Cyclic Pursuit in Single & Multi-Agent Crazyflies

Hanson Huang, Guoxi Wu

Multi-Agent Robotics Laboratory (MURO), UC San Diego

UC San Diego JACOBS SCHOOL OF ENGINEERING

GEAR Guided Engineering Apprenticeship in Research

OBJECTIVES

- Our goal is to develop and implement cyclic pursuit algorithms for single and multiple drone systems, allowing Crazyflies drones to autonomously maintain geometric formations.
- Cyclic pursuit algorithms allow a group of robotic agents to autonomously follow each other in a closed trajectory under constant speed, forming geometric patterns and maintaining relative distances.

Figure 1: Image of Crazyflie drone

PROGRAM/PACKAGE

- Initially, we will develop our algorithms in a ROS 2 simulation environment, allowing for safe, rapid experimentation without risking physical hardware.
- Crazyswarm2 – A package based on ROS 2 to allow for controlling the Crazyflie drone. It also provides Motion Capture integration, Broadcast, and Simulation functionality.
- Following successful simulation tests, we will conduct physical experiments within the aerodrome equipped with Vicon motion tracking cameras.
- This setup enables precise motion tracking of the drones, significantly enhancing the accuracy and reliability of our experimental data.

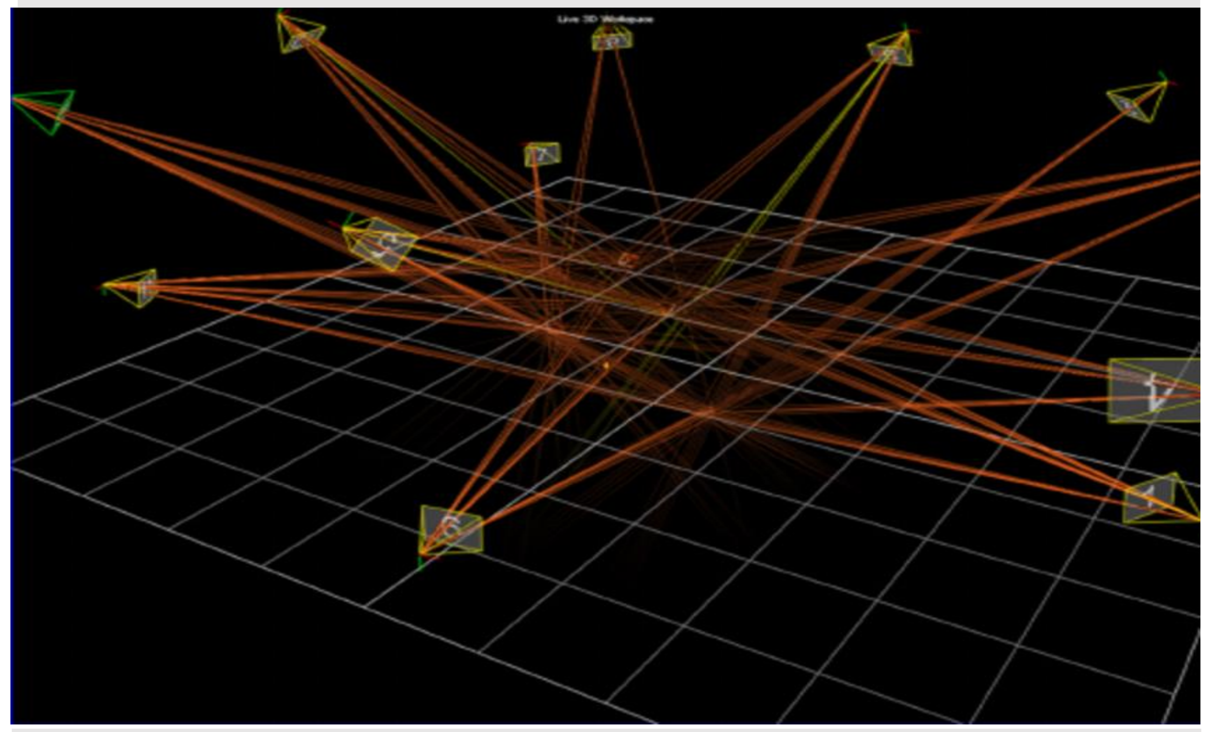

Figure 2: Image of Motion Capture Interface

ROS 2™

METHODOLOGY

To implement cyclic pursuit, the Crazyflie drones will need to receive motion capture data.

- This requires a subscriber node to receive the position and motion data from the motion capture system (Vicon/OptiTrack) and a publisher node to send the processed data to the Crazyflie drones.
- The subscriber node will subscribe to position/motion and IMU data from the motion capture system, capturing the real-time locations and orientations of the drones.

This data will then be modified using our cyclic pursuit algorithm. The publisher node will send velocity commands to the Crazyflie drones, based on this modified data. These commands include the velocities in the x and y directions, the yaw rate, and the z-distance of each drone.

Under the cyclic pursuit algorithm, the speed of each Crazyflie drone remains constant. The velocities in the x and y directions are calculated based on the angle of each drone relative to its predecessor, determined from the Vicon and IMU data obtained from the motion capture system, lead to equation:

$$\begin{bmatrix}\dot{x}_i\\ \dot{y}_i\\ \dot{\theta}_i\end{bmatrix} = \begin{bmatrix} cos\theta_i & 0\\ sin\theta_i & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix} v_i\\ \omega_i\end{bmatrix} \quad (1)$$

The mathematical expression for relative positioning of multiple robots is:

$$\dot{r}_i = -s[cos\alpha_i + \cos(\alpha_i + \beta_i)]$$

$$\dot{\alpha}_i = \frac{s}{r_i}[sin\alpha_i + \sin(\alpha_i + \beta_i)] - k\alpha_i \quad (2)$$

$$\dot{\beta}_i = k(\alpha_i - \alpha_{i+1})$$

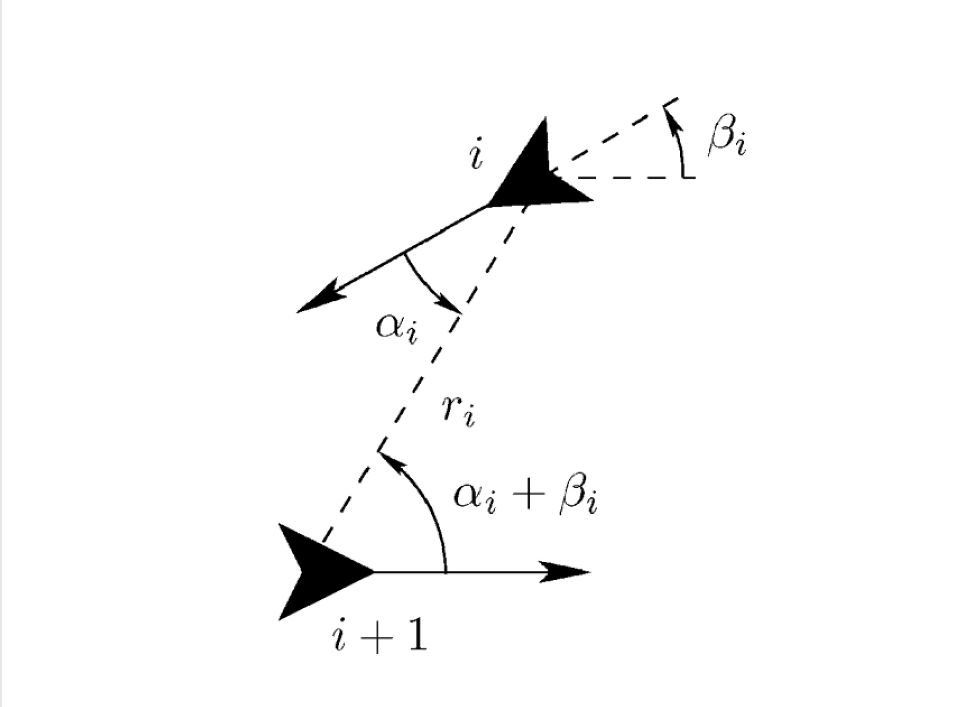

Figure 3: Coordinate Visualization [1]

This kinematic equation was in in polar coordinate. Which the (r, α, β) corresponds to (r, θ, φ). While s and k are some positive constant.

1. During the code implementation, we are primarily working with the built function in the ROS 2 and Crazyswarm2 packages.
2. Our cyclic pursuit uses the cmd_hover functionality which is based on the CFlib backend in the crazyswarm2 package.
3. However, since the backend of cmd_hover is CFlib, it does not support broadcast functionality that is needed in the crazyflie communication. We need to make adjustments to the CFlib backend file.
4. After adjustments to the backend are made, we will use the cmd_hover functionality which requires input of x and y velocities, z-distance, and yaw rate that was obtained through our cyclic pursuit algorithms.

RESULTS

- Through extensive simulations, we improved the algorithm to ensure stability and accuracy in maintaining geometric formations.
- By using the Vicon motion tracking system in our aerodrome, we were able to achieve precise position and orientation data for the drones, which was crucial for refining the pursuit dynamics.
- The algorithm enabled each drone to autonomously follow its predecessor while maintaining a consistent distance, resulting in smooth and coordinated circular movements.

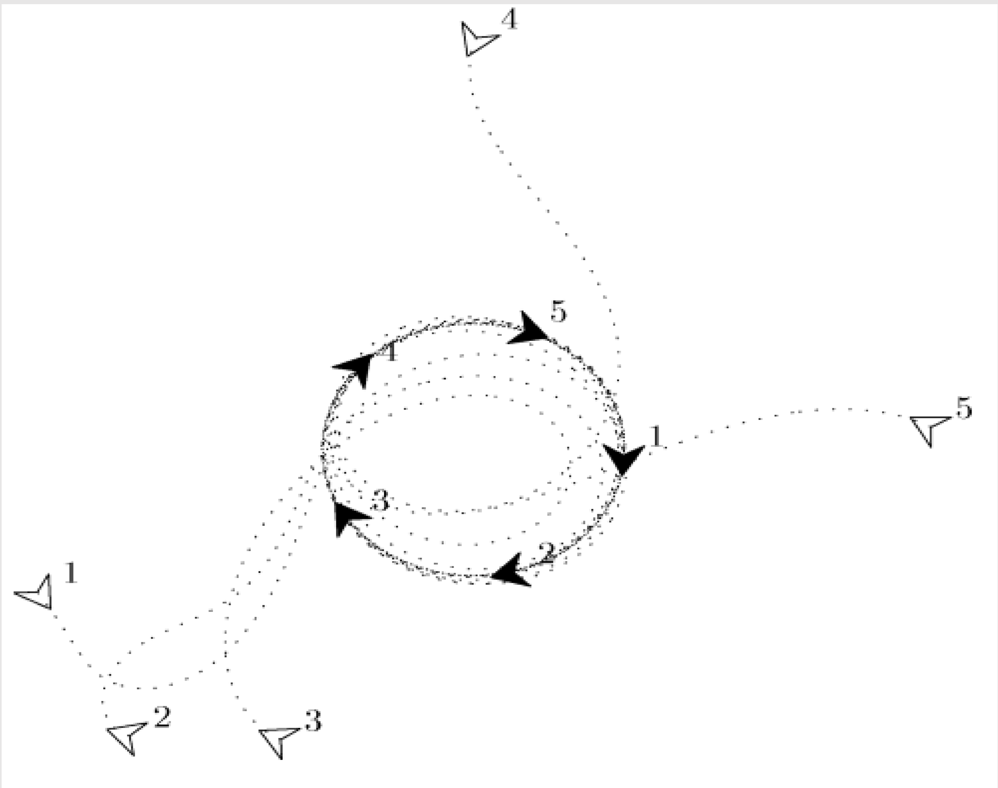

Figure 4: Cyclic Pursuit graph [1]

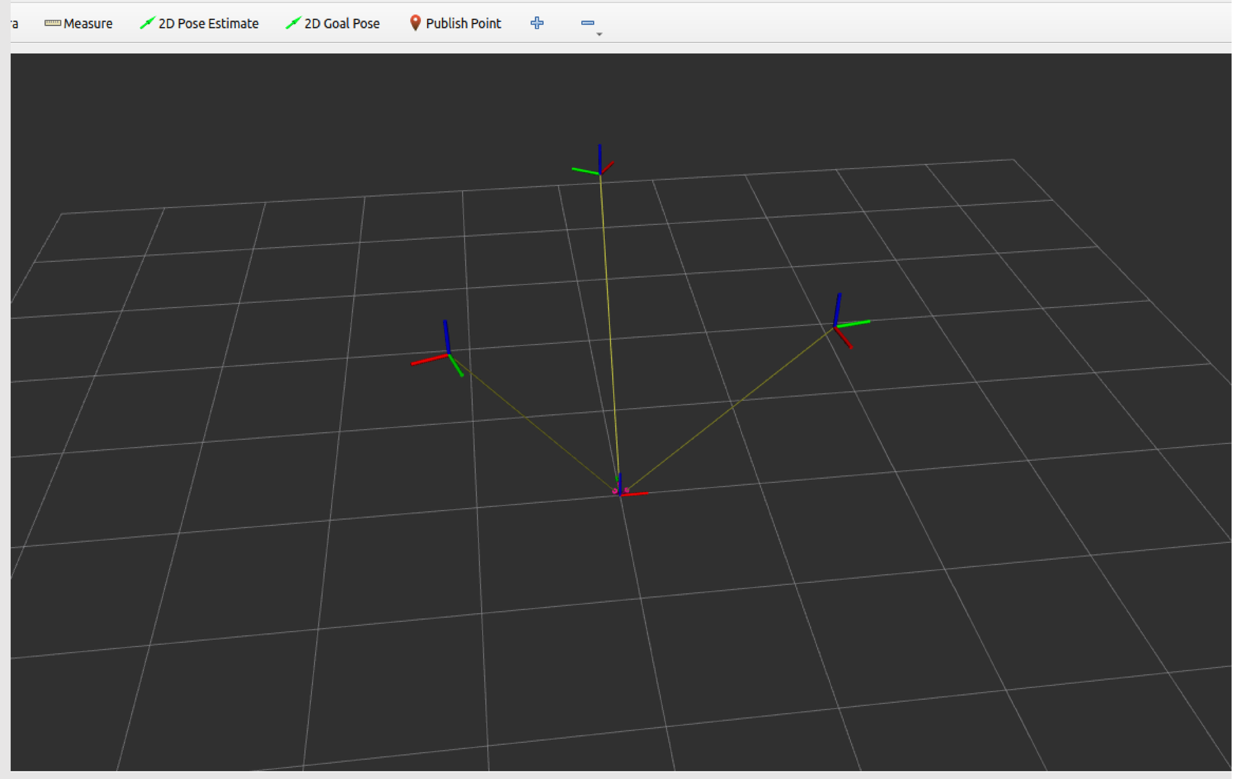

Figure 5: Cyclic Pursuit in RViz

- In the simulation setting, our algorithm consistently performed well, showing robust and reliable behavior under various conditions.
- However, while these initial results are promising, further testing is required in real-world physical settings to validate the algorithm's performance in less controlled environments.

REFERENCES

[1] J. A. Marshall, M. E. Broucke and B. A. Francis, "Formations of vehicles in cyclic pursuit," in IEEE Transactions on Automatic Control, vol. 49, no. 11, pp. 1963-1974, Nov. 2004, doi: 10.1109/TAC.2004.837589.

CONCLUSIONS & FUTURE WORK

- Despite our success with the cyclic pursuit algorithm, our progress in developing the Q-learning algorithm was limited due to time restraints and technical difficulties with the aerodrome.
- As a result, we were unable to fully explore the potential of Q-learning for autonomous navigation and cooperative behaviors in multi-agent systems.
- Future work will focus on resolving these technical issues, enhancing the reliability of our motion capture system, and further developing the Q-learning algorithm to enable Crazyflie drones to learn and adapt to their environment through experience.

ACKNOWLEDGEMENTS

We would like to thank Scott and Mihir for guiding us through the research. Dr. Cortés , Dr. Martínez, MURO lab, and GEAR program for the opportunity to be part of this research. We also want to thank Dr. Katya and Sergio Godinez for helping us to prepare for the lab. Also, we want to acknowledge the Aerodrome and OptiTrack manager for allowing us to use the motion capture system.