Flock-Formation Control of Multi-Agent Systems using Imperfect Relative Distance Measurements

Andreas Brandstätter[1], Scott A. Smolka[2], Scott D. Stoller[2], Ashish Tiwari[3], Radu Grosu[1]

***Abstract*— We present *distributed distance-based control* (DDC), a novel approach for controlling a multi-agent system, such that it achieves a desired formation, in a resource-constrained setting. Our controller is fully distributed and only requires local state-estimation and scalar measurements of inter-agent distances. It does not require an external localization system or inter-agent exchange of state information. Our approach uses spatial-predictive control (SPC), to optimize a cost function given strictly in terms of inter-agent distances and the distance to the target location. In DDC, each agent continuously learns and updates a very abstract model of the actual system, in the form of a dictionary of three independent key-value pairs $(\Delta\vec{s}, \Delta\boldsymbol{d})$, where $\Delta\boldsymbol{d}$ is the partial derivative of the distance measurements along a spatial direction $\Delta\vec{s}$. This is sufficient for an agent to choose the best next action. We validate our approach by using DDC to control a collection of Crazyflie drones to achieve formation flight and reach a target while maintaining flock formation.**

I. INTRODUCTION

Multi-agent systems (MASs) can collectively perform tasks which are beyond the abilities of individual agents. For search-and-rescue (SAR) applications, there is a wide variety of multi-agent approaches using ground, aerial, floating, and underwater vehicles [1]–[3]. MASs can also play a role in environmental monitoring, space exploration, agriculture, entertainment, and industrial maintenance [4]. All of these applications require a coordination and control method for formation establishment and maintenance.

MAS formations, such as flocking, can be described by a (distributed) cost function $c(\boldsymbol{x})$ over the state variables $\boldsymbol{x}$ of an agent, such that when each agent minimizes this cost, the system reaches the desired formation [5], [6]. We call cost functions that are defined in terms of scalar distances $\boldsymbol{d} \subset \boldsymbol{x}$ to other agents only, *distance-based cost functions*. For flocking, formulations using distance-based cost functions can be found in [7], [8]. However, it turns out that controllers, such as potential-field control (PFC) [7] and spatial-predictive control (SPC) [8], choose their actions based on the cost-function's spatial gradient. Consequently, even for distance-based cost functions, existing approaches require knowledge of relative-position vectors to derive the spatial gradient, and subsequently the control actions.

A ***key challenge****, therefore, is to determine if the controller design for distance-based cost functions can be generalized such that control actions are chosen based only on scalar distances without knowledge of relative position vectors?*

While global-navigation satellite systems (GNSS) and comparable types of indoor localization systems [9], [10], can

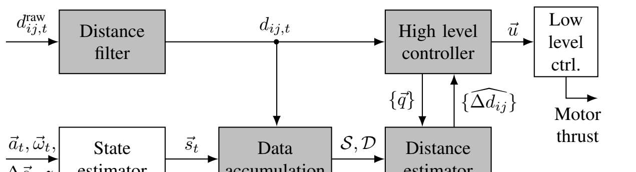

Fig. 1: Block diagram of our DDC distributed formation controller, running locally on each agent. Grey blocks were developed by us; for the white blocks, we use third-party state-of-the-art implementations.

determine relative position vectors, they might not be installed or currently available at some locations, and they are unlikely to be available in some applications (e.g., underwater SAR, cave exploration). This further motivates the above challenge.

One approach to solve the above challenge is to estimate the relative positions of other agents. This can be accomplished by measuring distances only, and using a coordinated movement schedule in each time step, as in [11]. One can then apply methods such as PFC or SPC to derive the actions. However, as shown by our experiments in Section V-B, this approach is impractical, since it is slow (only a subset of agents can move simultaneously), and susceptible to sensing noise.

In this work, we propose a novel approach, called DDC (for *distributed distance-based control*), that obviates the estimation of relative positions. Instead, it directly uses estimated distance changes in the spatial gradient of the cost-function. A key idea is to use a first order Taylor approximation of such changes. Intuitively, each agent maintains a dictionary of three independent key-value pairs $(\Delta\vec{s}, \Delta\boldsymbol{d})$, representing the partial derivative $\Delta\boldsymbol{d}$ of distance measurements along spatial directions $\Delta\vec{s}$.

During an initial startup-phase, agents perform *exploration* in orthogonal (or independent) directions, to populate this dictionary. Agents then *exploit* this dictionary to estimate the expected change $\widehat{\Delta\boldsymbol{d}}$ when they displace themselves by $\Delta\vec{u}$, as predicted by the Taylor approximation. This enables them to estimate the new cost when moving by a certain vector. As in [8], [12], to find the set-point with minimal cost, each agent performs in every time-step, a search over a grid of points surrounding the current location.

Importantly, each agent continuously *updates* its copy of the dictionary at each time step with its last observation. In particular, agents replace stale information with newer information, and they also occasionally perform an exploration step, where they take a potentially non-optimal action, to maintain independence of the keys in this dictionary. The dictionary is reminiscent of an attention layer used in transformers.

Note that each agent implicitly makes the simplifying, but not generally accurate, assumption that other agents do not move. Under the constrained setting, this is our only recourse.

[1] CPS, Technische Universität Wien (TU Wien), Austria
[2] Department of Computer Science, Stony Brook University, USA
[3] Microsoft, USA

Our experiments show that our approach is able to tolerate this inaccurate assumption. DDC passes the selected movement action to the agent's low-level controller that controls the motor thrust. Low-level controllers for quadcopters and their stability is studied in numerous works, such as [13]–[16].

MAS DDC is suitable in resource-constrained settings, where individual agents: (1) Are only able to perform local state-estimation with onboard sensors, (2) Do not communicate any state information, and (3) Can only measure scalar distances to other agents (and to the fixed target location). The local state-estimation of an agent is comprised of its change in position relative to a prior position, velocity, acceleration, orientation and optionally altitude. For this, onboard sensors, such as inertial-measurement units (IMUs), optical flow modules, barometers, and altimeters are used. These sensors do not provide any information about other agents.

As a consequence of (2), there is no central coordinator, and we thus present a fully distributed approach. As another consequence of (2), triangulation, multilateration, rigid graph based methods and joint (global) state-estimation are not applicable. Scalar distance measurements to other agents (and to the fixed target location) contain significantly less information than relative position, pose, or angle measurements would do.

In our hardware implementation, distance measurements also turned out to be extremely noisy, providing only imperfect measurements. In contrast to model-predictive control (MPC) approaches, DDC does not need a physical model of the agents.

Summarising, the main contributions of this paper are:

- We introduce the concept of DDC, a novel formation control method, which is solely based on onboard sensing, and on scalar distance measurements.
- DDC is model-free, fully distributed, and does not require internal-state information of other agents.
- We evaluate DDC on the drone-flocking problem with target seeking, in a drone simulation environment.
- We experimentally validate our approach, by achieving flocking with real off-the-shelf quadcopters, *Crazyflie 2.1* [13].

II. RELATED WORK

In [17]–[21], agents are able to localize themselves with cameras, LiDAR sensors and/or external systems. In our setting, these sensors are not available. Other works [22]–[28], including the survey [29], assume individual agents can sense relative positions of their neighboring agents with respect to their own local coordinate systems. This provides considerably more information than scalar distance measurements.

Position estimation and control for 2D scenarios are studied in [30], [31], but generalization to 3D is not trivial. In [32]–[36], message exchange (transmitting acceleration, angular velocity, or other data) is used between at least some of the agents. DDC does not use such communication.

The work in [37] makes use of *anchor nodes*, with fully known positions in a global reference frame. In [38], a centralized approach to localization is presented. In contrast, DDC is fully distributed. In [39], a single leader moving with constant velocity is tracked by a single follower. In [40], one leading and two following agents are described.

Formation control in [41] is based on distance measurements involving sinusoidal perturbations to the agent's actions. Perturbation frequencies are assumed to be pairwise distinct, which limits application to larger numbers of agents, if in practice the allowed range for frequencies is bounded by the agent's mechanical limitations. Our previous work in [12] sketched the idea of distance-based flocking using precise distance values, but it could not handle imperfect measurements due to significant sensor noise and it performed no hardware experiments.

In this paper, we therefore introduce DDC, which features a different method of data accumulation, additional signal filtering, the use of exploration steps to assure an orthogonal basis, and a new slack parameter for the separation term. We evaluated DDC for altitude-aided drone flocking on real hardware. The work in [11] proposes a control scheme with three alternating periods: identification, control, and resting (where the agent needs to remain stationary). In Section V-A we provide a comparison of DDC to this method, which turns out to be very susceptible to sensing noise and much slower.

III. FORMATION CONTROLLER

Consider a collection of $|A|$ agents. Each agent i, where i ranges from 1 to $|A|$, runs a controller, consisting of the following blocks, as shown in Figure 1:

- **Distance filter**: As measurements of the scalar inter-agent distances coming from real hardware are imperfect (noisy), the raw measurements $d^{\text{raw}}_{ij,t}$ of the distance between agents i and j at time t are filtered.
- **State estimator**: All sensor data that is available for the individual agent i (e.g. acceleration $\vec{a}_t$, rotational velocity $\vec{\omega}_t$) is processed by a standard Kalman filter (using the state-of-the-art implementation available for Crazyflies [13]) for local state estimation. This is used to estimate relative position $\vec{s}_t$ (for Agent i), i.e., relative to its position at $t = 0$. Additional optional sensors that can directly measure displacement $\Delta\vec{s}_t$ and/or absolute elevation z_t can be incorporated when available.
- **Data accumulation**: As the system operates, historical data of the change in inter-agent distances $\Delta d_{ij,t}$ for displacements $\Delta\vec{s}_t$ is stored in two matrices $\mathcal{D}, \mathcal{S}$.
- **Distance estimator**: The data stored in $\mathcal{S}$ and $\mathcal{D}$ is used to (linearly) estimate the change in distances $\widehat{\Delta d_{ij}}$ (for all j) for any given candidate displacement $\vec{q}$.
- **High-level controller**: The cost $c(\boldsymbol{d})$ is evaluated on a set of candidate displacements $\{\vec{q}\}$. The best is chosen as $\vec{u}$.
- **Low-level controller**: We use a PID controller (as implemented in [13], [42]) to set the motor thrust for $\vec{u}$.

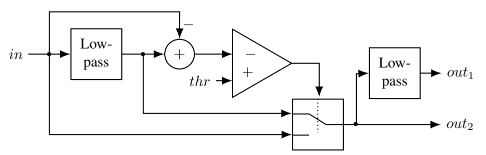

Fig. 2: Rejection filter: When the input is within threshold thr of the low-passed signal, the signal is fed through. Otherwise the filter outputs the low-passed signal (out_2). Finally there is an optional second low-pass (out_1).

A. *Outlier Detection and Noise Filtering*

Our hardware experiments use a collection of *Crazyflie 2.1* quadcopters [13], equipped with the *DWM1000* ultra-wideband (UWB) hardware module [43] on *Loco-positioning deck* with the software implementation of [44], to sense the inter-agent distances. The radio signals are used solely for inter-agent distance measurements. No internal state information is exchanged over this channel. There are no fixed UWB beacons.

As noise in distance measurements is a critical concern for our system, we did an analysis of the noise model. It turns out that *outliers* are common, and there is some additive *random noise*. When looking at its energy spectrum, we found higher energy in low-frequency ranges, as compared to white noise. Such a noise profile was also reported in [44].

Our filter tracks the input in, using an infinite impulse response (IIR) low-pass filter; see Figure 2. *To detect outliers*, we subtract the input in from the output of this low-pass and compare it with a threshold thr. If it is below the threshold, we directly use in as output; otherwise, we feed the low-passed signal to the output out_2. After this rejection filter, we use an optional *IIR low-pass filter* to produce out_1. Cutoff frequencies and threshold were determined empirically. In order to retain changes due to real movements, the cutoff frequency must not be too low. Therefore low-frequency components of noise cannot be filtered effectively. We observe, however, that higher-frequency noise is also effectively suppressed by *linear regression* in our *data accumulation* block (step 3a in Section III-B).

B. *Data Accumulation in Key-Value Dictionary*

Each agent i measures the current distances to neighboring agents (and the target) and estimates it's local position. For simplicity, we describe the distance estimation process from the perspective of agent i, only. The same procedure is analogously applied for estimating distance to the target.

Agent i stores the relevant data history in two matrices, a (3×3)-matrix $\mathcal{S}$ and a $(3\times|A|)$-matrix $\mathcal{D}$. Row-k of $\mathcal{S}$ is a *key* displacement vector $\Delta\vec{s}$ (in 3-dimensional space) for agent i. Row-k of $\mathcal{D}$ is a *value* vector $\Delta\boldsymbol{d}$ of change in distances to every other agent, as seen by agent i when it moved by displacement $\Delta\vec{s}$. In particular, $\mathcal{D}_{kj}$ is the change in distance to agent j as seen by agent i, when it moved by the vector $\mathcal{S}_{k*}$ ($\mathcal{S}_{k*}$ is $\mathcal{S}$'s k-th row vector). The dictionary thus represents partial derivatives $\mathcal{D}_{k*}$ of distance measurements along spatial directions $\mathcal{S}_{k*}$.

Let $d_{ij,t}$ be the distance to agent j at time t. Each agent's local state estimator is capable of estimating its own position $\vec{s}_t$ relative to its initial position at $t=0$. Agent i continuously updates the matrices $\mathcal{S}$ and $\mathcal{D}$ as follows:

0) Set t_0 and $\vec{s}_{t_0}$ to be the initial time and position of Agent i.
1) Initialize empty sets T and M_j for all $j \in A$.
2) Save the current time t and distances in sets: $T = T \cup \{t\}$, $M_j = M_j \cup \{d_{ij,t}\}$. Calculate the displacement vector $\Delta\vec{s}_t$ based on relative position information:
$$\Delta\vec{s}_t = \vec{s}_t - \vec{s}_{t_0} \tag{1}$$
3) a) If the norm of this displacement vector is larger than a given threshold, i.e. if $||\Delta\vec{s}_t|| > s_{thr}$, calculate the changes in distances by linear regression over the saved measurements within $\Delta t = t - t_0$ as follows:
$$\bar{d}_{ij} = \tfrac{1}{|T|}\ \textstyle\sum_{r\in T} d_{ij,r} \tag{2}$$
$$\bar{t} = \tfrac{1}{|T|}\ \textstyle\sum_{r\in T} r \tag{3}$$
$$\Delta d_{ij} = \frac{\sum_{r\in T}(r-\bar{t})\ (d_{ij,r}-\bar{d}_{ij})}{\sum_{r\in T}(r-\bar{t})^2}\ \Delta t \tag{4}$$
Note, that $|T| = |M_j|$. Go to Step (4).
   b) If Δt is larger than a threshold $\Delta t > t_{thr}$, set t_0 to be t and s_{t_0} to be s_t, and go back to Step (1). In this case the agent did not move considerably within Δt, and we therefore discard such measurements.
   c) Otherwise, go back to Step (2). Continue measuring.
4) Select the row k in $\mathcal{S}$, which is most similar to $\Delta\vec{s}_t$, by $k = \mathrm{argmax}_{r\in\{1,2,3\}}\{|\mathcal{S}_{r*} \bullet \Delta\vec{s}_t|\}$.
Replace row k in matrix $\mathcal{S}$ with the normalized displacement vector $\frac{\Delta\vec{s}_t}{\|\Delta\vec{s}_t\|}$, and replace row k in matrix $\mathcal{D}$ with the vector $\langle\frac{\Delta d_{i1}}{\|\Delta\vec{s}_t\|},\ldots,\frac{\Delta d_{i|A|}}{\|\Delta\vec{s}_t\|}\rangle$.
5) Set t_0 to t and s_{t_0} to s_t, and start again at (1) and keep updating rows in the matrices $\mathcal{S}$ and $\mathcal{D}$, representing recent displacements of agent i and associated changes in distance measurements.

Note that relative position $\vec{s}_t$ is obtained by a state estimator based on IMU data and similar sensors. State estimation is therefore prone to sensor noise, and might drift over time. However, our overall approach is immune to such drifts since it only depends on displacements $\Delta\vec{s}_t$ that happen in time duration Δt, which is bounded by t_{thr}.

C. *Exploration*

As the matrices $\mathcal{S}$ and $\mathcal{D}$ are initially empty, the agents first need to perform some exploration: move in some non-optimal direction, measure inter-agent distances, and populate the matrices with that information. This is done mainly in the startup phase. Later, the moves are the computed control actions. We also keep track of when each row was updated. If a row becomes older than some threshold t_{old}, it is deleted. Exploration is performed according to the following rules ($\times$ denotes the cross product of two vectors, and $\bullet$ denotes the dot product):

1) If all rows of $\mathcal{S}$ and $\mathcal{D}$ are empty: Sample three random variables as the components of vector $\vec{r}$. Apply action $\vec{q}_{expl,1} = w_{expl}\ \frac{\vec{r}}{\|\vec{r}\|}$, where w_{expl} is a scaling parameter.
2) Only $\mathcal{S}_{1*}$ is not empty: Sample random vector $\vec{r}$ as in step (1). Apply the vector $\vec{q}_{expl,2} = w_{expl}\ \frac{\mathcal{S}_{1*}\times\vec{r}}{\|\mathcal{S}_{1*}\times\vec{r}\|}$ as action (which is by construction orthogonal to $\mathcal{S}_{1*}$).
3) Only $\mathcal{S}_{1*}$, and $\mathcal{S}_{2*}$ are not empty: Apply the vector $\vec{q}_{expl,3} = w_{expl}\ \frac{\mathcal{S}_{1*}\times\mathcal{S}_{2*}}{\|\mathcal{S}_{1*}\times\mathcal{S}_{2*}\|}$ as action (which is by construction orthogonal to $\mathcal{S}_{1*}$ and $\mathcal{S}_{2*}$).
4) All entries in $\mathcal{S}$ are non-empty:
   a) If there exist two dependent rows k and $m \neq k$ (pointing in a similar direction ($|\mathcal{S}_{k*}\bullet\mathcal{S}_{m*}| > \kappa_{thr}$), where κ_{thr} is a parameter to quantify this similarity): Apply the

vector $\vec{q}_{expl,4} = w_{expl} \frac{\mathcal{S}_{k*} \times \mathcal{S}_{m*}}{\|\mathcal{S}_{k*} \times \mathcal{S}_{m*}\|}$ as action (which is by construction orthogonal to these row vectors).

b) If such rows do not exist, the vectors $\mathcal{S}_{1*}$, $\mathcal{S}_{2*}$, $\mathcal{S}_{3*}$ are linearly independent; i.e. they are all different ($\mathcal{S}_{1*} \neq \mathcal{S}_{2*} \neq \mathcal{S}_{3*}$, ensured by 4a), nonzero ($\mathcal{S}_{1*} \neq \vec{0}$, $\mathcal{S}_{2*} \neq \vec{0}$, $\mathcal{S}_{3*} \neq \vec{0}$, ensured by $||\Delta\vec{s}_t|| > s_{thr}$), and not in a common plane ($(\mathcal{S}_{1*} \times \mathcal{S}_{2*}) \bullet \mathcal{S}_{3*} \neq \vec{0}$, ensured by 2, 3, and 4). $\vec{0}$ is used as shorthand notation for $(0,0,0)^T$. Exploration is finished and the controller can go to exploitation mode.

D. Exploitation

The dictionary represents the partial derivative of distance measurements along spatial directions $\mathcal{S}_{k*,k\in\{1,2,3\}}$ yielding associated values $\mathcal{D}_{k*}$. For any displacement vector $\vec{q}$, we compute the estimated change in distances to each other agent $\widehat{\Delta d_{i*}}(\vec{q})$ by a first order Taylor approximation:

$$\widehat{\Delta d_{i*}} = \lambda_1 \; \mathcal{D}_{1*} + \lambda_2 \; \mathcal{D}_{2*} + \lambda_3 \; \mathcal{D}_{3*} \tag{5}$$

(here, addition and multiplication are applied element-wise). Since $\mathcal{S}_{1*}$, $\mathcal{S}_{2*}$, $\mathcal{S}_{3*}$ form a basis, the unique coefficients λ_1, λ_2, λ_3 of the linear combination are given by:

$$\vec{q} = \lambda_1 \; \mathcal{S}_{1*} + \lambda_2 \; \mathcal{S}_{2*} + \lambda_3 \; \mathcal{S}_{3*} \tag{6}$$

Likewise we can compute the estimated distances after the agent i would have moved by displacement vector $\vec{q}$:

$$\widehat{d_{ij}}(\vec{q}) = d_{ij} + \widehat{\Delta d_{ij}}(\vec{q}) \tag{7}$$

E. High-Level Controller

Our high-level controller works with any distance-based cost function. It makes use of distance measurements provided by our *distance estimator* to choose the best action from a set of candidate actions as the resulting action.
The set of candidate actions Q is defined as follows:

$$E = \left\{ (x,y,z)^T \,\middle|\, x,y,z \in \{-1,0,1\} \right\} \setminus \vec{0} \tag{8}$$

$$Q = \left\{ \epsilon_Q \, n \, \frac{\vec{q}}{||\vec{q}||} \,\middle|\, \vec{q} \in E, n \in \{1,..,N_Q\} \right\} \cup \vec{0} \tag{9}$$

This gives a set of $26 \cdot N_Q + 1$ points which are spaced by a distance of ϵ_Q each in every direction, including diagonals. The estimated distances, if action $\vec{q}$ is taken, are estimated by Eq. (7). Over the set Q, the best action $\vec{q}_{best}$ is chosen by minimizing cost function c:

$$\vec{q}_{best} = \underset{\vec{q}\in Q}{\text{argmin}}\{c(\widehat{d_{i*}}(\vec{q}))\} \tag{10}$$

Each agent computes the desired position set-point at every time s.t. its local cost function (Eq. 10) is minimized. This set-point is passed as input to a low-level controller to control the drone's propeller motors, as it is also done in [8].

IV. APPLICATION TO DRONE FLOCKING

In this section, we introduce a distance-based cost function for flocking and describe quality metrics of flock formations.

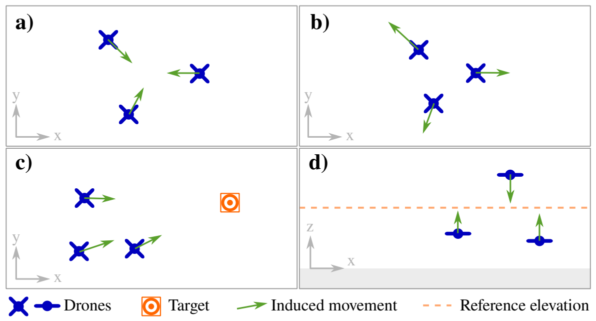

Fig. 3: Cost-function terms for drone flocking: **a**: *cohesion*, **b**: *separation*, **c**: *target-seeking*, and **d**: *elevation*.

A. Drone Flocking

Let A denote the set of agents. They are in flock formation if the distance between every pair of agents is range bounded; that is, the agents are neither too close to each other nor too far apart. To formulate this concept, we define a cost function that each agent tries to minimize in order to achieve the formation. Cost function based formulations for flocking were studied in [5], [7], [45]. These works show the usefulness of such a formulation for achieving flock formations and discuss properties w.r.t. to non-collision and non-dispersion.

Consider drones i and j. Let d_{ij} denote the distance between drones i and j, as it appears to drone i (measured from their centers of mass). Let l_i denote the distance between drone i and the target location p_{tar}. The radius of the circumscribed sphere of a drone is denoted by r_{drone}. Drone i has only access to distances of a subset $H_i \subseteq A$ of the drones, called its neighbourhood. Hence, the local cost function is parameterized by i, and uses only the distances to drones in H_i. We define H_i using a neighborhood radius parameter r_H, as follows:

$$H_i = \{j \,|\, d_{ij} < r_H, j \in A \setminus \{i\}\} \tag{11}$$

The tuple of distances from drone i to drones in H_i is denoted by d_{H_i}. Our cost function c_a (a for aerial) is defined for every drone $i \in A$ as follows:

$$c_a(d_{H_i}, l_i) = c_{coh}(d_{H_i}) + c_{sep}(d_{H_i}) + c_{tarA}(l_i) \tag{12}$$

a) Cohesion term:

$$c_{coh}(d_{H_i}) = \frac{\omega_{coh}}{|H_i|} \sum_{j\in H_i} {d_{ij}}^2 \tag{13}$$

As $c_{coh}(d_{H_i})$ increases when drones drift apart, this term keeps drones together. Each term includes a subscripted ω as a weight.

b) Separation term:

$$c_{sep}(d_{H_i}) = \frac{\omega_{sep}}{|H_i|} \sum_{j\in H_i} \frac{1}{max(d_{ij} - 2r_{drone} - \chi_{sep}, \hat{0})^2} \tag{14}$$

The *separation* term keeps drones apart, as it increases when drones get closer. Here $\hat{0}$ denotes a very small positive value. Function $max(., \hat{0})$ ensures a strictly positive denominator. The slack parameter χ_{sep} is used to influence the minimum distance between two drones. This parameter is different from previous works [5], [8], [12]. During experimental validation, this showed significant impact on avoiding collisions.

c) Aerial target-seeking term:

$$c_{tarA}(l_i) = \omega_{tar} \; l_i \tag{15}$$

Our *aerial target-seeking* lets us move the flock towards a specified target location. In simulation, we move the target location, while on real hardware, we switch between different targets, where only the active one is used in the cost function. As all agents have the same target location, we assume shared knowledge of that information. However, the control algorithm itself is still fully distributed. Instead of a target location, one of the drones could be designated as a leader. This leader could have additional sensors to obtain information about its absolute position, to employ an alternative control scheme.

B. *Altitude-Aided Drone Flocking*

The size of the indoor space constrains movement in all three dimensions, but typical-room height is most restrictive. We also want to place our target on the ground; as such, the flock should reach this target indirectly, at a certain altitude. We therefore describe an alternative cost function c_g (Eq. 16) for *altitude-aided drone flocking*. In real-world environments, altitude measurements can be gathered by barometers or altimeters (e.g. downward-facing range sensors).

$$\begin{aligned} c_g(d_{H_i}, l_i, z_i) = c_{coh}(d_{H_i}) + c_{sep}(d_{H_i}) + \\ c_{tarG}(l_i, z_i) + c_{elev}(z_i) \end{aligned} \tag{16}$$

d) Elevation term:

$$c_{elev}(z_i) = \omega_{elev}\ (\zeta_{elev} - z_i)^2 \tag{17}$$

Here z_i denotes the z-coordinate of the position of agent i, which is it's altitude. This term keeps the flock at a certain altitude, which is determined by the reference elevation ζ_{elev}.

e) Ground target-seeking term:
The distance l_i of drone i to the current target is projected to the x-y plane to compute the cost term c_{tarG}:

$$\tilde{l}_i(l_i, z_i) = \sqrt{max({l_i}^2 - {z_i}^2, 0)} \tag{18}$$

$$c_{tarG}(l_i, z_i) = c_{tarA}(\tilde{l}_i(l_i, z_i)) \tag{19}$$

C. *Flock-Formation Quality Metrics*

We define metrics, and associated constraints, to assess the flock quality achieved by DDC.

a) Collision avoidance: The distance between all pairs of drones $\text{dist}(A)$ must remain above a specified threshold $\text{dist}_{thr} = 2 \cdot r_{drone} + r_{safety}$, where r_{safety} is a safety margin.

$$\text{dist}(A) = \min_{i,j \in A; i \neq j} d_{ij} \tag{20}$$

$$\text{dist}(A) \geq \text{dist}_{thr} \tag{21}$$

b) Compactness: The radius of the sphere circumscribing all agents would be the ideal metric. This would require us, however, to know each agent's position in a global coordinate system, which we do not have. We thus instead use the maximum radius of the pairwise inter-agent circumscribed spheres:

$$\text{comp}(A) = \frac{1}{2} \max_{i,j \in A; i \neq j} d_{ij} \tag{22}$$

c) Target reaching: To assess the quality of target-seeking, we determine, for each target k, the average distance to the target:

$$\text{tar}_k(A) = \frac{1}{|A|} \sum_{i \in A} l_{ik} \tag{23}$$

When using ground a target-seeking term, l_{ik} is replaced by $\tilde{l}_{ik}$.

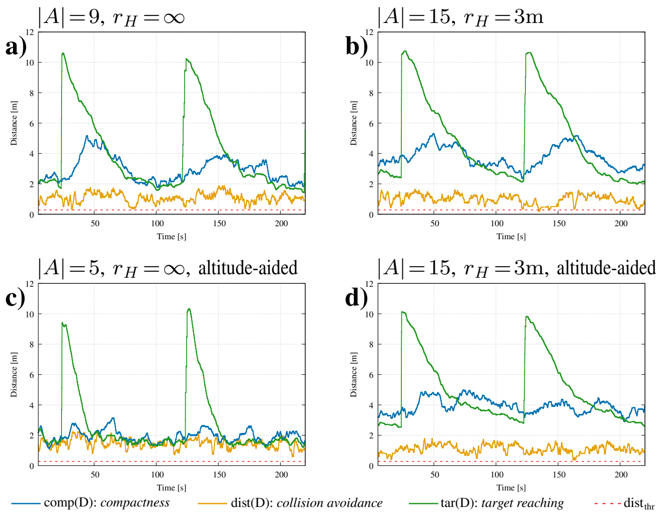

Fig. 4: Metrics over time for representative simulation experiments. The target is updated with a new position at $t_1 = 20$ s and $t_2 = 120$ s, resulting in a saw-tooth shape. **a**: drone flocking with 9 agents and global neighborhood, **b**: 15 agents and local neighborhood, **c**: altitude-aided drone flocking with 5 agents and global neighborhood, **d**: 15 agents and local neighborhood.

V. EVALUATION

We evaluate DDC's performance in simulation and on real hardware using *Crazyflie 2.1* quadcopters [13]. In both cases, we use the same software implementation, with only minor adjustments of empirically determined parameters.

A. *Simulation Experiments*

As simulation environment for our experiments, we used *crazys* [42]. This is based on the *Gazebo* [46] physics and visualization engine and the Robotic Operating System (ROS) [47]. We implemented DDC (the gray blocks shown in Figure 1) in C++, as a separate ROS node. DDC receives measured distances $d^{\text{raw}}_{ij,t}$, and relative position information $\vec{s}_t$. Based on the cost function, it calculates the next action $\vec{u}$ and passes it to the low-level controller. Controller parameters were determined empirically from a range of values for each parameter by manual inspection of performance metrics.

As noise is critical (cf. III-A), we modeled it in simulation by a sum of low-passed white noise plus additional white noise. We tested different noise levels to assess DDC's robustness.

For evaluation, we used two positions sequentially supplied as target location: After the drones formed a flock around the starting position $(x, y, z) = (0, 0, \zeta_{elev})$, they move to $(10\text{m}, 0, \zeta_{elev})$ and then to $(10\text{m}, 10\text{m}, \zeta_{elev})$. We performed simulations with $|A| = 5$, 9, and 15 drones. Quality metrics over time for representative simulation runs are shown in Figure 4. The metrics for *compactness* (blue) and *collision avoidance* (orange) show that DDC successfully maintains a stable flock without collisions. The flock moves towards the target locations, as shown by the decreasing *target-reaching* metric (green). While moving, *compactness* (blue) is temporarily degraded.

B. *Comparison to Cyclic Stop-And-Go Strategy*

We compared DDC to the *cyclic stop-and-go strategy* [11], consisting of three alternating periods: *identification*, *control*, and *resting*. This method allows only a subset of agents, which

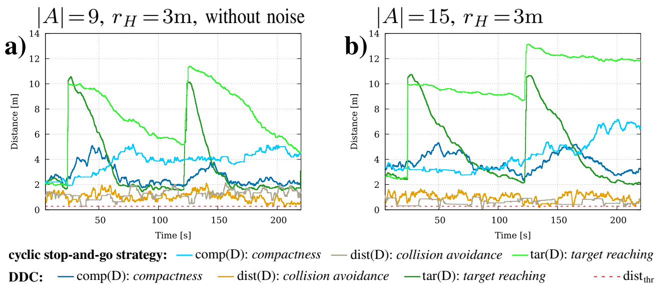

Fig. 5: Comparison of DDC and cyclic stop-and-go strategy [11], for simulations of drone flocking with local neighborhood. For the latter, *target reaching* (green) decays much more slowly, and *compactness* (blue) is considerably worse, than for DDC. **a**: 9 agents without distance measurement noise, **b**: 15 agents with same noise level as in all other simulations of DDC.

are not neighbours of each other, to move simultaneously (i.e. to perform *identification* or *control*). At the same time, all other agents are in *resting* phase, where they need to remain stationary. This is easy in simulation, but hard to achieve in the real world.

As [11] does not provide an implementation, we took it upon ourselves to implement it: In the *identification* phase, we use three orthogonal movements and true-range multilateration [48] to estimate other agent's relative positions. For the *control* phase, we use a variant of SPC [8], with the same parameters and cost function weights as in DDC. For arbitration of the different phases, we allowed for central coordination,even though in our problem statement agents are not able to exchange such information. In our DDC approach, such coordination is not necessary. Comparison in Fig. 5 shows that this method is susceptible to sensing noise and much slower in reaching the target.

C. Hardware Experiments

We experimented with *Crazyflie 2.1* quadcopters [13] with *Loco-positioning* deck, featuring *DWM1000* UWB modules [43] for distance measurements and *Flow deck v2* (with z-ranging altimeter and optical flow module); see Figure 6. To determine inter-drone distances, we used a ranging software implementation [44]. For each drone, the measured distances $d_{ij,t}^{\text{raw}}$ and relative position information $\vec{s}_t$ are transmitted to a computer. There, DDC is executed in a separate ROS node for each drone (same as in simulation), the next action $\vec{u}$ is computed, which is then transmitted to the drone. Even though it is executed on the same computer for all drones, DDC is fully distributed, as there is no additional information exchange between the individual nodes.

We evaluated DDC in three scenarios. In Figure 7, the starting locations and traces of representative experiments are visualized. In scenario LINE, the drones start in a line. As the drones move into a more compact formation, the *compactness* metric improves (becomes lower) over time. For JOIN, two drones start further away from target 0. They join the other three, which are already closer to the target. For MOVE, the drones start around target 0, indicated by *forming flock*. Then, when moving towards target 1 in the next section, the average distance to the target continuously decreases. In the last section, the flock returns to target 0. While moving, *compactness* is temporarily degraded. In all of these scenarios, the metric for *collision avoidance* stays above the threshold. A video of these hardware experiments is appended.

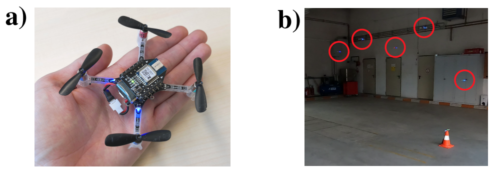

Fig. 6: Drones used for hardware experiments: **a**: *Crazyflie 2.1* quadcopter **b**: Five drones (highlighted in red circles) while testing scenario JOIN.

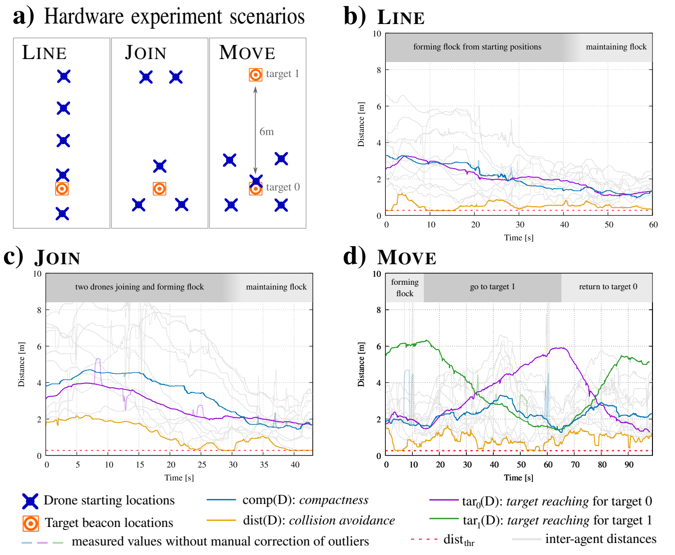

Fig. 7: Hardware experiments. **a** Scenarios: LINE, forming a flock from starting positions along a line; JOIN, two agents joining three agents at the target location; MOVE, a flock moves between two alternating target locations. Plots of metrics and inter-agent distances for scenarios **b**: LINE, **c**: JOIN, and **d**: MOVE.

VI. CONCLUSION

We introduced DDC, a MAS-formation-control approach which is fully distributed, and solely based on scalar-distance measurements, and local-position estimation. We demonstrated DDC on drone-flocking with target-seeking. To validate DDC, we took a two-pronged approach: (1) We performed simulation experiments using a physics engine with a detailed drone model, and (2) We performed experiments with real drones, specifically, *Crazyflie 2.1* quadcopters. Our results demonstrate DDC's ability to form and maintain a flock, and move towards a target location. To the best of our knowledge, we are the first to demonstrate such a controller on aerial MASs. While DDC is able to satisfy the specified performance metrics, future work will focus on ensuring that the flock reaches the target location by a given deadline and extending it with obstacle avoidance capabilities. We plan to perform analysis on stability and guarantees for non-collision and non-dispersion. We will also explore replacing our dictionary structure with more general attention-based models and learning techniques.

ACKNOWLEDGMENT

A.B. was supported by a research exchange scholarship from Austrian Marshall Plan Foundation. R.G. was partially supported by EU-H2020 NimbleAI and AT-BMBWF DK-RES and CPS/IoT Ecosystem. This work was supported in part by NSF grants CCF-1954837, CCF-1918225, CPS-1446832, ITE-2134840, and OIA-2040599. We thank the volunteer fire department of Sieghartskirchen for letting us use their vehicle hall for our hardware experiments.

References

[1] J. P. Queralta, J. Taipalmaa, B. Can Pullinen, *et al.*, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 8, pp. 191 617–191 643, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3030190.

[2] D. Câmara, "Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios," in *2014 IEEE Conference on Antenna Measurements Applications (CAMA)*, 2014, pp. 1–4. DOI: 10.1109/CAMA.2014.7003421.

[3] N. Michael, S. Shen, K. Mohta, *et al.*, "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)*, Jul. 2012, pp. 33–47.

[4] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, Apr. 2, 2020, ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00036.

[5] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87, New York, NY, USA: Association for Computing Machinery, 1987, pp. 25–34, ISBN: 0897912276. DOI: 10.1145/37401.37406.

[6] 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. DOI: 10.1109/TAC.2005.864190.

[7] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stability of flocking motion," University of Pennsylvania, Tech. Rep., 2003. [Online]. Available: https://www.georgejpappas.org/papers/boids03.pdf.

[8] A. Brandstätter, S. A. Smolka, S. D. Stoller, A. Tiwari, and R. Grosu, "Multi-agent spatial predictive control with application to drone flocking," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1221–1227. DOI: 10.1109/ICRA48891.2023.10160617.

[9] A. F. G. Ferreira, D. M. A. Fernandes, A. P. Catarino, and J. L. Monteiro, "Localization and positioning systems for emergency responders: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2836–2870, 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2703620.

[10] C. Laoudias, A. Moreira, S. Kim, S. Lee, L. Wirola, and C. Fischione, "A survey of enabling technologies for network localization, tracking, and navigation," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3607–3644, 2018, ISSN: 1553-877X, 2373-745X. DOI: 10.1109/COMST.2018.2855063.

[11] M. Cao, C. Yu, and B. D. Anderson, "Formation control using range-only measurements," *Automatica*, vol. 47, no. 4, pp. 776–781, Apr. 2011, ISSN: 00051098. DOI: 10.1016/j.automatica.2011.01.067.

[12] A. Brandstätter, S. A. Smolka, S. D. Stoller, A. Tiwari, and R. Grosu, "Towards drone flocking using relative distance measurements," in *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning*, T. Margaria and B. Steffen, Eds., Cham: Springer Nature Switzerland, 2022, pp. 97–109, ISBN: 978-3-031-19759-8. DOI: 10.1007/978-3-031-19759-8_7.

[13] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "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)*, 2017, pp. 37–42. DOI: 10.1109/MMAR.2017.8046794.

[14] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[15] M. Florek, M. Huba, F. Duchoň, J. Šovčík, and M. Kajan, "Comparing approaches to quadrocopter control," in *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2014, pp. 1–6. DOI: 10.1109/RAAD.2014.7002241.

[16] G. A. Garcia, A. R. Kim, E. Jackson, S. S. Keshmiri, and D. Shukla, "Modeling and flight control of a commercial nano quadrotor," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 524–532. DOI: 10.1109/ICUAS.2017.7991439.

[17] J. Lu, H. Shen, L. Pan, X. Zhang, B. Tian, and Q. Zong, "Autonomous flight for multi-UAV in GPS-denied environment," in *Proceedings of 2021 5th Chinese Conference on Swarm Intelligence and Cooperative Control*, Z. Ren, M. Wang, and Y. Hua, Eds., vol. 934, Series Title: Lecture Notes in Electrical Engineering, Singapore: Springer Nature Singapore, 2023, pp. 892–901. DOI: 10.1007/978-981-19-3998-3_85.

[18] M. Saska, T. Baca, J. Thomas, *et al.*, "System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization," *Autonomous Robots*, vol. 41, no. 4, pp. 919–944, Apr. 2017, ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-016-9567-z.

[19] K. Mohta, M. Watterson, Y. Mulgaonkar, *et al.*, "Fast, autonomous flight in GPS-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, Jan. 2018, ISSN: 15564959. DOI: 10.1002/rob.21774.

[20] D. O. Wheeler, D. P. Koch, J. S. Jackson, *et al.*, "Relative navigation of autonomous GPS-degraded micro air vehicles," *Autonomous Robots*, vol. 44, no. 5, pp. 811–830, May 2020, ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-019-09899-4.

[21] Z. Hepeng, D. Jian, Y. Han, W. Xinghu, and J. Haibo, "Varying-coefficient based distributed formation control of multiple uavs," in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 5524–5528. DOI: 10.23919/CCC52363.2021.9549453.

[22] K.-K. Oh and H.-S. Ahn, "Distance-based undirected formations of single-integrator and double-integrator modeled agents in n-dimensional space: DISTANCE-BASED UNDIRECTED FORMATIONS," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 12, pp. 1809–1820, Aug. 2014, ISSN: 10498923. DOI: 10.1002/rnc.2967.

[23] M. Aranda, G. Lopez-Nicolas, C. Sagues, and M. M. Zavlanos, "Distributed formation stabilization using relative position measurements in local coordinates," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3925–3935, Dec. 2016, ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2016.2527719.

[24] S.-M. Kang, M.-C. Park, and H.-S. Ahn, "Distance-based cycle-free persistent formation: Global convergence and experimental test with a group of quadcopters," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 1, pp. 380–389, Jan. 2017, ISSN: 0278-0046, 1557-9948. DOI: 10.1109/TIE.2016.2606585.

[25] X. Lan, W. Xu, and Y.-S. Wei, "Adaptive 3d distance-based formation control of multiagent systems with unknown leader velocity and coplanar initial positions," *Complexity*, vol. 2018, pp. 1–9, Sep. 6, 2018, ISSN: 1076-2787, 1099-0526. DOI: 10.1155/2018/1814653.

[26] B. Wu, D. Wang, and E. K. Poh, "Decentralized control for satellite formation using local relative measurements only," in *IEEE ICCA 2010*, 2010, pp. 661–666. DOI: 10.1109/ICCA.2010.5524080.

[27] R. Babazadeh and R. Selmic, "Distance-based multiagent formation control with energy constraints using sdre," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 1, pp. 41–56, 2020. DOI: 10.1109/TAES.2019.2910361.

[28] F. He, Y. Wang, Y. Yao, L. Wang, and W. Chen, "Distributed formation control of mobile autonomous agents using relative position measurements," *IET Control Theory & Applications*, vol. 7, no. 11, pp. 1540–1552, Jul. 2013, ISSN: 1751-8652, 1751-8652. DOI: 10.1049/iet-cta.2012.1034.

[29] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, Mar. 2015, ISSN: 00051098. DOI: 10.1016/j.automatica.2014.10.022.

[30] B. Jiang, M. Deghat, and B. D. O. Anderson, "Simultaneous velocity and position estimation via distance-only measurements with application to multi-agent system control," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 869–875, Feb. 2017, ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2016.2558040.

[31] A. Chakraborty, K. Brink, R. Sharma, and L. Sahawneh, "Relative pose estimation using range-only measurements with large initial uncertainty," in *2018 Annual American Control Conference (ACC)*, Milwaukee, WI, USA: IEEE, Jun. 2018, pp. 5055–5061, ISBN: 978-1-5386-5428-6. DOI: 10.23919/ACC.2018.8431743.

[32] B. D. O. Anderson and C. Yu, "Range-only sensing for formation shape control and easy sensor network localization," in *2011 Chinese Control and Decision Conference (CCDC)*, Mianyang, China: IEEE, May 2011, pp. 3310–3315, ISBN: 978-1-4244-8737-0. DOI: 10.1109/CCDC.2011.5968829.

[33] R. Sharma and C. Taylor, "Cooperative navigation of MAVs in GPS denied areas," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seoul: IEEE, Aug.

2008, pp. 481–486, ISBN: 978-1-4244-2143-5. DOI: 10.1109/MFI.2008.4648041.

[34] Y. Zou, C. Wen, and M. Guan, "Distributed adaptive control for distance-based formation and flocking control of multi-agent systems," *IET Control Theory & Applications*, vol. 13, no. 6, pp. 878–885, Apr. 2019, ISSN: 1751-8652, 1751-8652. DOI: 10.1049/iet-cta.2018.6001.

[35] K. Guo, X. Li, and L. Xie, "Ultra-wideband and odometry-based cooperative relative localization with application to multi-UAV formation control," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2590–2603, Jun. 2020, ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TCYB.2019.2905570.

[36] B. Jiang, B. D. O. Anderson, and H. Hmam, "3-d relative localization of mobile systems using distance-only measurements via semidefinite optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 3, pp. 1903–1916, Jun. 2020, ISSN: 0018-9251, 1557-9603, 2371-9877. DOI: 10.1109/TAES.2019.2935926.

[37] E. Stump, V. Kumar, B. Grocholsky, and P. M. Shiroma, "Control for localization of targets using range-only sensors," *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 743–757, Jun. 2009, ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364908098559.

[38] Y. Cai and Y. Shen, "An integrated localization and control framework for multi-agent formation," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1941–1956, Apr. 2019, ISSN: 1053-587X, 1941-0476. DOI: 10.1109/TSP.2019.2897968.

[39] M. Cao, C. Yu, and B. D. O. Anderson, "Coordination with the leader in a robotic team without active communication," in *2009 17th Mediterranean Conference on Control and Automation*, Thessaloniki, Greece: IEEE, Jun. 2009, pp. 252–257, ISBN: 978-1-4244-4684-1. DOI: 10.1109/MED.2009.5164548.

[40] S.-M. Kang, M.-C. Park, B.-H. Lee, and H.-S. Ahn, "Distance-based formation control with a single moving leader," in *2014 American Control Conference*, Portland, OR, USA: IEEE, Jun. 2014, pp. 305–310. DOI: 10.1109/ACC.2014.6858587.

[41] R. Suttner and Z. Sun, "Formation shape control based on distance measurements using lie bracket approximations," *SIAM Journal on Control and Optimization*, vol. 56, no. 6, pp. 4405–4433, Jan. 2018, ISSN: 0363-0129, 1095-7138. DOI: 10.1137/18M117131X.

[42] G. Silano, E. Aucone, and L. Iannelli, "CrazyS: A software-in-the-loop platform for the Crazyflie 2.0 nano-quadcopter," in *2018 26th Mediterranean Conference on Control and Automation (MED)*, 2018, pp. 1–6. DOI: 10.1109/MED.2018.8442759.

[43] Qorvo Inc. "DWM1000 – 3.5 - 6.5 GHz Ultra-Wideband (UWB) Transceiver Module." (2021), [Online]. Available: https://www.qorvo.com/products/d/da007948 (visited on 09/14/2023).

[44] F. Shan, H. Huo, J. Zeng, Z. Li, W. Wu, and J. Luo, "Ultra-wideband swarm ranging protocol for dynamic and dense networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2834–2848, 2022. DOI: 10.1109/TNET.2022.3186071.

[45] U. Mehmood, N. Paoletti, D. Phan, *et al.*, "Declarative vs rule-based control for flocking dynamics," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18, Pau, France: Association for Computing Machinery, 2018, pp. 816–823, ISBN: 9781450351911. DOI: 10.1145/3167132.3167222.

[46] N. Koenig and A. Howard, "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, 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[47] Stanford Artificial Intelligence Laboratory et al., *Robotic operating system*. [Online]. Available: https://www.ros.org (visited on 09/14/2023).

[48] B. T. Fang, "Trilateration and extension to global positioning system navigation," *Journal of Guidance, Control, and Dynamics*, vol. 9, no. 6, pp. 715–717, Nov. 1986, ISSN: 0731-5090, 1533-3884. DOI: 10.2514/3.20169.