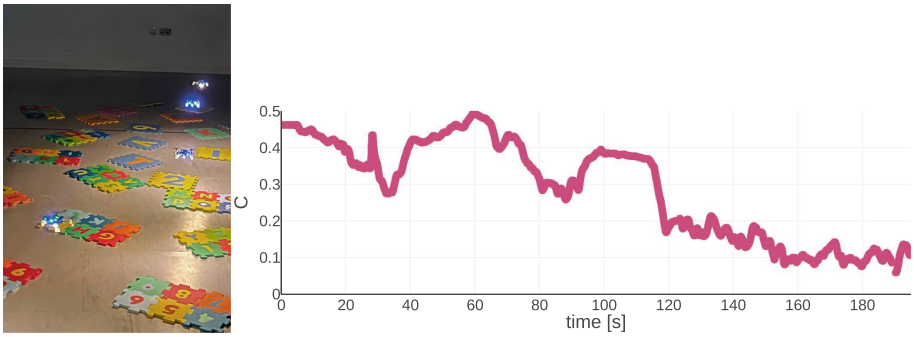

Fig. 5. (left) A photo of robots during an experiment. Three robots already formed a chain and the fourth one is joining it. (right) Convergence of the order parameter for a robotic experiment.

4.2 Results

We conducted experiments with four drones using the described setup and confirmed similar behavior as the one observed in the simulation. The main challenges were related to the fact that the FOV sensor requires that the only sources of light seen by the drones are the other agents. Additional sources of light and reflections from shiny surfaces negatively influence the sensor's measurements and might lead to a failed attempt. However, with some care to minimize these effects, we were able to successfully present the proof of concept.

We recorded the trajectories of robots using Qualisys motion capture system. Figure 5 shows the value of the order parameter C calculated based on the captured trajectories. It shows that the drones are able to construct and maintain the chain. The FOV sensor is able to provide around 1 measurement per second. The results show that even with such a low update rate the chaining algorithm performs well. We have published a video [1] presenting the proof of concept.

5 Conclusions and Outlook

We presented a method capable of building a chain of agents in an open space. It has a minimal sensing requirements with a limited field of view range and bearing sensor. The main advantage is its robustness against noise that enables real-life applications in robotic systems.

The future work on this topic will include exploration of the environment to connect target to a base, and the removal of the incremental deployment assumption. The real-life applications will benefit from improving the FOV sensor, it may include utilizing artificial intelligence methods [9].

References

1. Barciś, A., Barciś, M., Natalizio, E., Ferrante, E.: Video: Self-Organized Chain Formation of Nano-Drones in an Open Space (2022). https://youtu.be/Fqp-9Et3lmw

2. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intell. **7**(1), 1–41 (2013). https://doi.org/10.1007/s11721-012-0075-2
3. Coppola, M., McGuire, K.N., De Wagter, C., de Croon, G.C.H.E.: A survey on swarming with micro air vehicles: fundamental challenges and constraints. Front. Robot. AI **7**, 18 (2020). https://doi.org/10.3389/frobt.2020.00018
4. Cormen, T.H.: Data structures for disjoint sets. In: Introduction to Algorithms, 3rd edn, pp. 561–568. MIT Press, Cambridge (2009)
5. Ducatelle, F., Di Caro, G.A., Pinciroli, C., Mondada, F., Gambardella, L.: Communication assisted navigation in robotic swarms: self-organization and cooperation. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4981–4988 (2011). https://doi.org/10.1109/IROS.2011.6094454
6. Garattoni, L., Birattari, M.: Autonomous task sequencing in a robot swarm. Sci. Robot. **3**(20), eaat0430 (2018). https://doi.org/10.1126/scirobotics.aat0430
7. Jiang, Z., Wang, X., Yang, J.: Distributed line formation control in swarm robots. In: 2018 IEEE International Conference on Information and Automation (ICIA), pp. 636–641 (2018). https://doi.org/10.1109/ICInfA.2018.8812317
8. Laclau, P., Tempez, V., Ruffier, F., Natalizio, E., Mouret, J.B.: Signal-based self-organization of a chain of UAVs for subterranean exploration. Front. Robot. AI **8**, 614206 (2021)
9. Li, S., De Wagter, C., de Croon, G.C.H.E.: Self-supervised Monocular Multi-robot Relative Localization with Efficient Deep Neural Networks. arXiv:2105.12797 (2021)
10. Maxim, P.M., Spears, W.M., Spears, D.F.: Robotic chain formations. IFAC Proc. Vol. **42**(22), 19–24 (2009). https://doi.org/10.3182/20091006-3-US-4006.00004
11. Méndez, V., Campos, D., Bartumeus, F.: Random search strategies. In: Méndez, V., Campos, D., Bartumeus, F. (eds.) Stochastic Foundations in Movement Ecology. SSS, pp. 177–205. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-39010-4_6
12. Nouyan, S., Campo, A., Dorigo, M.: Path formation in a robot swarm. Swarm Intell. **2**(1), 1–23 (2008). https://doi.org/10.1007/s11721-007-0009-6
13. Nouyan, S., Dorigo, M.: Chain based path formation in swarms of robots. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 120–131. Springer, Heidelberg (2006). https://doi.org/10.1007/11839088_11
14. Nouyan, S., Gross, R., Bonani, M., Mondada, F., Dorigo, M.: Teamwork in self-organized robot colonies. IEEE Trans. Evol. Comput. **13**(4), 695–711 (2009). https://doi.org/10.1109/TEVC.2008.2011746
15. Samet, H., Tamminen, M.: Efficient component labeling of images of arbitrary dimension represented by linear bintrees. IEEE Trans. Pattern Anal. Mach. Intell. **10**(4), 579–586 (1988). https://doi.org/10.1109/34.3918
16. Schilling, F., Soria, E., Floreano, D.: On the scalability of vision-based drone swarms in the presence of occlusions. IEEE Access **10**, 28133–28146 (2022). https://doi.org/10.1109/ACCESS.2022.3158758
17. Sousselier, T., Dreo, J., Sevaux, M.: Line formation algorithm in a swarm of reactive robots constrained by underwater environment. Expert Syst. Appl. **42**(12), 5117–5127 (2015). https://doi.org/10.1016/j.eswa.2015.02.040
18. Sperati, V., Trianni, V., Nolfi, S.: Self-organised path formation in a swarm of robots. Swarm Intell. **5**(2), 97–119 (2011). https://doi.org/10.1007/s11721-011-0055-y