Robot Conga: A Leader-Follower Walking Approach to Sequential Path Following in Multi-Agent Systems

Pranav Tiwari*, Soumyodipta Nath*

***Abstract*—Coordinated path following in multi-agent systems is a key challenge in robotics, with applications in automated logistics, surveillance, and collaborative exploration. Traditional formation control techniques often rely on time-parameterized trajectories and path integrals, which can result in synchronization issues and rigid behavior. In this work, we address the problem of sequential path following, where agents maintain fixed spatial separation along a common trajectory, guided by a leader under centralized control. We introduce Robot Conga, a leader-follower control strategy that updates each agent's desired state based on the leader's spatial displacement rather than time, assuming access to a global position reference—an assumption valid in indoor environments equipped with motion capture, vision-based tracking, or UWB localization systems. The algorithm was validated in simulation using both TurtleBot3 and quadruped (Laikago) robots. Results demonstrate accurate trajectory tracking, stable inter-agent spacing, and fast convergence, with all agents aligning within 250 time steps (approx 0.25 seconds) in the quadruped case, and almost instantaneously in the TurtleBot3 implementation.**

***Index Terms*—Leader-follower system, Formation Control, Multi-Agent System, Sequential Path Planning**

I. Introduction

The multi-agent systems have been a very active area of research since the past few decades. The vast interest in multi-agent distributed or centralized control is vastly driven by it's potential large-scale application in military such as unmanned arial vehicles [1], transportation-based robots [2], satellite formation flying [3], synchronization of coupled oscillations [4], microeconomics [5], power grids [6] and so on.

A key challenge in multi-agent robotics lies in achieving coordinated motion while maintaining specified spatial relationships between agents. Traditional approaches to formation control often rely on time-parameterized trajectories [7], which can suffer from synchronization issues, especially when agents experience actuation delays or environmental disturbances. These methods can lead to rigid and brittle behavior, especially in dynamic or uncertain environments.

In this work, we address the problem of sequential path following, where multiple agents are required to move along a shared trajectory while maintaining fixed spatial separations. Unlike rigid formation control, the focus here is on consistent spacing rather than preserving a fixed geometric shape. This problem arises in many real-world scenarios where smooth, orderly traversal through constrained environments is needed.

To address this, we propose Robot Conga, a centralized leader-follower control strategy in which each follower updates its desired state based on the spatial displacement of the leader along the path—rather than time. By decoupling trajectory progression from time, the method naturally avoids desynchronization issues and ensures smoother coordination, even among heterogeneous robots with differing dynamics.

This spatial coordination framework is particularly suited to indoor environments equipped with global localization infrastructure such as motion capture systems, vision-based tracking [8]–[10], inertial measurement units (IMUs) [11], or UWB localization systems [12]. Under this assumption, each agent has access to its position in a global frame, enabling precise spatial updates without requiring time synchronization or peer-to-peer communication.

Example use cases include:

- Indoor warehouse automation, where delivery bots move along narrow aisles in a train-like manner to fetch or deliver goods.
- Mobile inspection in industrial plants, where multiple sensor-equipped robots survey pipelines or factory infrastructure while avoiding congestion.
- Guided tours in museums or exhibitions, where a lead robot conducts visitors while follower robots carry assistive technology, supplies, or displays.
- Coordinated disinfection in hospitals or airports, where robots follow a prescribed cleaning route with regulated spacing to ensure full coverage.
- Search and rescue missions in indoor disaster zones, where a lead robot explores the path and followers carry resources or communication equipment.

In such applications, rigid formation control may be unnecessary or even impractical. Instead, ensuring that each robot follows the same trajectory with temporal or spatial offset becomes the central objective.

We validate the proposed Robot Conga algorithm in simulated environments using two distinct robotic platforms: the wheeled TurtleBot3 and a quadruped (Laikago) robot. Results demonstrate accurate path tracking, consistent inter-agent spacing, and rapid convergence. Compared to existing methods based on time-parameterized trajectories and path integrals [7], our approach avoids the accumulation of syn-

All the authors belong to Cyber-Physical Systems, Indian Institute of Science (IISc), Bengaluru. {pranavtiwari, soumyodiptan}@iisc.ac.in
* denotes equal contribution.

chronization errors and is better suited to heterogeneous teams and infrastructure-assisted deployment.

TABLE I
NOTATION USED IN THE PAPER

Symbol	Description
$x_1(t), x_2(t)$	Position coordinates of the robot at time t
$x_3(t)$	Orientation (heading angle) of the robot at time t
$u(t), \omega(t)$	Linear and angular velocity inputs
$x_1^*(t), x_2^*(t)$	Desired (reference) position coordinates
$x_3^*(t)$	Desired (reference) orientation
$u^*(t), \omega^*(t)$	Reference linear and angular velocity inputs
$e_1(t), e_2(t), e_3(t)$	Tracking errors in position and orientation
$\lambda_1, \lambda_2, \lambda_3$	Control gains for feedback law
$V(e,t)$	Lyapunov function used for stability analysis
$g(x)$	Parameterized path function

[a]All time-dependent variables are functions of t unless otherwise stated.

II. PRELIMINARIES

In this section, we present the foundational concepts required to understand our coordination strategy. First, we describe the kinematic model and error dynamics of wheeled mobile robots (WMRs). Next, we briefly introduce Lyapunov functions and their relevance in ensuring system stability.

A. Unicycle Dynamics

We consider a wheeled mobile robot (WMR) modeled as a unicycle system, which is subject to non-holonomic constraints. The state-space representation is given by:

$$\begin{aligned} \dot{x_1}(t) &= u(t)\cos[x_3(t)] \\ \dot{x_2}(t) &= u(t)\sin[x_3(t)] \\ \dot{x_3}(t) &= \omega(t) \end{aligned} \tag{1}$$

where $(x_1(t), x_2(t))$ denote the robot's position, $x_3(t)$ is the orientation, and $u(t), \omega(t)$ are the linear and angular velocity inputs, respectively.

We define a virtual robot (reference agent) whose desired trajectory and control inputs are given by:

$$\begin{aligned} \dot{x_1}^*(t) &= u^*(t)\cos[x_3^*(t)] \\ \dot{x_2}^*(t) &= u^*(t)\sin[x_3^*(t)] \\ \dot{x_3}^*(t) &= \omega^*(t) \end{aligned} \tag{2}$$

The control objective is to design inputs $u(t)$ and $\omega(t)$ such that the real robot's trajectory converges asymptotically to that of the virtual robot, i.e.,

$$\{x_1(t), x_2(t), x_3(t)\} \to \{x_1^*(t), x_2^*(t), x_3^*(t)\} \quad \text{as} \quad t \to \infty$$

This convergence ensures that each agent accurately tracks its assigned trajectory segment as determined by the centralized coordination scheme.

B. Error Dynamics and Control Law

To ensure the real robot tracks the desired trajectory, we define the tracking error between the actual and reference states as:

$$\mathbf{e}(t) = \begin{bmatrix} e_1(t) \\ e_2(t) \\ e_3(t) \end{bmatrix} = \begin{bmatrix} x_1(t) - x_1^*(t) \\ x_2(t) - x_2^*(t) \\ x_3(t) - x_3^*(t) \end{bmatrix} \tag{3}$$

Taking the time derivative of the error vector, we obtain the error dynamics:

$$\begin{bmatrix} \dot{e}_1(t) \\ \dot{e}_2(t) \\ \dot{e}_3(t) \end{bmatrix} = \begin{bmatrix} u(t)\cos[x_3(t)] - u^*(t)\cos[x_3^*(t)] \\ u(t)\sin[x_3(t)] - u^*(t)\sin[x_3^*(t)] \\ \omega(t) - \omega^*(t) \end{bmatrix} \tag{4}$$

C. Lyapunov Functions and Asymptotic Stability

Lyapunov theory provides a powerful tool to assess the stability of dynamical systems without requiring the explicit solution of differential equations. The fundamental idea is to construct a scalar function, called a *Lyapunov function*, that acts like an energy measure of the system.

Definition: Consider a nonlinear system of the form:

$$\dot{x} = f(x), \quad x \in \mathbb{R}^n, \quad f(0) = 0 \tag{5}$$

where the origin is an equilibrium point. A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is said to be a Lyapunov function if it satisfies the following properties in a neighborhood $\mathcal{D}$ of the origin:

- $V(0) = 0$ and $V(x) > 0$ for all $x \in \mathcal{D} \setminus \{0\}$ (positive definite),
- $\dot{V}(x) = \frac{dV}{dt} = \nabla V(x)^T f(x) \leq 0$ for all $x \in \mathcal{D}$ (negative semi-definite).

Stability Implications:

- If such a Lyapunov function exists, the equilibrium at the origin is *Lyapunov stable*.
- If $\dot{V}(x) < 0$ for all $x \neq 0$, the equilibrium is *asymptotically stable*.
- If $V(x) \to \infty$ as $\|x\| \to \infty$ (i.e., V is radially unbounded), and $\dot{V}(x) < 0$ globally, then the system is *globally asymptotically stable*.

Why Lyapunov Functions Matter: Lyapunov-based methods are widely used in robotics and control because they allow for the design and verification of controllers that guarantee stability, even in the presence of nonlinearities. Instead of solving the system's equations of motion, we analyze the behavior of a carefully constructed scalar function whose decrease over time implies convergence to a desired state.

This framework will be used in later sections to verify the stability of our proposed control strategy.

III. STABILITY OF THE SYSTEM

To analyze the stability of the closed-loop system, we adopt a Lyapunov-based approach similar to the one proposed in [13].

$$V(e) = \frac{1}{2} e^T \begin{bmatrix} \delta & 0 & 0 \\ 0 & \delta_1 & 1 \\ 0 & 1 & 1 \end{bmatrix} e \tag{6}$$

where $\delta > 0$, $\delta_1 > 0$ are constants, and $e = [e_1, e_2, e_3]^T$ is the error vector defined earlier.

A. Time Derivative of the Lyapunov Function

The time derivative of $V(e)$ along the trajectories of the system is:

$$\begin{aligned} \dot{V}(e,t) &= \delta e_1 \dot{e}_1 + \delta_1 e_2 \dot{e}_2 + \dot{e}_2 e_3 + e_2 \dot{e}_3 + e_3 \dot{e}_3 \\ &= -\delta \lambda_3 e_1^2 - \delta_1 \sigma e_1 e_2 - \sigma e_1 e_3 \\ &\quad + \delta_1 \psi e_2 \sin(e_3) + \psi e_3 \sin(e_3) \\ &\quad - a e_2^2 - b e_3^2 - a e_2 e_3 - b e_2 e_3 \end{aligned} \tag{7}$$

where

$$\sigma(e_3, t) = \lambda_3 \tan(e_3 + x_3^*(t)), \quad \psi(e_3, t) = \frac{u^*(t)}{\cos(e_3 + x_3^*(t))}.$$

B. Control Law

We employ control approach similar to the one proposed in [13] :

$$\begin{aligned} u(t) &= \frac{u^*(t)\cos[x_3^*(t)] - \lambda_3 e_1}{\cos(x_3^*(t) + e_3)}, \\ \omega(t) &= \omega^*(t) - \lambda_2 e_3 - \lambda_1 e_2, \end{aligned} \tag{8}$$

where $\lambda_1, \lambda_2, \lambda_3 > 0$ are control gains.

Resulting Error Dynamics

Substituting the control law into the system yields the following closed-loop error dynamics:

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} -\lambda_3 e_1 \\ -\lambda_3 e_1 \tan(x_3^*) + \dfrac{u^* \sin(e_3)}{\cos(e_3 + x_3^*)} \\ -\lambda_1 e_2 - \lambda_2 e_3 \end{bmatrix} \tag{9}$$

Exponential Stability

As shown in [13], for any $\lambda_3 > 0$, one can choose positive constants $\delta, \delta_1, \lambda_1, \lambda_2$ such that the Lyapunov function $V(e)$ is positive definite and its derivative $\dot{V}(e)$ is negative definite in a neighborhood $D \subset \mathbb{R}^3$, i.e.,

$$\dot{V}(e) < 0 \quad \forall e \in D.$$

Moreover, there exist positive constants α, β, ρ such that:

$$\alpha \|e\|^2 \leq V(e) \leq \beta \|e\|^2, \quad \dot{V}(e) \leq -\rho \|e\|^2, \tag{10}$$

implying that the origin is an exponentially stable equilibrium point of the system in D.

Thus, the proposed control law guarantees that the real robot's trajectory converges to the virtual reference trajectory, ensuring asymptotic stability of the system.

IV. Trajectory Propagation and Real-Time Adaptation in Leader-Follower Systems

In our proposed framework, the leader's movement along the path determines the reference positions and velocities for all follower robots. By using arc-length-based updates instead of time-parameterized trajectories, we ensure that inter-bot distances remain consistent and synchronization issues are avoided. The approach combines spatial propagation of reference states and real-time path updates through user interaction. This dual-layered strategy ensures scalability, adaptability, and smooth navigation in dynamically changing environments.

A. Spatial Propagation of Reference States

Whenever the leader robot advances a small displacement (δs) along the path, the desired states $(x_{i1}^*(t), x_{i2}^*(t), x_{i3}^*(t))$ of each follower robot are updated proportionally to maintain equal spacing along the curve, where i denotes the index of the robot in the formation (with $i = 1$ being the leader and $i > 1$ referring to followers) as showncased in Figure 1.

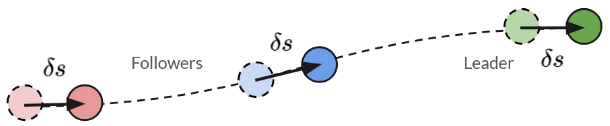

Fig. 1. Displacement-based State Update

Assuming that the desired linear velocity of all bots along the trajectory is equal and is controlled in real-time via joystick input.

$$u_i^* = \mathbb{V}_{cmd}$$

We discretize the trajectory propagation for real-time implementation by updating the desired states of each follower at discrete time steps. Let **k** denote the discrete time step, and δt be the time interval between updates. The updated desired states at each time step are given by:

$$\begin{aligned} \dot{x}_{i1}^* = \dot{u}_i^* cos(x_{i3}^*) \Rightarrow x_{i1}^*[k] &= x_{i1}^*[k-1] + \dot{u}_i \times cos(x_{i3}^*[k-1]) \times \delta t \\ x_{i2}^*[k] &= g(x_{i1}^*[k]) \\ x_{i3}^*[k] &= \tan^{-1}\left(\frac{dg(x_1)}{dx_1}\right)\Bigg|_{x_1 = x_{i1}^*[k]} \end{aligned} \tag{11}$$

The desired angular velocities are determined on the basis of the instantaneous radius of curvature (IROC) of the path at each bot's location:

$$\omega_i^* = \frac{u_i^*}{\text{IROC}} = \left| \frac{\mathbb{V}_{cmd} \cdot \frac{d^2 g(x_1)}{dx_1^2}}{\left[1 + \left(\frac{dg(x_1)}{dx_1}\right)^2\right]^{3/2}} \right|_{x_1 = x_{i1}^*[k]} \tag{12}$$

Initial Configuration

At $\mathbf{k} = 0$, the follower robots are initialized at fixed arc-length intervals behind the leader along the curve. This initialization ensures uniform propagation as the leader moves as shown in Figure 2.

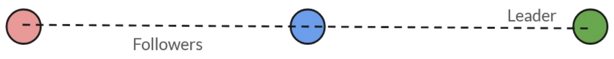

Fig. 2. Leader-Follower Initialization

B. Dynamic Path Adaptation Using Joystick Steering

To further enhance responsiveness, we incorporate joystick or keyboard-based steering to dynamically adjust the leader robot's heading. This allows the formation to adapt dynamically to new goals or environmental constraints.

Whenever the leader's heading is modified, the original path is no longer valid beyond the current point. Therefore, we recalculate a new reference trajectory $g^*(x)$ that smoothly transitions from the robot's current position and heading to the intended direction (Figure 3).

To generate this updated trajectory, we interpolate a set of virtual waypoints based on the current pose and the desired steering input. The result is a spatial path that serves as a new reference for both the leader and the follower robots

To do so we interpolate the bot locations

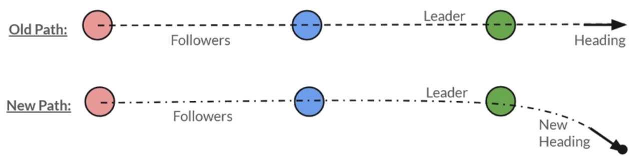

Fig. 3. Key-Controlled Dynamic Trajectory

Several interpolation methods can be used to construct the smooth trajectory from the updated waypoints:

- **Barycentric Interpolation** [14]: While computationally efficient, this method is highly sensitive to changes in orientation and can result in undesirable global distortions or sharp turns, especially under dynamic user inputs.
- **B-Spline Interpolation** [15]: This piecewise polynomial method offers local control, ensuring smooth transitions in curvature. It performs robustly under real-time heading changes, producing natural and collision-free trajectories.

Based on comparative performance (Fig 4), B-spline interpolation was selected as the default approach for generating dynamically updated reference trajectories in real time. Its smooth curvature and local adaptability make it ideal for human-in-the-loop navigation scenarios.

C. Trajectory Correction on Steep Ascent

Since the angular speed (ω) depended on the slope of the trajectory, the measurement would become unstable as the

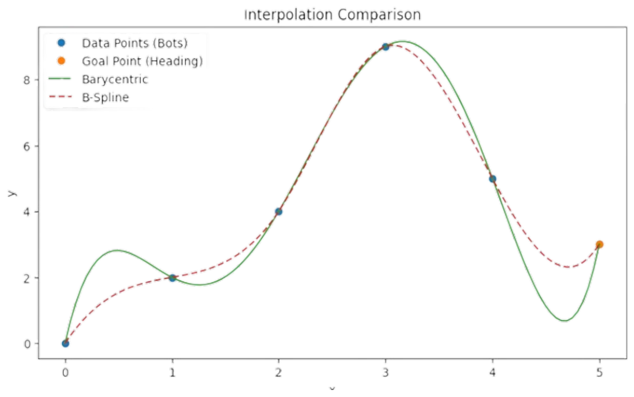

Fig. 4. Interpolation Method Comparison

trajectory aligned with the Y-axis—that is, when the slope approached ∞. To mitigate this issue, a mechanism was implemented to allow a frame change whenever the heading angle began aligning with the Y-axis. This caused the local frame to rotate, updating the coordinates in the local frame and ensuring that the trajectory would never become parallel to the Y-axis at any point.

V. Results

To evaluate the performance, robustness, and versatility of the proposed *Robot Conga* algorithm, we conducted experiments across a diverse set of robotic platforms: wheeled robots (TurtleBot3), legged robots (Laikago Quadrupeds), and heterogeneous combinations of both. The architecture abstracts away low-level actuation differences and requires tuning only high-level parameters $\lambda_1, \lambda_2, \lambda_3$ for stable convergence and consistent formation behavior.

Table II summarizes the values of these control parameters for each tested configuration.

TABLE II
Control Parameters Used for Different Robot Types

Robot Type	$\boldsymbol{\lambda_1}$	$\boldsymbol{\lambda_2}$	$\boldsymbol{\lambda_3}$
TurtleBot3	4.5	7.5	2.5
Laikago Quadruped	4.5	1.5	2.5
Heterogeneous (Mixed)	5.0	1.0	1.5

A. TurtleBot3 Implementation

The algorithm was implemented in ROS2 [16] using multiple TurtleBot3 robots[1] within the Gazebo simulation environment [17], where each robot was spawned under a unique namespace. A keyboard-based interface controlled the heading and speed of the leader in real time, while follower bots adapted using arc-length-based spatial propagation.

The system's performance was highly sensitive to the tuning of high-level control parameters $\lambda_1, \lambda_2, \lambda_3$, commanded linear velocity $\mathbb{V}_{cmd}$, and heading adjustment per keystroke. Each

[1]https://wiki.ros.org/turtlebot3

keypress served as a discrete perturbation to the leader's trajectory, introducing a transient increase in error across the formation. However, due to the convergence properties of the control law, both positional and angular tracking errors consistently diminished over time.

Notably:

- **Linear velocity convergence**: Follower bots reached the commanded linear velocity almost instantaneously.
- **Angular convergence**: Angular deviations were corrected within a few milliseconds post-perturbation.
- **Asymptotic stability**: Tracking errors reduced smoothly and converged nearly to zero as shown in Figure 5.

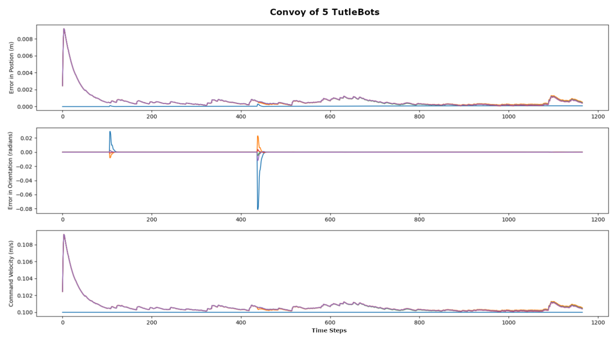

Fig. 5. Tracking Performance on TurtleBot3: Positional and angular errors converge despite dynamic path changes. [Multiple spikes represent individual direction and velocity commands]

Figure 5 demonstrates this stability, even under frequent trajectory modifications introduced by user input. A video demonstration is available online[2].

B. Laikago Quadruped Implementation

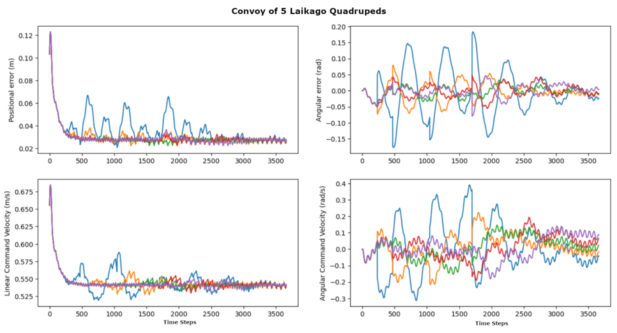

Fig. 6. Tracking Performance on Laikago Quadrupeds: Rapid damping of trajectory error following steering input. [Multiple spikes represent individual direction and velocity commands]

To validate the generalizability of our approach, we implemented the algorithm on Laikago quadruped robots in the PyBullet simulation environment [18]. These robots use a low-level Model Predictive Control (MPC) system[3] , while our high-level framework provided reference updates based on the leader's motion.

Despite the platform's drastically different dynamics, the system demonstrated rapid error convergence. As shown in Figure 6, each joystick-induced heading change caused a spike in error, which stabilized within 250 simulation steps (approx 0.25 seconds). This response underscores the algorithm's robustness under sudden direction shifts and its compatibility with legged locomotion systems. A demonstration video is also available[4].

C. Heterogeneous Multi-Robot Implementation

The final experiment evaluated the algorithm's performance in a mixed-robot setting combining TurtleBot3 (wheeled) and Laikago (legged) platforms (Figure 8). Despite their differing locomotion and control architectures, both platforms successfully maintained consistent tracking behavior under the same high-level propagation framework.

Figure 7 shows the evolution of tracking error over time for both robots. Error peaks correspond to steering commands, while rapid convergence back to baseline demonstrates the stability and coordination effectiveness of the proposed method.

This result highlights the platform-agnostic nature of the control architecture: although the physical actuation mechanisms differ, both robots interpret the propagated reference consistently. A video of this experiment is available[5].

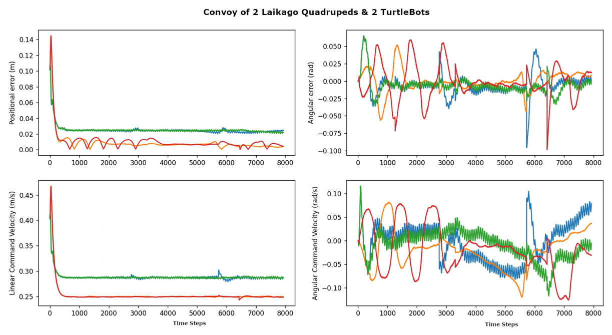

Fig. 7. Tracking Error in Mixed-Platform Setup: Both TurtleBot3 and Laikago robots exhibit stable convergence following dynamic path updates. [Multiple spikes represent individual direction and velocity commands]

VI. Conclusion

In this work, we introduced a spatial propagation-based control framework for trajectory tracking in multi-robot systems, termed the *Robot Conga* algorithm. By decoupling trajectory propagation from time and instead anchoring it to the leader's displacement, we maintained fixed inter-bot spacing and achieved synchronization across heterogeneous platforms. The proposed method was demonstrated to be adaptable, robust, and scalable across both wheeled and legged robots, with minimal tuning required per platform.

[2]TurtleBot3 experiment video: https://www.youtube.com/watch?v=b6I9cNeFR_4

[3]https://github.com/Farama-Foundation/a2perf-quadruped-locomotion

[4]Laikago experiment video: https://www.youtube.com/watch?v=foqDSqTVpeE

[5]Heterogeneous robot experiment video: https://www.youtube.com/watch?v=A-Nygq5zwCc

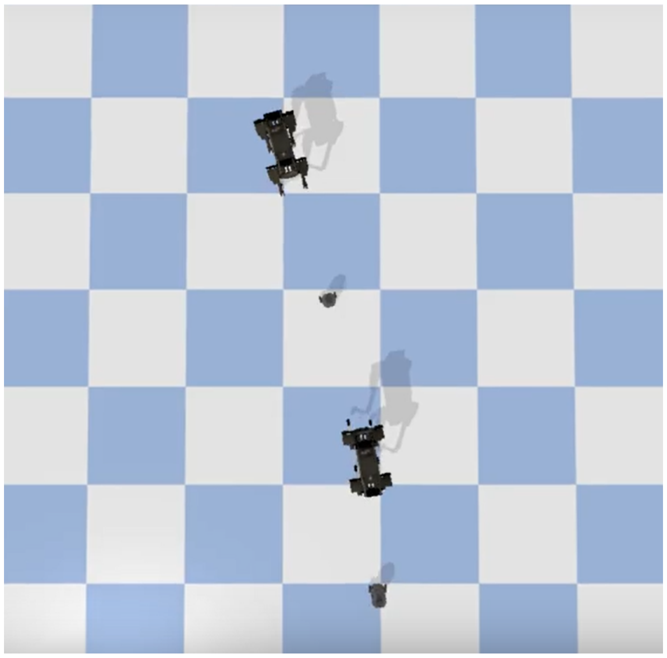

Fig. 8. TurtleBot3 and Laikago robots exhibit stable convergence following dynamic path updates.

Our results confirmed that the system achieves asymptotic stability even under frequent heading changes and external disturbances such as joystick inputs. Through careful selection of interpolation methods—specifically B-splines—we ensured smooth and dynamic trajectory generation in real time.

The algorithm's abstraction over low-level control layers and its compatibility with both simulated and real-world robotic systems make it a promising candidate for formation control tasks in complex environments.

VII. Future Work

Several directions remain open for extending the current work. One potential avenue is adapting the proposed algorithm for deployment in GPS-denied environments, using onboard perception and SLAM-based localization to estimate the leader's trajectory and propagate it throughout the formation.

Another promising direction involves incorporating advanced control strategies to address communication and sensing delays. Integrating Observer-Based Feedback Protocols, as discussed in [19], could enhance robustness in networked robotic systems where latency is a critical factor.

Further exploration into Consensus Control frameworks may help achieve prescribed performance guarantees in leader-follower multi-agent systems, as outlined in [20].

Lastly, to improve reliability in real-world deployments, it is valuable to investigate resilient formation control under partial system failures, such as leader dropout. This aligns with ongoing work in leader-failure resilience, including studies such as [21].

References

[1] Y. Zheng, C. Zheng, X. Zhang, F. Chen, Z. Chen, and S. Zhao, "Detection, localization, and tracking of multiple mavs with panoramic stereo camera networks," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1226–1243, 2023.

[2] Y. Yang, Y. Xiao, and T. Li, "A survey of autonomous underwater vehicle formation: Performance, formation control, and communication capability," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 815–841, 2021.

[3] L. Perea and P. Elosegui, "Extension of the cucker–smale control law to space flight formations," *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, vol. 32, pp. 527–537, 03 2009.

[4] H. Yin, P. G. Mehta, S. P. Meyn, and U. V. Shanbhag, "Synchronization of coupled oscillators is a game," *IEEE Transactions on Automatic Control*, vol. 57, no. 4, pp. 920–935, 2012.

[5] V. E. Lambson, "Self-enforcing collusion in large dynamic markets," *Journal of Economic Theory*, vol. 34, no. 2, pp. 282–291, 1984.

[6] Z. Yan and Y. Xu, "A multi-agent deep reinforcement learning method for cooperative load frequency control of a multi-area power system," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4599–4608, 2020.

[7] A. Ailon and I. Zohar, "Control strategies for driving a group of nonholonomic kinematic mobile robots in formation along a time-parameterized path," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 2, pp. 326–336, 2012.

[8] Y. Alkendi, L. Seneviratne, and Y. Zweiri, "State of the art in vision-based localization techniques for autonomous navigation systems," *IEEE Access*, vol. 9, pp. 76847–76874, 2021.

[9] A. R. Vetrella, G. Fasano, D. Accardo, and A. Moccia, "Differential gnss and vision-based tracking to improve navigation performance in cooperative multi-uav systems," *Sensors*, vol. 16, no. 12, 2016.

[10] M. Pachter, N. Ceccarelli, and P. Chandler, "Vision-based target geo-location using feature tracking," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6863, 2007.

[11] A. Poulose, O. S. Eyobu, and D. S. Han, "An indoor position-estimation algorithm using smartphone imu sensor data," *IEEE Access*, vol. 7, pp. 11165–11177, 2019.

[12] L. Yao, Y.-W. A. Wu, L. Yao, and Z. Z. Liao, "An integrated imu and uwb sensor based indoor positioning system," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, 2017.

[13] A. Ailon and I. Zohar, "Motion planning and optimal control in a kinematic model of an automobile," *IFAC Proceedings Volumes*, vol. 40, no. 15, pp. 499–504, 2007. 6th IFAC Symposium on Intelligent Autonomous Vehicles.

[14] J.-P. Berrut and L. N. Trefethen, "Barycentric lagrange interpolation," *SIAM Review*, vol. 46, no. 3, pp. 501–517, 2004.

[15] A. Chaudhuri, "B-splines," *arXiv preprint arXiv:2108.06617*, 2021.

[16] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[17] Open Source Robotics Foundation, "Gazebo Simulator." https://gazebosim.org, 2024. Accessed: 2025-07-12.

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[19] Y. Yang and Y. He, "Time-varying formation-containment control for heterogeneous multi-agent systems with communication and output delays via observer-based feedback protocol," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 15435–15448, 2025.

[20] F. Chen and D. V. Dimarogonas, "Consensus control for leader-follower multi-agent systems under prescribed performance guarantees," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 4785–4790, 2019.

[21] T. Murakami and T. Namerikawa, "Resilient formation control in multi-agent systems considering leader failure," *SICE Journal of Control, Measurement, and System Integration*, vol. 18, no. 1, p. 2510766, 2025.