IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Decentralized Swarm Control Via SO(3) Embeddings for 3D Trajectories

Dimitria Silveria, Kleber Cabral, *Member, IEEE*, Peter T. Jardine, *Member, IEEE*, and Sidney Givigi, *Senior Member, IEEE*

***Abstract*—This letter presents a novel decentralized approach for achieving emergent behavior in multi-agent systems with minimal information sharing. Based on prior work in simple orbits, our method produces a broad class of stable, periodic trajectories by stabilizing the system around a Lie group-based geometric embedding. Employing the Lie group SO(3), we generate a wider range of periodic curves than existing quaternion-based methods. Furthermore, we exploit SO(3) properties to eliminate the need for velocity inputs, allowing agents to receive only position inputs. We also propose a novel phase controller that ensures uniform agent separation, along with a formal stability proof. Validation through simulations and experiments showcases the method's adaptability to complex low-level dynamics and disturbances.**

***Index Terms*—Autonomous aerial vehicles, decentralized control, lie groups, swarm robotics.**

I. Introduction

A SWARM is a decentralized form of multi-agent system (MAS) that displays emergent behavior —that is, complex behaviors arising from local interactions governed by simple rules without centralized coordination [1]. Swarm agents are often robotic platforms such as uncrewed aerial vehicles (UAVs) used in various domains, including entertainment, surveillance, and defense.

This letter addresses the challenge of generating stable, closed 3D formations around a fixed point for UAVs using only local position information. Such formations are relevant in dynamic capture, surveillance, and mobbing scenarios [2], and relate to applications such as lattice formation [3], encirclement [4], epitrochoidal motion [5], target enclosing [6], and other dynamic patterns [7].

Existing approaches often rely on consensus-based algorithms. For example, [8] uses consensus control and heading error compensation for 2D circular trajectories, with particle swarm optimization (PSO) applied to tune controller gains. However, this method scales poorly, lacks real-world validation, and is vulnerable to agent loss. Similarly, [9] applies consensus-based optimization for simulated circular patrolling.

Prior work confines trajectories to circles due to their simplicity and inherent collision avoidance through phase spacing [10]. Some exceptions exist, such as [5], which introduces 3D hypotrochoidal and epitrochoidal trajectories. However, these paths depend on initial conditions, assume double integrator dynamics, lack collision guarantees, and are not validated in physical systems.

To overcome the above limitations, [11] introduced an embedding approach where agents follow virtual circular paths with uniform phase spacing enforced via angular velocity control. The circular embedding is a projection of the multi-agent coordination problem into a lower-dimensional space, enabling the use of linear control laws that would otherwise not be directly applicable. Actual trajectories are derived through quaternion-based 3D deformations of this circular trajectory. While this enables the representation of certain complex 3D shapes and adapts to agent loss or addition, it suffers from ambiguities in even-sized swarms: quaternion rotation may map two antipodal agents to the same point, thereby causing a risk of collision. Additionally, this approach is limited to generating lemniscate curves and requires both position and velocity inputs, which constrains the range of applications. Moreover, the previous work restricted its validation to simulation. The authors identify these shortcomings as areas for future work.

This letter leverages the concept of circular embedding proposed in [11] and makes the following contributions:

- A Lie group SO(3)-based trajectory deformation method that generalizes better than quaternion-based approaches, enabling linear control in lower-dimensional space while supporting a greater range of 3D formations, not achieved in the previous literature.
- A resolution to the problem of antipodal collisions identified in previous work.
- A position-only reference generation technique, allowing for broader applicability to UAVs without velocity control (e.g., Crazyflies [12]).
- A scalable, electrostatic repulsion-inspired phase separation controller that maintains uniform angular separation with formal guarantees of stability.
- Experimental validation on physical quadcopters, bridging the gap between theory and practice and demonstrating robustness to unmodeled dynamics.

Received 5 July 2025; accepted 5 November 2025. Date of publication 19 November 2025; date of current version 28 November 2025. This article was recommended for publication by Associate Editor T. Nascimento and Editor G. Loianno upon evaluation of the reviewers' comments. This work was supported by NSERC under Grant RGPIN-2022-01277 and Grant ALLRP-576937-22. *(Corresponding author: Dimitria Silveria.)*

Dimitria Silveria is with the Department of Electrical and Computer Engineering, Ingenuity Labs Research Institute, Queen's University, Kingston, ON K7L 3N9, Canada (e-mail: dimitria.s@queensu.ca).

Kleber Cabral, Peter T. Jardine, and Sidney Givigi are with the School of Computing and the Ingenuity Labs Research Institute, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: kleber.cabral@queensu.ca; p.jardine@queensu.ca; sidney.givigi@queensu.ca).

Digital Object Identifier 10.1109/LRA.2025.3634882

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

The remainder of the letter is organized as follows. Section II introduces the relevant Lie group concepts. Section III details the controller design for generating periodic formations. Stability analysis is presented in Section IV. Section V outlines the experimental setup for hardware validation, while Section VI reports results for both simulation and physical experiments. Finally, Section VII concludes the letter and outlines directions for future work.

II. Preliminaries

This section provides a concise overview of the mathematical foundations relevant to this letter, focusing on the Lie group $\mathrm{SO}(3)$ and its associated Lie algebra $\mathfrak{so}(3)$. For a more comprehensive treatment, see [13], [14], [15].

The group comprising all 3D rotation matrices, including the identity, is defined as $\mathrm{SO}(3) := \{\boldsymbol{R} \in \mathbb{R}^{3\times 3} \mid \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I} \text{ and } \det(\boldsymbol{R}) = +1\}$. Its tangent space at the identity, $T_1\mathrm{SO}(3)$, forms the Lie algebra, which consists of skew-symmetric matrices and is defined as $\mathfrak{so}(3) := \{\boldsymbol{\Omega} \in \mathbb{R}^{3\times 3} \mid \boldsymbol{\Omega}^\top = -\boldsymbol{\Omega}\}$.

Using $\mathrm{SO}(3)$ to represent rotations offers notable advantages over unit quaternions. Quaternion rotations are sign-ambiguous (i.e., q and $-q$ encode the same rotation). In contrast, $\mathrm{SO}(3)$ avoids such ambiguities and is free of numerical drift, eliminating the need for periodic renormalization [16].

Rotations in $\mathfrak{so}(3)$ can be expressed via the hat operator $(\cdot)^\wedge$, which maps a 3D angular velocity vector $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top$ into a skew-symmetric matrix:

$$\boldsymbol{\Omega} = \boldsymbol{\omega}^\wedge = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}. \tag{1}$$

The inverse mapping is given by the vee operator $(\cdot)^\vee$.

The exponential map converts elements of $\mathfrak{so}(3)$ into $\mathrm{SO}(3)$:

$$\boldsymbol{R} = e^{(\boldsymbol{\Omega})}. \tag{2}$$

In this letter, $\mathfrak{so}(3)$ serves as the mathematical foundation for a novel control strategy to generate periodic trajectories in multi-agent robotic swarms. Working in $\mathfrak{so}(3)$ allows us to specify unconstrained parametric deformations; this contrasts with unit quaternions and rotation matrices, which must satisfy norm constraints. Consequently, the elements in $\boldsymbol{\Omega} \in \mathfrak{so}(3)$ may contain any values in $\mathbb{R}$, or be expressed as functions of other system variables (e.g., parametric equations). If the matrix $\boldsymbol{\Omega}$ is mapped to a rotation matrix using (2), the resulting rotation can be used to *deform* a 2D circular trajectory into a more complex 3D shape, similar to the quaternion-based method in [11]. This flexibility in the range of values of $\boldsymbol{\Omega}$ expands the range of achievable 3D trajectories compared to quaternion-based approaches.

The swarm pipeline proposed in this letter controls the motion of agents in 3D by regulating their angular velocity in a 2D circular plane. The angular velocity is represented as a skew-symmetric matrix $\mathfrak{so}(3)$ and mapped to a rotation matrix using (2). Another advantage of this formulation is that it simplifies the low-level controller, requiring only position references rather than position and velocity as in [11]. This process is discussed in detail in Section III-D.

Additionally, the position noise and uncertainties can be encoded in the Lie algebra, where rotations are defined as linear disturbances that are projected back to the group space using the exponential map. This approach of encoding uncertainties in Lie algebras is widely used in state estimation, particularly with the invariant Kalman filter [17].

III. Controller Design

This section outlines the control system architecture for a swarm using $\mathrm{SO}(3)$ rotations, where the Lie Group embedding acts as a form of feedback linearization, enabling standard control of the desired periodic motion.

Each agent i is modeled as a particle in free space, characterized by its position ($\boldsymbol{x}_i \tilde{} \in \tilde{}\mathbb{R}^3$), velocity ($\boldsymbol{v}_i \tilde{} \in \tilde{}\mathbb{R}^3$), and control input ($\boldsymbol{u}_i \tilde{} \in \tilde{}\mathbb{R}^3$). The agent's dynamics is governed by the following second-order system:

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= \boldsymbol{v}_i, \\ \dot{\boldsymbol{v}}_i &= \boldsymbol{u}_i, \end{aligned} \tag{3}$$

where $\boldsymbol{u}_i = \ddot{\boldsymbol{x}}_i$ is the agent's acceleration input.

A. Lie Group Embedding

Let us introduce a geometric embedding based on Lie group theory. Consider a circular trajectory of radius r and phase angle $\phi \in [0, 2\pi)$. Let $\omega_x(\phi)$ and $\omega_y(\phi)$ define parametric angular velocities that deform this trajectory over time about the X and Y axes, respectively. These deformations are encoded in the following angular velocities vector in $\mathfrak{so}(3)$:

$$\boldsymbol{\omega}_{xy}(\phi) = \begin{bmatrix} \omega_x(\phi) & \omega_y(\phi) & 0 \end{bmatrix}^\top, \qquad \boldsymbol{\Omega}_{xy}(\phi) = \boldsymbol{\omega}_{xy}(\phi)^\wedge. \tag{4}$$

The exponential map transforms this Lie algebra element into a rotation matrix in $\mathrm{SO}(3)$:

$$\boldsymbol{R} = e^{\boldsymbol{\Omega}_{xy}(\phi)}. \tag{5}$$

where $\boldsymbol{R} \in SO(3)$ is the time-varying rotation that deforms the base circular trajectory into a 3D curve. Applying this transformation yields the deformed trajectory:

$$\boldsymbol{x} = \boldsymbol{R}\hat{\boldsymbol{x}}, \tag{6}$$

where

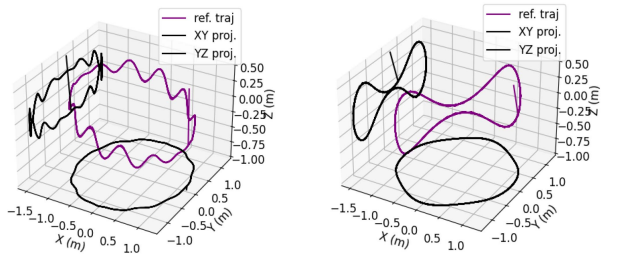
$$\hat{\boldsymbol{x}} = \begin{bmatrix} r\cos(\phi) & r\sin(\phi) & 0 \end{bmatrix}^\top \tag{7}$$

is the original circular trajectory in the XY-plane.

Fig. 1 shows four distinct trajectory deformations using this method. For example, the distortion shown in Fig. 1(b) results in a Dumbbell-shaped projection in the YZ plane – analogous to the XY-plane Dumbbell structure in [11] – demonstrating the versatility of the approach.

The distortion factor s governs the degree of deformation: $s = 0$ corresponds to an undistorted circle (i.e., a real-world circular trajectory), while larger values of s induce increasingly complex 3D structures. Parameters for each shape are provided in the figure captions.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) $\omega_x = s\sin(6\phi)\cos(6\phi)$, $\omega_y = s$, $\dot{\phi}_d = 0.8$ rad/s, $s = 0.3$

(b) $\omega_x = s\sin(\phi)\cos(\phi)$, $\omega_y = 0$, $\dot{\phi}_d = 2$ rad/s, $s = 1$

(c) $\omega_x = s\cos(2\phi)$, $\omega_y = s\cos^2(\phi)$, $\dot{\phi}_d = 0.5$ rad/s, $s = 0.6$

(d) $\omega_x = s\cos(3\phi)\sin(\phi)$, $\omega_y = 0.5s$, $\dot{\phi}_d = 1.8$ rad/s, $s = 0.9$

Fig. 1. Examples of 3D geometries generated using the method described in Section III-A along with XY and YZ projections. (a) $\omega_x = s\sin(6\phi)\cos(6\phi)$, $\omega_y = s$, $\dot{\phi}_d = 0.8$ rad/s, $s = 0.3$ (b) $\omega_x = s\sin(\phi)\cos(\phi)$, $\omega_y = 0$, $\dot{\phi}_d = 2$ rad/s, $s = 1$ (c) $\omega_x = s\cos(2\phi)$, $\omega_y = s\cos^2(\phi)$, $\dot{\phi}_d = 0.5$ rad/s, $s = 0.6$ (d) $\omega_x = s\cos(3\phi)\sin(\phi)$, $\omega_y = 0.5s$, $\dot{\phi}_d = 1.8$ rad/s, $s = 0.9$

B. Controller Overview

Consider an agent i following a 3D trajectory derived via the rotation (6) from a circular reference path with radius r_d and phase $\phi_i = (\omega_{z,d,i})t$, rotating about the Z axis. The virtual agent $\hat{\boldsymbol{x}}_i$ is defined by mapping the agent's position into the circular embedding via the inverse of (6):

$$\hat{\boldsymbol{x}}_i = \boldsymbol{R}_i{}^{\top}\boldsymbol{x}_i. \tag{8}$$

To achieve the desired behavior, we regulate $\hat{\boldsymbol{x}}_i$ to asymptotically track the reference circular trajectory:

$$\lim_{t\to\infty}\hat{\boldsymbol{x}}_i(t) = \begin{bmatrix} r_d\cos(\omega_{z,d}t) & r_d\sin(\omega_{z,d}t) & 0\end{bmatrix}^{\top}, \tag{9}$$

where $\omega_{z,d}$ is the common desired angular velocity for all agents, which is constant over time. Regulating this embedding indirectly regulates real-world coordinates $\boldsymbol{x}_i$. This control strategy is implemented in two stages: a phase controller and a position controller. These stages ensure, respectively, the convergence of both angular velocity and radial distance, and are designed to satisfy:

$$\lim_{t\to\infty}\omega_{z,d,i}(t) = \omega_{z,d}, \quad \lim_{t\to\infty} r_i(t) = r_d. \tag{10}$$

where $\omega_{z,d,i}(t)$ is the z angular velocity of the virtual agent i at the time instant t, and $r_i(t)$ is the radius of this same agent at the same time instant, calculated from the center of the circular embedding.

The phase controller governs the emergent behavior by adjusting each agent's angular velocity based on the phase differences with its leading and lagging neighbors. This promotes uniform angular spacing and avoids collisions in the embedding. Meanwhile, the linear position controller ensures each agent converges to the appropriate point on the desired trajectory.

C. Phase Control

The phase of agent i is $\phi_i = \arctan 2(\hat{x}_{y,i}, \hat{x}_{x,i})$, where $\hat{x}_{x,i}$ and $\hat{x}_{y,i}$ are the x and y coordinates of the virtual agent in the circular embedding, according to (7). Let us define the phase of the leading and lagging agents of i as ϕ_k, and ϕ_j, which are obtained in the same way as ϕ_i, using the respective coordinates of the agents. The phase control law driving the agent's desired angular velocity $\omega_{z,d,i}$ is given by:

$$\omega_{z,d,i} = \omega_{z,d} + k_\phi\left(\frac{1}{\phi_{ki}} + \frac{1}{\phi_{ji}}\right), \tag{11}$$

where $\omega_{z,d}$ is the nominal angular velocity shared by the swarm, $\phi_{ki} = \phi_i - \phi_k$ is the phase difference between the agent i and its leading agent k, $\phi_{ji} = \phi_i - \phi_j$ is the phase difference between agent i and its lagging agent j, and $k_\phi > 0$ is a control gain. To ensure consistent angular relationships, both ϕ_{ki} and ϕ_{ji} are constrained to $(-\pi, \pi)$.

This control law induces a repulsive behavior between neighboring agents, analogous to electrostatic repulsion between like charges: the repulsion strength increases as the agents approach each other in phase, due to the inverse dependence on ϕ_{ki} and ϕ_{ji}. Notably, ϕ_{ki} is always negative (as k leads i), while ϕ_{ji} is always positive (as j trails i). When agent i moves too close to its leader (i.e., $|\phi_{ki}| < |\phi_{ji}|$), the term $1/\phi_{ki}$ dominates, reducing $\omega_{z,d,i}$ and thus slowing the agent down. This causes ϕ_{ki} to increase and ϕ_{ji} to decrease, restoring balance. Conversely, when i is closer to its follower than its leader, the term $1/\phi_{ji}$ prevails, increasing $\omega_{z,d,i}$ and accelerating the agent to reestablish even spacing.

D. Position Control

While the phase controller ensures uniform angular separation, a separate controller is required to stabilize each agent's radial position. This is accomplished by first defining the current desired position in the circular embedding:

$$\hat{\boldsymbol{x}}_{d,i} = \begin{bmatrix} r_d\cos(\phi_i) & r_d\sin(\phi_i) & 0\end{bmatrix}^{\top}. \tag{12}$$

where r_d is the desired radius and ϕ_i is the current phase of agent i. Next, an incremental rotation about the Z-axis is constructed using the agent's desired angular velocity $\omega_{z,d,i}$, calculated with (11). The corresponding Lie algebra vector is:

$$\boldsymbol{\omega}_{z,i} = \begin{bmatrix} 0 & 0 & \omega_{z,d,i}\end{bmatrix}^{\top}. \tag{13}$$

Applying the exponential map yields the incremental rotation matrix:

$$\Delta\boldsymbol{R}_z = e^{\boldsymbol{\omega}_{z,i}^{\wedge}dt}. \tag{14}$$

which advances the agent's phase. This rotation is applied to the current desired position $\hat{\boldsymbol{x}}_{d,i}$ to form the next desired position $\hat{\boldsymbol{x}}'_{d,i}$, following

$$\hat{\boldsymbol{x}}'_{d,i} = \Delta\boldsymbol{R}_z\hat{\boldsymbol{x}}_{d,i}. \tag{15}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

The updated phase is then extracted from this position as $\phi'_i = \arctan 2(\hat{x}'_{y,i}, \hat{x}'_{x,i})$. The new phase ϕ'_i is used to obtain the angular velocity matrix $\boldsymbol{\Omega}_{xy,i}$, according to (4). Substituting $\boldsymbol{\Omega}_{xy,i}$ into (5) obtains the distortion matrix $\boldsymbol{R}_i$.

The desired updated position is obtained by applying the embedding transformation $\boldsymbol{R}_i$ in (5):

$$\boldsymbol{x}_{d,i} = \boldsymbol{R}_i \hat{\boldsymbol{x}}'_{d,i}, \tag{16}$$

where, $\boldsymbol{x}_{d,i}$ is given with respect to the center of the embedding.

To drive the agent toward this target, we employ a linear proportional-derivative (PD) controller:

$$\boldsymbol{u}_i = k_x \boldsymbol{e}_x + k_v \frac{d\boldsymbol{e}_x}{dt}, \tag{17}$$

where $\boldsymbol{e}_x = \boldsymbol{x}_{d,i} - \boldsymbol{x}_i$, and k_x and k_v are controller gains selected to ensure stability and desired dynamic performance.

Although we adopted a PD controller under double integrator dynamics for simplicity, the proposed trajectory generation framework is agnostic to the low-level controller. As demonstrated in Section VI-C, it remains effective with more complex systems, including quadcopter platforms.

IV. Stability Analysis

The stability analysis is divided into two parts: (1) stability of the embedding, and (2) stability of the transformed trajectories.

A. Stability of the Embedding

The following theorem establishes the Lyapunov stability of the embedding under the phase control law in (11).

Theorem 1: Let $n \geq 3$ be the number of agents in a swarm that rotate in a common plane around a fixed point. If each agent's angular velocity $\omega_{z,d,i}$ is governed by the control law in (11), then the system is Lyapunov stable, and all agents converge to a steady-state angular velocity $\omega_{z,d,i} = \omega_{z,d}$, and uniform phase separation (*i.e.*, $360°/n$).

Proof: Define the angular phase errors for agent i with respect to its leading (k) and lagging (j) agents as:

$$\begin{aligned} e_{ji} &= \phi_i - \phi_j = \phi_{ji}, \\ e_{ki} &= \phi_i - \phi_k = \phi_{ki}. \end{aligned} \tag{18}$$

Consider the following Lyapunov function:

$$V(e_{ji}, e_{ki}) = \sum_{i=1}^{n} \frac{1}{2} \left(\frac{1}{e_{ji}} + \frac{1}{e_{ki}} \right)^2. \tag{19}$$

Its time derivative is given by:

$$\dot{V}(e_{ji}, e_{ki}) = \sum_{i=1}^{n} \left[\left(\frac{1}{e_{ji}} + \frac{1}{e_{ki}} \right) \left(-\frac{\dot{e}_{ji}}{e_{ji}^2} - \frac{\dot{e}_{ki}}{e_{ki}^2} \right) \right]. \tag{20}$$

At equilibrium, where $e_{ji} = -e_{ki}$, we observe:

- $V(e_{ji}, -e_{ji}) = 0$ and $V(e_{ji}, e_{ki}) > 0$ elsewhere;
- $\dot{V}(e_{ji}, -e_{ji}) = 0$ and $\dot{V}(e_{ji}, e_{ki}) < 0$ elsewhere

To establish that $\dot{V}(e_{ji}, e_{ki}) < 0$ outside equilibrium, we analyze the two components of (20). From (11), we derive:

$$\frac{\omega_{z,d,i} - \omega_{z,d}}{k_\phi} = \left(\frac{1}{e_{ki}} + \frac{1}{e_{ji}} \right). \tag{21}$$

Taking the time derivative of both sides yields:

$$\frac{\dot{\omega}_{z,d,i}}{k_\phi} = \left(-\frac{\dot{e}_{ji}}{e_{ji}^2} - \frac{\dot{e}_{ki}}{e_{ki}^2} \right). \tag{22}$$

Now consider a perturbation where $|e_{ji}| > |e_{ki}|$, i.e., the agent is closer to its leader than to its follower. Since $e_{ji} > 0$ and $e_{ki} < 0$, the right-hand side of (21) is negative, causing a decrease in $\omega_{z,d,i}$ (i.e., $\omega_{z,d,i} < \omega_{z,d}$). This slows the agent down, increasing $|e_{ki}|$ and decreasing $|e_{ji}|$, eventually restoring balance (i.e., $\omega_{z,d,i} = \omega_{z,d}$). During this process, the angular acceleration $\dot{\omega}_{z,d,i} > 0$, so (22) is positive, ensuring that $\dot{V}(e_{ji}, e_{ki}) < 0$.

A symmetric argument holds for the case $|e_{ji}| < |e_{ki}|$. In all non-equilibrium configurations, the terms in (21) and (22) have opposite signs, guaranteeing that $\dot{V} < 0$.

Therefore, the control law (11) ensures Lyapunov stability.

B. Stability of the Trajectory

The next result establishes that the stability of the embedding is preserved when mapped into the 3D trajectory via rotation. This is achieved by showing that the mapping is a one-parameter homeomorphism.

Theorem 2: Let $C : \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_n \tilde{|} \tilde{\|} \hat{\boldsymbol{x}}_i \| = r \tilde{\forall} \tilde{i}, \tilde{\hat{\boldsymbol{x}}}_i \in \mathbb{R}^3, \tilde{n} > 1\}$ represent points uniformly distributed on a circle of radius r in a fixed plane. Define a family of mappings

$$f_i : \hat{\boldsymbol{x}}_i \mapsto \boldsymbol{x}_i = \boldsymbol{R}_i(\phi_i(t)) \hat{\boldsymbol{x}}_i$$

where $\boldsymbol{R}_i(\phi_i(t)) \in SO(3)$ is a time-varying rotation matrix parameterized by the phase $\phi_i(t) \in [0, 2\pi)$. If each $\boldsymbol{R}_i(\phi_i(t)) \in SO(3)$ is continuous, then the family f_i defines a one-parameter homeomorphism.

Proof: A similar result is proven in [11] using quaternion-based rotations. Since both SO(3) and the unit quaternions $\mathbb{H}^1$ are Lie groups with equivalent topological properties, the proof extends naturally to SO(3).

To confirm that f_i is a homeomorphism, we verify the following properties [18]:

- Bijectivity:
  - If $\boldsymbol{R}(\phi)\boldsymbol{x}_1 = \boldsymbol{R}(\phi)\boldsymbol{x}_2$, then $\boldsymbol{x}_1 = \boldsymbol{x}_2$
  - For every $\boldsymbol{x} \in \mathbb{R}^3$, there exists $\hat{\boldsymbol{x}} = \boldsymbol{R}^{-1}(\phi)\boldsymbol{x}$ such that $\boldsymbol{R}(\phi)\hat{\boldsymbol{x}} = \boldsymbol{x}$.
- Continuity of the forward transformation follows from the continuity of matrix multiplication.
- Continuity of the inverse transformation also holds due to the continuity of transposition and inversion in SO3.

Therefore, the mapping from the embedding to the 3D trajectory is a homeomorphism and preserves stability.

V. Experimental Methodology

The architecture described in Sections III-A and III-B was implemented in a physical swarm of five UAVs to validate the proposed approach. The experimental platform consisted

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Fig. 2. Platform used for physical experiments.

of Crazyflie quadcopters, by Bitcraze, shown in Fig. 2. Each UAV was controlled by cascaded PID controllers [19] that take position waypoints as input, which justifies the design decision for the trajectory generator to output only position commands, omitting velocity information. The flights were conducted in a $5 \times 7.5 \times 3$ m indoor test area. The overall experimental setup is illustrated in Fig. 3. The proposed algorithm, represented in the green box, was implemented in Python [1], with inter-process communication managed using ROS2. The inertial coordinate frame is fixed at the center of the room at ground level, with the $X-Y$ plane representing the horizontal plane, and the Z-axis pointing upward. Position vectors $\boldsymbol{x}$ and $\boldsymbol{x}_d$ correspond to coordinates in this frame. As shown in Fig. 3, each agent (gray box) comprises a UAV (orange box) and a ROS2 node (green area) running externally on a ground station, at 10 Hz. Global position data are provided by a Vicon Camera System (not shown in the diagram). This data is fused onboard with internal sensor readings to estimate the UAV's position $\boldsymbol{x}_i$, which is then passed to the corresponding ROS2 node. Each ROS2 node also receives phase values (ϕ_k, ϕ_j) from its leading and lagging agents, computes the desired 3D target position $\boldsymbol{x}_{d,i}$ using the control algorithm, and sends it to the UAV over a radio interface. Communication with the Crazyflie hardware is handled by a dedicated Python library.

Although the control computations were performed on an external computer for implementation convenience, the distributed nature of the proposed architecture permits deployment on embedded systems, such as the Crazyflie's STM32F405 microcontroller or another platform of choice. Each UAV was aware of its leading and lagging agents (e.g., UAV 2 recognized UAV 1 as its leader and UAV 3 as its follower). An external node, with access to the Vicon coordinates, assigned the numbers to each UAV.

VI. Results

This section provides results from both simulation and physical experiments.

A. Simulation With 50 Agents

The simulation results demonstrate the effectiveness of the proposed pipeline for a 50-agent swarm. The phase controller discussed in Section III-C ran independently for each agent and was responsible for coordinating the swarm and avoiding collisions between agents. Regardless of the number of agents in the swarm, this local controller required only the phase position of two neighbouring agents. For this reason, communication requirements did not change with an increase in the swarm's size. Therefore — as long as the trajectory is large enough to fit the agents — the approach scales well to larger swarms. For example, let us consider the case where each agent is a sphere with radius r_a such that r_a comprises the dimensions of the agent plus a buffer distance to avoid collisions. Given a circular embedding with perimeter $2\pi r_d$, the maximum number of agents that could fit in a trajectory would roughly be $n_{max} \approx 2\pi r_d / r_a$.

The agent dynamics follow the double integrator model (3), with a white Gaussian noise of $\sigma = 0.03$ added to position measurements to test robustness.

The gains used were $k_x = 6$, and $k_v = 6.5\sqrt{2}$ for the position controller (17), and $k_\phi = 0.02$ for the phase controller (11). The time step for incremental rotation updates (14) was $dt = 0.1$ s.

The radius of the circular embedding trajectory was $r_d = 10$ m, the angular velocity was $\omega_{z,d} = 1.5$ rad/s, and the height was $h = 10$ m. Note that assuming that $r_a = 0.2$ m, the maximum number of agents is $n_{max} \approx 314 > 50$. The parametric equations governing the embedding deformation in $\mathfrak{so}(3)$, defined in (4), were:

$$\begin{aligned} \omega_{x,i}(\phi_i) &= s(\cos(\phi_i)\sin(\phi_i) - \sin^3(\phi_i)), \\ \omega_{y,i}(\phi_i) &= s\,\cos^2(\phi_i)\sin(-\phi_i), \end{aligned} \tag{23}$$

where $s = 0.4$.

The agents were initialized with random positions up to 10 m away from the desired trajectory. Fig. 4 illustrates the resulting 3D trajectories, showing both the reference and the executed paths. As seen in Fig. 5, which plots the components X, Y, and Z over time, agent 1 quickly converged to the desired trajectory. This figure also shows that the deformation defined in (23) yielded z coordinates ranging from approximately 5 m to 11 m. These results confirm the controller's ability to handle large initial deviations while maintaining robustness to noise. Fig. 6 shows that the phase controller (11) maintained a target phase separation of 7.2° between agents, as expected. The system converged to this value within approximately 5 s. Oscillations of up to 0.8° amplitude were observed, representing just 11% of the target separation. These results confirm the controller's effectiveness in achieving precise phase regulation with numerous agents in the presence of noise. Considering an agent radius plus buffer distance of $r_a = 0.24$ m (consistent with the platform used in the real experiments), the minimum distance for collision avoidance was ensured by the controller, as shown in Fig. 7. Inter-agent distances ranged from 16.5 m to 0.54 m. The steady-state behavior was reached within the initial 5 s

B. Simulation Inserting One Agent

In this simulation, the swarm was initiated with three agents. A fourth agent was added after 15 s to analyze how long the phase controller would take to restore equilibrium. The trajectory parameters were $k_x = 6$, $k_v = 6.5\sqrt{2}$, $k_\phi = 3$, $\omega_{z,d} = 1$ rad/s, $r_d = 3$ m, and $h = 1.5$ m. The embedding deformation followed (4).

[1][Online]. Available: https://github.com/QUARRG/Lie_group_swarm

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

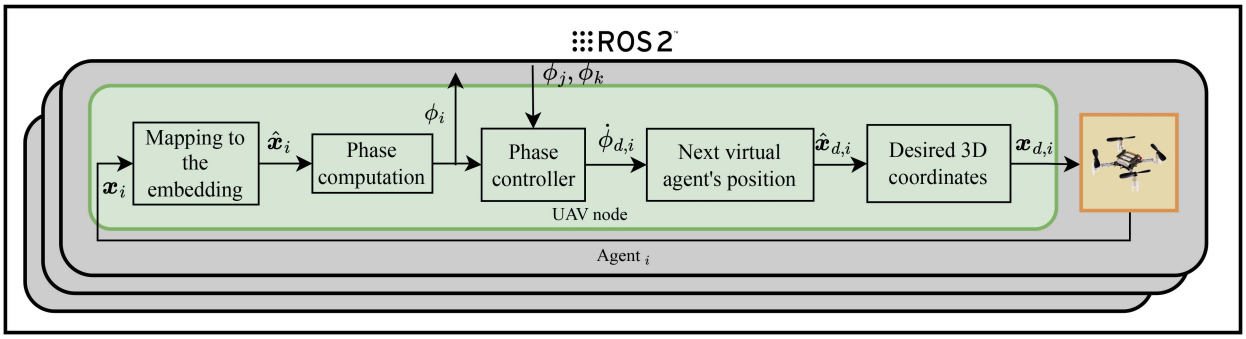


Fig. 3. Experimental setup containing the vehicle (orange box), and the control strategy implemented on each UAV (green boxes).

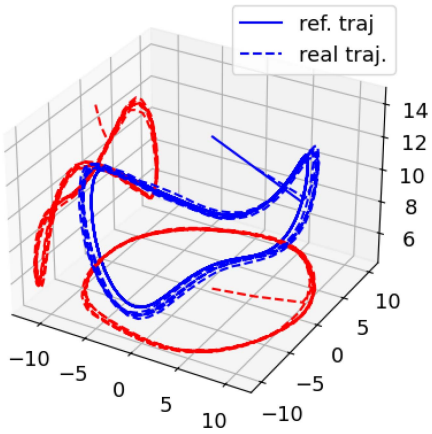


Fig. 4. 3D reference and simulated trajectories of Agent 1 following (23), with XY and YZ projections (red dotted lines).

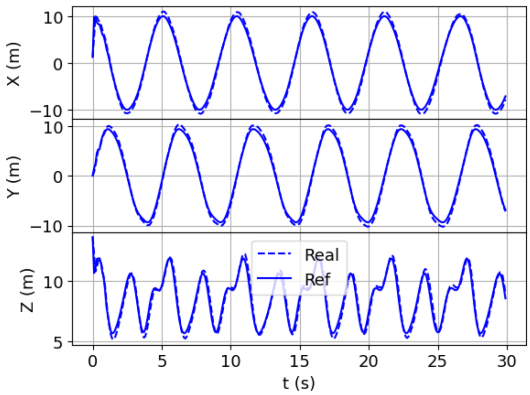


Fig. 5. Desired vs. simulated positions of Agent 1, in X, Y and Z axes, as a function of time.

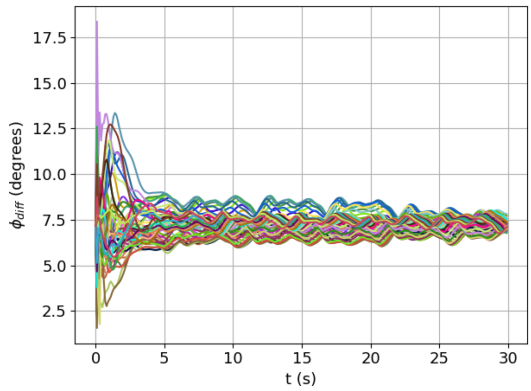


Fig. 6. Phase difference between 50 agents in simulation.

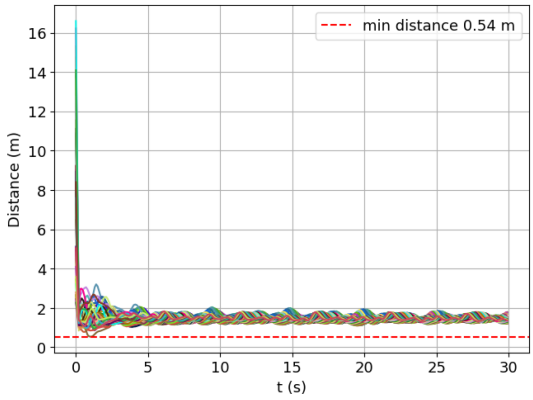


Fig. 7. Cartesian distance between 50 agents in simulation.

Fig. 8 shows the angular separations between agents. Initially, with three agents, the phase controllers took 5 s to converge to the uniform phase separation (120°). After the insertion of the fourth agent, the controllers self-adjusted within 2.5 s, thereby accommodating all agents with a new desired angular of 90°. Fig. 9 shows a minimum separation between agents of 0.8 m. Considering $r_a = 0.24$ m, as defined in Section VI-A, this means collision between agents was avoided throughout the simulation. These results demonstrate our technique automatically re-partitions the phase when an agent is added, with no centralized coordination or reparameterization required.

C. Physical Experiment

For the physical experiment that follows, the parametric equations for deformation at (23) and the parameter s were kept identical to those used in the previous simulations. This design choice was intended to demonstrate that the proposed algorithm

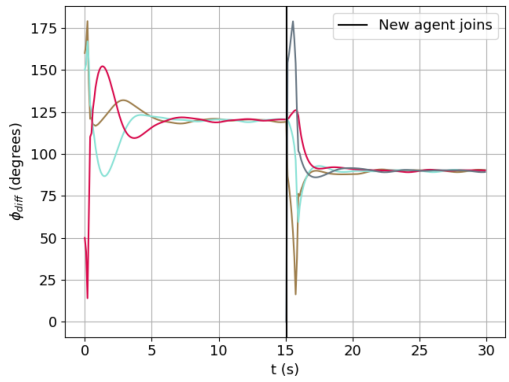


Fig. 8. Angular separations between agents in the simulation initiated with three agents and one agent added at 15 s.

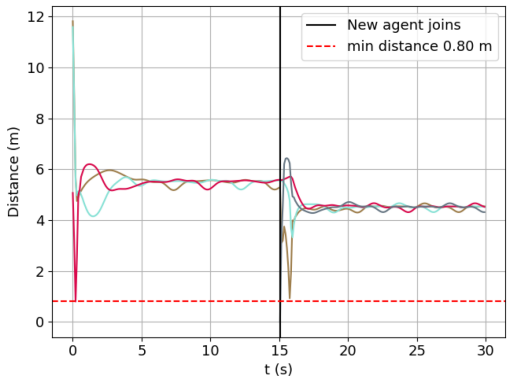


Fig. 9. Distances between agents in the simulation initiated with three agents and one agent added at 15 s.

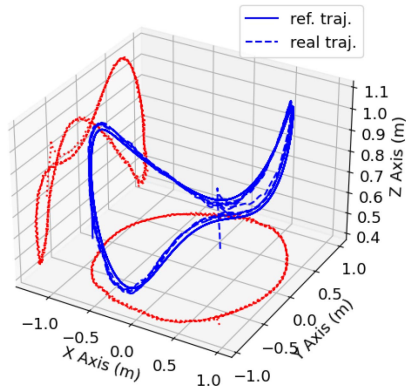


Fig. 10. Desired vs. actual 3D trajectories of Quadcopter 1 and XY and YZ projections (red dotted lines).

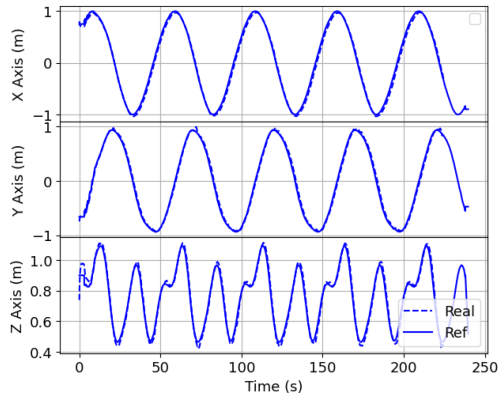


Fig. 11. Desired vs. actual positions of Quadcopter 1 in X, Y, and Z axes, as a function of time.

consistently generates the desired positions, regardless of the agent's underlying dynamics. However, the desired angular velocity was reduced to $\omega_{z,d} = 0.8$ rad/s – lower than in the simulation – for safety reasons. When operating multiple Crazyflie UAVs simultaneously, the radio communication channel can become congested, occasionally causing one or more drones to lose communication and control. Operating at lower speeds provides a safety margin, allowing time to trigger emergency stop procedures and prevent potential collisions. Although our technique is decentralized, all information exchanged between Crazyflies and the ground station is done via the same radio channel, due to Bitcraze's design choice.

The experiment was conducted with five Crazyflies units. Initially, all agents were placed at $y > 0$, with randomly-distributed x and z values between -2 m and 2 m. The phase controller gain was set to $k_\phi = 8$, the desired radius was $r_d = 1$ m, the height of the circular embedding was $h = 0.9$ m, and the time step remained $dt = 0.1$ s, as in Section VI-A. The UAVs operated under the manufacturer-provided low-level PID controller.

Fig. 10 shows the 3D trajectory of Quadcopter 1 in the same view as the simulation result in Fig. 4. The strong similarity between the two confirms that the proposed pipeline yields consistent and reliable behavior in both simulation and real-world settings, validating its applicability to physical UAV systems.

Fig. 11 displays the desired and actual trajectories for one quadcopter along the X, Y, and Z axes over time. A comparison with the simulation results in Fig. 5 reveals a strong correspondence between the simulated and real-world responses across all three dimensions, confirming the consistency and accuracy of the proposed pipeline in replicating the desired behavior on physical hardware. Although Fig. 11 resembles Fig. 5 in shape, its dimensions are smaller, as the desired radius was ten times smaller. Fig. 11 also shows that the agent followed the reference trajectory closely. Fig. 12 illustrates the effectiveness of the phase separation controller, which converged to the target angular separation of 72° in about 10 s and maintained it consistently throughout the experiment. Although with minor fluctuations about the desired value, the overall stability of the phase separation was preserved. Fig. 13 shows that the inter-agent distances achieved a minimum value of 0.6 m, and a maximum of 2.9 m in the transient state. After about 15 s, it converged and remained within 1.05 m to 1.35 m, confirming that collisions were successfully avoided, considering $r_a = 0.24$ m. Periodic behavior is noticed in Fig. 12 and Fig. 13 due to the periodic nature of the 3D shape.

The results in this section demonstrate that the pipeline proposed in this work can be applied to real-world agents, with more complex dynamics than a double-integrator.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

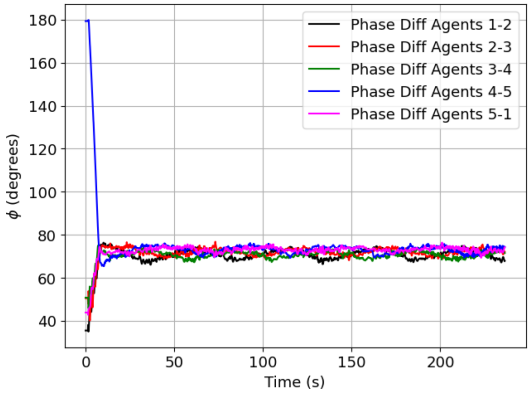


Fig. 12. Phase separation between five agents, in the real experiments.

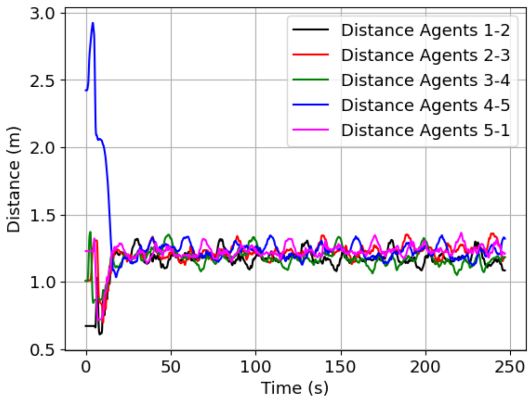


Fig. 13. Cartesian distance between five agents, in the real experiments.

VII. Conclusion

This letter introduced a novel method for generating closed 3D trajectories by deforming circular embeddings using the Lie group $\mathrm{SO}(3)$. By leveraging the relationship between $\mathrm{SO}(3)$ and its Lie algebra $\mathfrak{so}(3)$, we defined parametric equations directly in the algebra and converted them into rotation matrices, allowing for a richer class of closed curves than previously possible.

Our approach requires only position inputs, eliminating the need for velocity inputs, and reducing computational overhead. This design is well-suited for resource-constrained platforms such as Crazyflies quadcopters.

We also proposed a novel phase controller that maintains uniform angular separation (*i.e*, $360°/n$, where n is the number of agents) among agents. A formal Lyapunov stability analysis confirmed the robustness of this controller in maintaining collision-free formations.

The method was validated through simulations involving 50 agents with high angular velocity, demonstrating scalability and fast convergence even under dispersed initial conditions. Physical experiments with five quadcopters further confirmed the algorithm's effectiveness under unmodeled dynamics and with a different low-level controller architecture. Those results proved our technique to be agnostic to the agent and the low-level controller implied. As future work, we aim to implement fully decentralized agent recognition, enabling each UAV to autonomously identify its leading and lagging neighbors without relying on an external coordinator. Furthermore, the method could be extended by adding orientation control for vehicles such as fixed-wing aircraft.

Additionally, our work assumed perfect position measurements. In future work, we will relax this assumption and encode the measurement noise and uncertainties in the Lie algebra, as we mentioned in Section II, obtaining a more robust trajectory generation method.

References

[1] E. Debie, K. Kasmarik, and M. Garratt, "Swarm robotics: A survey from a multi-tasking perspective," *ACM Comput. Surveys*, vol. 56, no. 2, pp. 1–38, Sep. 2023, doi: 10.1145/3611652.

[2] P. T. Jardine and S. N. Givigi, "Flocks, mobs, and figure eights: Swarming as a lemniscatic arch," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 2, pp. 675–686, Mar.-Apr. 2023.

[3] J. Hu, B. Lennox, and F. Arvin, "Robust formation control for networked robotic systems using negative imaginary dynamics," *Automatica*, vol. 140, 2022, Art. no. 110235.

[4] A. T. Hafez, A. J. Marasco, S. N. Givigi, M. Iskandarani, S. Yousefi, and C. A. Rabbath, "Solving multi-UAV dynamic encirclement via model predictive control," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 6, pp. 2251–2265, Nov. 2015.

[5] G. Fedele, L. D'Alfonso, and A. Bono, "Emergent hypotrochoidal and epitrochoidal formations in a swarm of agents," *IEEE Trans. Autom. Control*, vol. 68, no. 12, pp. 8087–8094, Dec. 2023.

[6] A. Sinha and Y. Cao, "Nonlinear guidance law for target enclosing with arbitrary smooth shapes," *J. Guid., Control, Dyn.*, vol. 45, no. 11, pp. 2182–2192, 2022, doi: 10.2514/1.G00695710.2514/1.G006957.

[7] F. Dong, K. You, and S. Song, "Target encirclement with any smooth pattern using range-based measurements," *Automatica*, vol. 116, 2020, Art. no. 108932.

[8] T. Muslimov, "Particle swarm optimization for target encirclement by a UAV formation," *Eng. Proc.*, vol. 33, no. 1, 2023, Art. no. 15. [Online]. Available: https://www.mdpi.com/2673-4591/33/1/15

[9] L. Pichierri, G. Carnevale, and G. Notarstefano, "Distributed feedback optimization for multi-robot target encirclement and patrolling," in *Proc. IEEE 20th Int. Conf. Automat. Sci. Eng.*, 2024, pp. 1181–1186.

[10] H. Litimein, Z.-Y. Huang, and A. Hamza, "A survey on techniques in the circular formation of multi-agent systems," *Electronics*, vol. 10, no. 23, 2021, Art. no. 2959. [Online]. Available: https://www.mdpi.com/2079-9292/10/23/2959

[11] P. T. Jardine and S. Givigi, "Emergent homeomorphic curves in swarms," *Automatica*, vol. 176, 2025, Art. no. 112221. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000510982500113X

[12] 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 *Proc. 22nd Int. Conf. Methods Models Automat. Robot.*, 2017, pp. 37–42.

[13] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 1994.

[14] B. C. Hall, "An elementary introduction to groups and representations," 2000, *arXiv:math-ph/0005032*.

[15] L. Martin, *Lie Groups Latin Amer. Math. Ser.*. Berlin, Germany: Springer Int. Publishing, 2021. [Online]. Available: https://books.google.com.br/books?id=yOQfEAAAQBAJ

[16] B. C. Hall, *Quantum Theory for Mathematicians, (Graduate Texts in Mathematics Series)*. New York, NY, USA: Springer, 2013.

[17] J. Xu, P. Zhu, Y. Zhou, and W. Ren, "Distributed invariant extended Kalman filter using lie groups: Algorithm and experiments," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 6, pp. 2777–2789, Nov. 2023.

[18] J. Stillwell, *Naive Lie Theory*. New York, NY, USA: Springer Sci. Bus. Media, 2008.

[19] Bitcraze, "Controllers in the crazyflie," 2025. Accessed: Jun. 26, 2024. [Online]. Available: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/controllers/