Project LegionAir

1st Darshit Miteshkumar Desai
University of Maryland
College Park
darshit@umd.edu

2nd Patrik Dominik Pördi
University of Maryland
College Park
ppordi@umd.edu

***Abstract*—This document summarizes the project proposal for the Swarm Robotics course Research project. The project aims to emulate a swarm formation of a roman legion where if a single comrade falls out of formation there is a readjustment of the whole formation such that the coverage can be maximized. The project's outcomes would be the application of our knowledge of drone hardware, perception, control systems, software architectures and swarm algorithms learnt during our coursework of the Robotics masters degree.**

Please find the code: Github Link

Please find the videos: Drive Videos Link

***Index Terms*—Formation control, Coverage control, Shape formation, Aerial robotics, UAV, PX4, ROS2**

I. Introduction

Within our project, we aim to investigate the collaboration among a multi-agent system employing drones. Our objective is to develop a scalable decentralized algorithm capable of orchestrating a shape formation around a moving object using robot swarms.

II. Motivation

A. High-Level Motivation

Swarming behavior using vision is quite common in the animal kingdom, birds flock together while migrating to locations and maintain formation at the same time just based on limited field of view vision sensors. Our motivation is to use a similar limited FOV swarm of drones to track a mother ship drone in the center at the same time maintaining the formation while the mother ship guides the flock to the designated target. The second stage of the project involves emulating Roman legion tactics of replacing and maintaining coverage of troops. Roman legions functioned using continuous funneling of troops to the front lines so that exhausted troops could rest or recuperate while at the same time, fallen soldiers could be replaced. Our objective would be to emulate the second formation where one of the drones in the flock loses track or is sent on a scouting mission ahead, while the slower moving mother ship and its flock reorients itself to account for the missing drone.

III. Related Work

This section describes the papers we would be referring to for developing algorithms and software for the whole project. For the detection and tracking part we would be using Yolov5 algorithm [1] right out of the box. We will be retraining this algorithm to do single (or multi-class) identification of drones used in our project. For the tracking part, we have referred to the thesis of Zhu et. al. [2] which basically forms the basis of using a constant velocity Kalman filter to track a moving human in front of the drone for obstacle avoidance. We have adopted a similar tracking approach and designed a custom position-based controller to test the drone-to-drone following functionality.

We would like to work in a moving uncertain environment with drones, this problem was addressed in [3], where they apply leader-follower consensus control and in [4] where they used a directed graph. Another idea we would like to explore is the field of flexible swarm formations [5] when the shape can alter and adjust according to commands or environment changes.

Apart from this, we have also researched on a number of papers that implement swarm algorithms on drone hardware. One such paper describes the use of Visual Inertial Odometry where the researchers [6] at GRASP Lab in UPenn use onboard visual odometry estimates to guide a swarm of 10 drones into different formations. We will try to build upon that concept where we have the individual odometry poses of the drones and we use it only for individual position control, whereas the overall swarm's position control rests on the vision sensors feedback. Another paper [7] works on a similar principle and uses only vision sensors which have a 360-degree fov around the whole environment to track all visible drones in the flocking radius. This information is then used to run a flocking algorithm.

Numerous studies aim to simplify the issue to its core essence. The primary concept involves constructing a representation of the spaces accessible to agents and establishing a set of guidelines for the swarm to adhere to, ensuring goal achievement while avoiding collisions. The concept of employing a straightforward grid and action set is presented in [8]. They combine Hamiltonian Path Planning with a basic set of geometric movement rules, such as allowing swarm agents to move in either a clockwise or counterclockwise direction. In a related work [9], the grid concept is enhanced by integrating numerous small algorithms that synergistically optimize the swarm, enabling the formation of intricate shapes. Mathematical proofs demonstrate the convergence and effectiveness of this approach.

IV. Problem Statement

The objective of the project is to solve the problem of vision based formation control and with respect to that we have the following objectives

- Localize the follower drone with respect to the tracked object
- Control the follower drone with respect to the tracked object
- Implement smooth transition between formations depending on number of followers

The overall goal of the project can be explained in the following images,

The following image is of stage#1 represented here in Figure 1

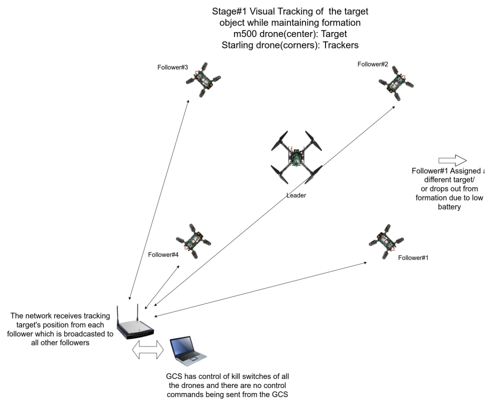

Stage#1 Visual Tracking of the target
object while maintaining formation
m500 drone(center): Target
Starling drone(corners): Trackers
Follower#3
Follower#2
Leader
Follower#1 Assigned a different target/ or drops out from formation due to low battery
Follower#4
Follower#1
The network receives tracking target's position from each follower which is broadcasted to all other followers
GCS has control of kill switches of all the drones and there are no control commands being sent from the GCS

Fig. 1. Representation of the drone swarm formation control

The second stage is of stage#2 where the actual formation change happens is represented in Figure 2

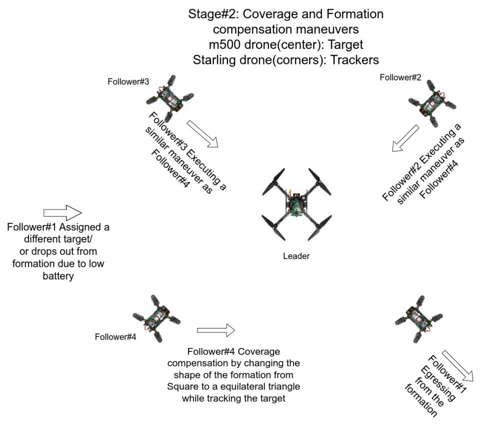

Stage#2: Coverage and Formation
compensation maneuvers
m500 drone(center): Target
Starling drone(corners): Trackers
Follower#3
Follower#2
Follower#3 Executing a similar maneuver as Follower#4
Follower#2 Executing a similar maneuver as Follower#4
Follower#1 Assigned a different target/ or drops out from formation due to low battery
Leader
Follower#4
Follower#4 Coverage compensation by changing the shape of the formation from Square to a equilateral triangle while tracking the target
Follower#1 Egressing from the formation

Fig. 2. Representation of the drone swarm formation control where one drone drops out

The third stage represents the final formation where the drones are reoriented into an isocelesequilateral triangle, which is given in figure 3

The first objective of localizing the follower agents in the swarm is achieved using a simple assumption that the

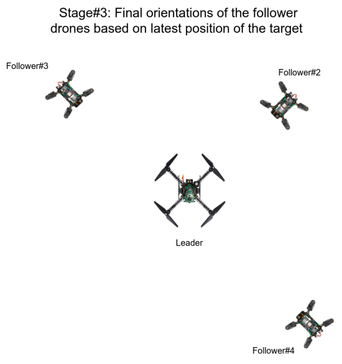

Stage#3: Final orientations of the follower
drones based on latest position of the target
Follower#3
Follower#2
Leader
Follower#4

Fig. 3. Representation of the final drone formation once the follower#1 drops out in Stage#2

object being tracked is the origin of the swarm formation and the agents are the vertices in different quadrants of the 2D cartesian plane. The same is represented in Figure 4

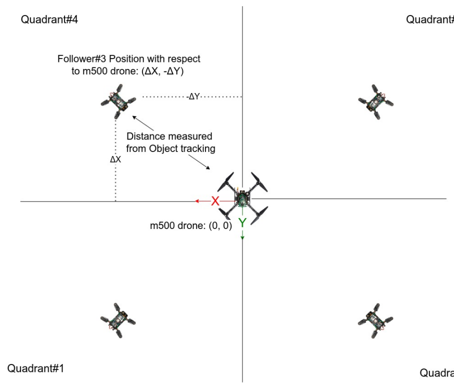


Fig. 4. Representation of drones coordinates in swarm formation

The coordinates of each drone will be figured out based on the assignment of quadrants hard coded during initialization. Whenever a drone performs a formation change maneuver the final location of the drone in the swarm system would be pushed along with the target quadrant which would help in localizing the drone back based on the feedback from vision sensors.

The problem is formulated according to the assumption that all follower drones lie on a circle whose radius is defined by the object being tracked on the center. Another assumption is that if the number of drones decrease the inter agent distance increases this is to make sure that the radius of the circle remains the same and at the same time the drones don't crash into each other. The interagent distances form the edges of a regular polygon inside of the circle.

The algorithm initiates from a predefined regular polygon inscribed within a circle whose radius is determined by object tracking. Each agent possesses knowledge of its global position and maintains communication with its two neighboring agents. The global position is determined using the convention illustrated in Figure 4. Every agent shares specific data with its

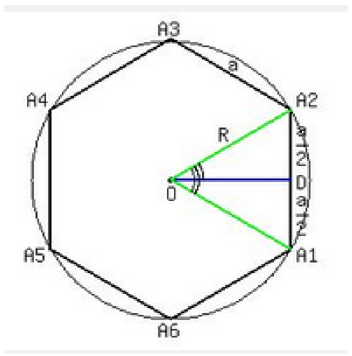


Fig. 5. Regular polygon inside of a circle

neighbors, including its global position and the internal angle of the current polygon formation, derived from the triangle formed by the drone and its neighbors. This information exchange occurs continuously. A subroutine checks whether the neighbors' angles remain consistent; a change indicates a new agent joining or leaving. By analyzing the previous angle information and the direction of angle change, we ascertain the number of drones present and consequently determine the required shape formation. To enhance algorithm speed and collision prevention, we designate an anchor drone that detects changes in its neighbors and specify the movement direction as either clockwise or counterclockwise. Starting from the anchor drone and using the calculated edge length, we establish new target positions along the chain.

As mentioned each drone would need to figure out its internal angle based on the positions received from the neighboring drones and in a regular polygon the internal angles are given by: $Interior\ angle = 180 - \frac{360}{n}$ where n is the number of sides of a polygon, the sum of interior angles in a regular polygon is given by $(n-2)180$, $a = 2Rsin(\frac{\pi}{n})$ where a is calculated using the Euclidean formula for two points A and B on a polygon which has coordinates (x_1, y_1) and (x_2, y_2) the euclidean distance is given by:

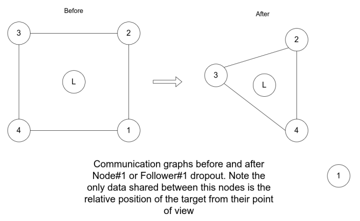
$$Side\ length\ of\ polygon = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

Finally for finding the interior angle made by agent B is given by:

Given coordinates of points $A(x_1, y_1)$, $B(x_2, y_2)$, and $C(x_3, y_3)$, we can calculate the angle $\angle ABC$ using the following equations:

$$\begin{aligned}
\text{Vector } \mathbf{AB} &: (x_2 - x_1, y_2 - y_1) \\
\text{Vector } \mathbf{BC} &: (x_3 - x_2, y_3 - y_2) \\
\text{Dot Product } \mathbf{AB} \cdot \mathbf{BC} &: (x_2 - x_1)(x_3 - x_2) + (y_2 - y_1)(y_3 - y_2) \\
\text{Magnitude } \|\mathbf{AB}\| &: \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\
\text{Magnitude } \|\mathbf{BC}\| &: \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2} \\
\cos(\angle ABC) &: \frac{\mathbf{AB} \cdot \mathbf{BC}}{\|\mathbf{AB}\| \cdot \|\mathbf{BC}\|} \\
\angle ABC \text{ (radians)} &: \arccos\left(\frac{\mathbf{AB} \cdot \mathbf{BC}}{\|\mathbf{AB}\| \cdot \|\mathbf{BC}\|}\right) \\
\angle ABC \text{ (degrees)} &: \angle ABC \text{ (radians)} \times \left(\frac{180}{\pi}\right)
\end{aligned}$$

Now the agent can transmit its own position and the interior angle it is making at that position to neighbors which can be further used by other drones to triangulate if a change in the formation has happened. If a drone drops out in the formation the position of the drones and their interior angles would have a mismatch which the algorithm would detect and recalculate the required the edge distance. We assume here that one of the drone would act as an anchor and the rest of the drones would get pulled towards the anchor based on the new calculated edge distances.



Communication graphs before and after Node#1 or Follower#1 dropout. Note the only data shared between this nodes is the relative position of the target from their point of view

Fig. 6. Communication graphs of before and after drop out of agents

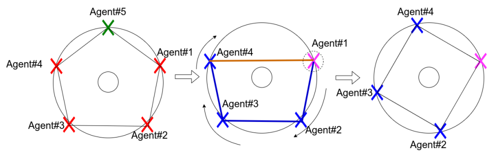


Fig. 7. Transition of formation graphs shown. The green drone initially drops out in the left most formation then in the middle formation the drones figure out change in interior angles and edge distances and decide an anchor and the moving agents. In the third formation after finally moving they achieve a regular square formation

In the algorithm showed in Algorithm 1 demonstrates the action a single drone has to take as soon as it detects change in formation. The change in the drone formation would be communicated by the drone which is dropping out to its immediate neighbors which would then establish communication with each other and decide the anchor drone. Based on the selection of the anchor drone new position would be calculated.

```
Data: Share position and phase angle inside the circle
      formation
Result: If someone drops out reorient the swarm
initialization;
while Neighbors remain unchanged do
    Share position and angles to the neighbors;
end
if Neighbor change detected then
    Select right agent as anchor drone;
    Select desired shape based on the difference in
      phase angle;
    Calculate the new phase angle;
    Calculate the new position;
    Move the drone in CCW manner;
end
```
Algorithm 1: The algorithm which will run on each drone

V. Methodology

The project is divided into three stages due to the inherent challenges of working with hardware and the complexity of decentralized communication and control with real-time onboard computer vision.

Fallback Goal: Develop a scalable decentralized shape formation simulation to demonstrate the basic algorithm. Desired Goal: Implement the swarm algorithm and the object detection and target localization on the actual hardware separately. Ambitious Goal: Integrate and implement both the swarm algorithm and the object detection and target localization on the actual hardware simultaneously.

VI. Experiments

A. Simualtion

In this section we simulate a multi agent system running a swarm algorithm where multiple agents are dropped out of a formation. The formation is same as described in Problem statement where nodes act as vertices of a polygon inscribed by a circle. The various stages of the simulation developed for that are shown in Figure 8, where an arbitrary number of drones are being dropped from a random location, and the swarm responds to the change.

1) Implemented algorithm: In the algorithm showed in Algorithm 2, it demonstrates the action a single drone has to take as soon as it detects change in formation. The change in the drone formation would be communicated by the drone which is dropping out to its immediate neighbors which would then establish communication with each other and decide the anchor drone. Based on the selection of the anchor the planning algorithm would start. The anchor would calculate the number of drones in the swarm before the dropout using the unchanged angle and geometry. It would assume that only one drone has been dropped, it would calculate the new phase angle of the drone on its right, the next agent would do the same, and this chain would go until it reaches the anchor again. If the angle between the anchor and its left and right neighbors are not the same the loop would continue, the anchor would assume that one more has been dropped and the chain would be adjusted accordingly, until the angle of the neighbors of the anchor becomes equal. When that happens the target locations are saved and the drone movement can be executed.

2) Simulation results: The algorithm was implemented in Python, the results can be seen in Figure 8. The swarm consisted of six agents, and they formed a regular hexagon, three agents were dropped, and when the change was detected the swarm was pulled toward the anchor, in the first iteration it was assumed that the new shape was a pentagon, the angles were adjusted since it did not achieve the regular shape in the next iteration rectangle was assumed that it is a square, and finally it was assumed that it was a triangle which ultimately satisfied the criteria. Based on the simulation the algorithm show decentralized communication principles and scalability.

```
Data: Share position and phase angle inside the circle
      formation
Result: If someone drops out reorient the swarm
initialization;
while Neighbors remain unchanged do
    Share position and angles with the neighbors;
end
if Neighbor change detected then
    Start planning;
    Select the left agent as the anchor drone;
    Assume that only one drone was dropped;
    Calculate the new phase angle;
    Calculate the new position;
    Pull the swarm toward the anchor CW;
    while The angle between the anchor and the right
      and the left neighbor are not equal do
        Assume one more drone was dropped;
        Calculate the new phase angle;
        Calculate the new position;
        Push the drones in CCW manner;
    end
    Save the target coordinates;
    Move the drones to the target locations;
end
```
Algorithm 2: The algorithm which will run on each drone

B. Hardware Setup

The project involves using knowledge of tools from this class as well as previous courses we took during our degree program. To enumerate we have itemized the list of tools and their short descriptions and their intended purpose:

- m500 Drone: This drone is the mother ship of the formation. This drone has a downward facing fisheye camera which does Visual Inertial Odometry to find its pose with respect to a inertial frame. In our case this drone would be the only one receiving commands from the Ground station which would lead the whole flock to move just based on vision estimates of the m500 drone position.

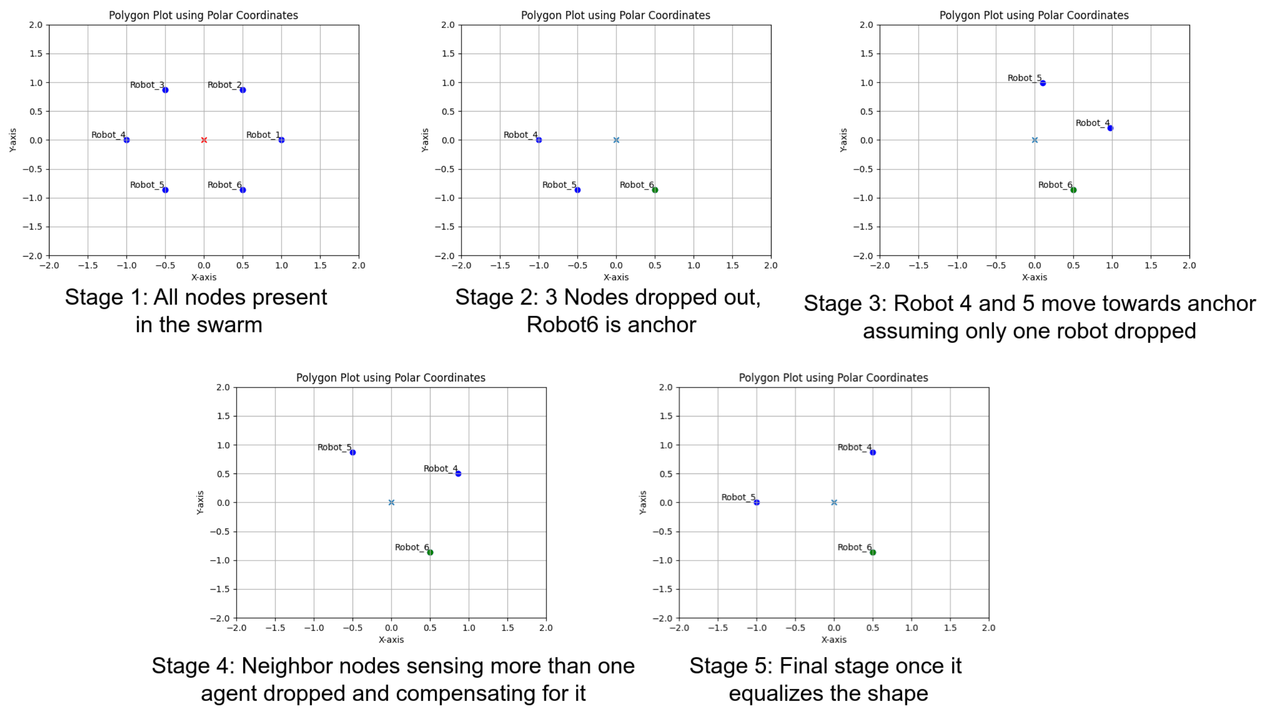


Fig. 8. Simulation stages of multi agent drop example in a circle inscribed polygon formation

- Starling Drone: This drone is part of the flock formation and is the flagship autonomy drones from the company MODAL AI. The drone has three types of sensors, Fisheye sensors are used for Visual odometry, The front facing RGB Hires camera's data will be used for object detection and the front facing Time of flight camera will be used to fuse detection and point cloud data to localize the target (m500 drone).
- PX4 Framework: The PX4 Autopilot Framework is the most commonly used open source firmware for unmanned vehicles. This range from ground rovers to vtol aircrafts to underwater ROVs. In our drones we plan to use it as the base control firmware upon which high level code would be written.
- ROS2: Robot Operating System would act as a middle ware for internal communication and exchange of data within individual drones and inter drone communication of pose estimates of the mother ship between flock drones. In our case, the code right now is running on ROS2 humble on the drones, that provides better user support.
- Yolov5 [1]: We will also use Yolov5 single shot detector for object detection. This algorithm is being used as it can be vectorized for the GPU architecture on board the drones we are using and the supporting libraries are pre installed.

Apart from the above tools, we will use GIT, ROS-Gazebo, Python packages like numpy, matplotlib, pytorch for testing and prototyping code. We plan to use C++ end to end for efficiency of running it on board the drones and to take advantage of clean C++ builds which handle memory management better then python.

1) Object detection algorithm implementation on the drones: The objective of this implementation was to provide the swarm agent low latency feedback of it's distance from the common origin in the swarm formation. This was supposed to be used as a radius parameter in the data shared amongst agents in the communication graph. The proposed pipline was to set the radius from one of the agents based on a service call to that agent from the ground. The algorithm implemented real time sensor fusion of two different sensors i.e., the RGB camera and the Time of Flight (ToF) camera. This sensors are onboard the Starling v2 drones. The sensor fusion pipeline is shown in Figure 9

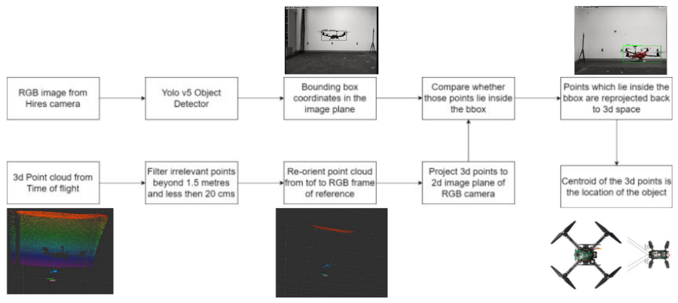

RGB image from Hires camera
Yolo v5 Object Detector
Bounding box coordinates in the image plane
Compare whether those points lie inside the bbox
Points which lie inside the bbox are reprojected back to 3d space
3d Point cloud from Time of flight
Filter irrelevant points beyond 1.5 metres and less then 20 cms
Re-orient point cloud from tof to RGB frame of reference
Project 3d points to 2d image plane of RGB camera
Centroid of the 3d points is the location of the object

Fig. 9. Sensor Fusion Pipeline for Object detection and tracking

There were various challenges faced while implementing this on the edge computer, ranging from CPU memory overruns to CPU Overheating and GPU overloading. Concepts like Multi-threading, GPU Parallelization, Pointer manipulation and object oriented programming were used to make the code efficient.

After doing all this changes, the frame rate of object detection was increased from 5 FPS to 15 FPS. The mean Average Precision of the YoLo detector was 98.5%. Even with such stellar performance implementing this in the control loop of a quad rotor where multiple commands for stabilization and flight mode changes are coming in a bit complicated. We came up with a smarter solution of just confirming the radius while the drones are on the ground and making setpoint adjustments for once, rather than implementing a looped service call.

2) Revisit algorithm implementation in ROS: To implement the decentralized algorithm, ROS2 Humble was employed. Two custom message types were developed: one for the drone's status and another to act as a bridge for setting setpoints in the flight controller, thereby enabling drone movement. The swarm algorithm was implemented using these interfaces. The architecture comprises three nested services. The DropDrone service, illustrated in Figure 10, lands the selected drone and simultaneously connects its neighboring drones to each other. The next service in the sequence is called from the left neighbor, which in the anchor.

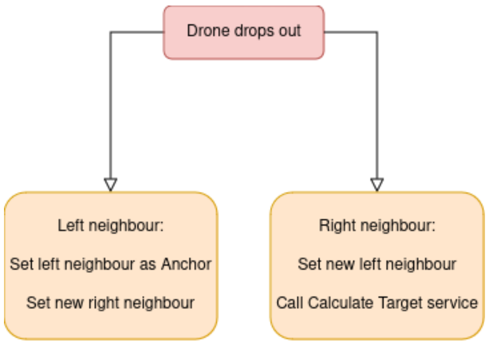


Fig. 10. DropDrone Service

The CalcTarget service, depicted in Figure 11, calculates the target positions for the subsequent drones and continues to call the same service until it reaches the anchor drone. The anchor drone, knowing the angle of its neighbors prior to dropout, calculates the number of vertices in the original shape. By subtracting one vertex, it determines the desired internal angle.

Using the logic shown in Figure 12, the increment angle for each drone can be calculated. With this increment angle and the radius, the relative displacement can be determined in each drone's local frame of reference. When the CalcTarget chain completes, the anchor calls the MoveDrone service (Figure 13 below), which sends the setpoints to the flight controller. The drones then move sequentially, forming a regular shape, which in this case is a triangle.

3) Final Results: The final hardware results are displayed below in Figure 15, and the videos are available via the attached link. Target localization was implemented only while the drones were stationary displayed below in Figure 14.

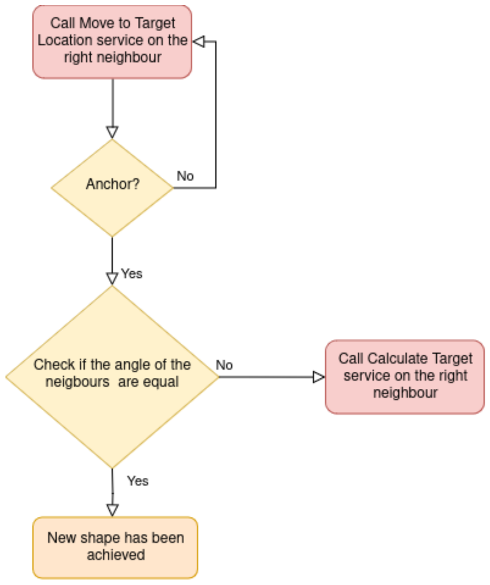

Call Move to Target Location service on the right neighbour
Anchor?
No
Yes
Check if the angle of the neigbours are equal
No
Call Calculate Target service on the right neighbour
Yes
New shape has been achieved

Fig. 11. CalcTarget Service

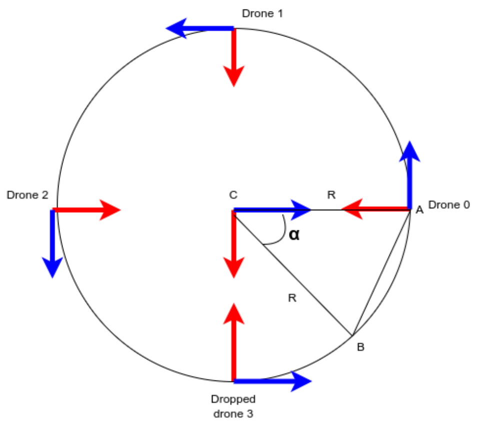


Fig. 12. Calculate Target graph

They measure the distance between themselves and the central drone, using this value as the radius for the regular shape. The drones' initial phase angles were predefined to enhance the solution's robustness. After the drones take off, one drops, and the left neighbor becomes the anchor, initiating the CalcTarget service in a CCW direction. Once this is completed, the MoveDrone service is called in CCW direction, moving each drone to its calculated target position.

4) Discussion: The desired goal was achieved: the swarm and vision algorithms were implemented separately. This represents a significant step toward the project's aim of providing an aerial robotics platform that leverages AI for swarm applications. While there is still much to accomplish, the progress made with the decentralized swarm algorithm and onboard object detection is promising.

Future work includes achieving the ambitious goal of integrating the two parts. Additionally, demonstrating scalability on hardware will require more drones, and the number of

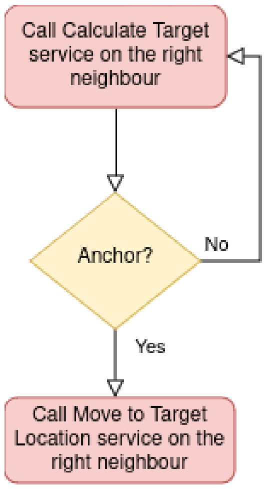

Call Calculate Target service on the right neighbour
Anchor?
No
Yes
Call Move to Target Location service on the right neighbour

Fig. 13. MoveDrone Service

drones that can be dropped could also be increased. On the vision side, once it can run onboard during flight, it should provide dynamic feedback. For example, if the central drone moves, the entire swarm should replicate the movement. Additionally, enabling the swarm to orbit around a target would be desirable.

VII. Conclusion

In this project, we delved deeply into drone swarm applications, a relatively new field with much to explore. We successfully implemented the swarm and vision algorithms separately, marking a significant milestone towards creating an aerial robotics platform that leverages AI for swarm applications. This accomplishment demonstrates the potential of decentralized communication and control in real-time onboard computer vision.

Overall, the decentralized swarm algorithm and onboard object detection are promising advancements. With continued development and refinement, we are moving closer to realizing a robust aerial robotics platform for various swarm applications.

VIII. Acknowledgement

The project benefitted greatly from Dr. Danilo Romero's support, offering valuable guidance and motivation for its execution. We are thankful to Dr. Romero for granting us access to state-of-the-art hardware resources, enabling us to learn and implement cutting-edge AI algorithms and develop drone software. We extend our special appreciation to Michael W. Otte for his invaluable contributions and brainstorming sessions on swarm technologies.

We also acknowledge the significant mentorship provided by Professor Joseph Conroy and Professor Zeid Kootbally throughout this project and our Masters degree program. Dr. Conroy's support was instrumental in resolving challenges associated with closed-door hardware ecosystems, enhancing our understanding of open-source drone software. Dr. Kootbally's coursework on ROS2 control and interface architectures has been integral to our codebase development.

References

[1] G. Jocher, "Ultralytics yolov5," 2020. [Online]. Available: https://github.com/ultralytics/yolov5
[2] H. Zhu, "Probabilistic motion planning for multi-robot systems," Dissertation (TU Delft), Delft University of Technology, 2022.
[3] L. Li, J. Gao, and P. X. Liu, "Finite-time robust formation control of multiple aerial robotic vehicles with uncertainties and time-varying complex perturbations," *Communications in Nonlinear Science and Numerical Simulation*, vol. 129, p. 107672, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1007570423005932
[4] Y. Zhu, S. Li, G. Guo, P. Yuan, and J. Bai, "Formation control of uav–usv based on distributed event-triggered adaptive mpc with virtual trajectory restriction," *Ocean Engineering*, vol. 294, p. 116850, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801824001872
[5] B. Alkouz and A. Bouguettaya, "Formation-based selection of drone swarm services," 12 2020, pp. 386–394.
[6] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2017.
[7] F. Schilling, F. Schiano, and D. Floreano, "Vision-based drone flocking in outdoor environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, 2021.
[8] A. G. Curtis, M. Yim, and M. Rubenstein, "Continuous sculpting: Persistent swarm shape formation adaptable to local environmental changes," 2024.
[9] H. Wang and M. Rubenstein, "Shape formation in homogeneous swarms using local task swapping," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 597–612, 2020.

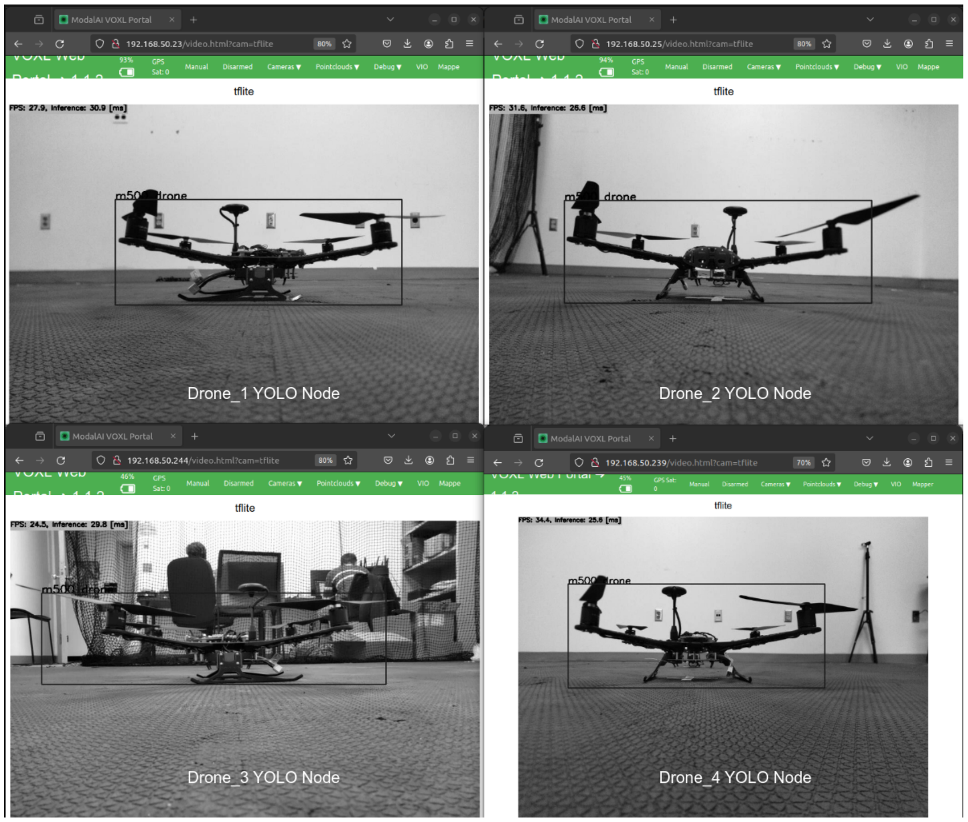


Fig. 14. The target localization during ground state, this image just represents the 2D localization as referred in the object detection subsection we fuse it with the pointcloud to get the 3D target point

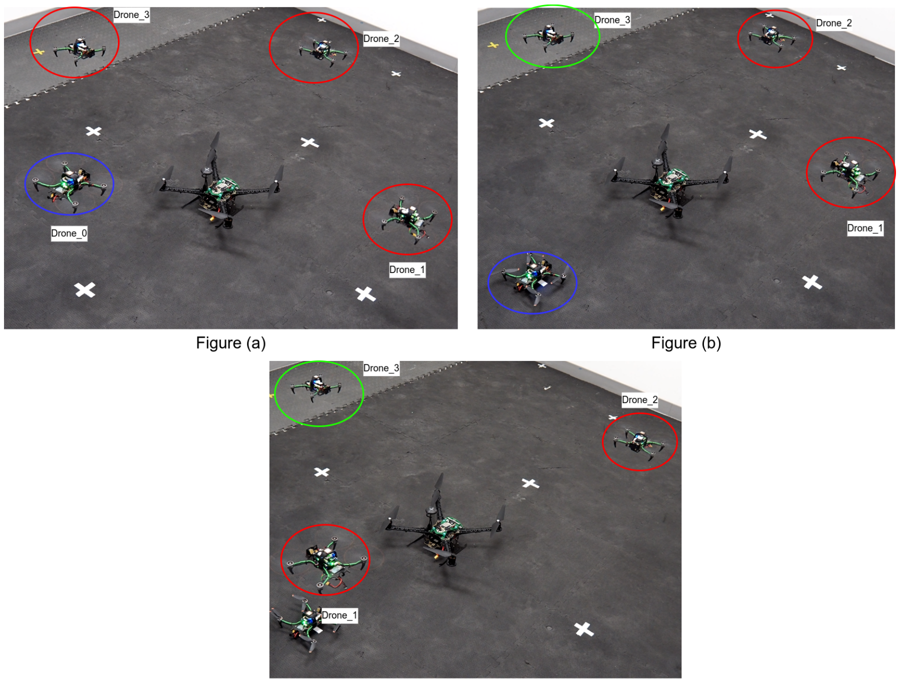


Fig. 15. Hardware experiment stages of multi agent drop example in a circle inscribed polygon formation. Figure (a): All agents flying in a steady state formation, the blue circle represents the agent about to drop; Figure (b): Representing the drone drop maneuver happening with drone_0 dropping out; Figure (c): Representing the maneuver where the drone_1 and drone_2 do a coverage compensation maneuver based on the analytical geometry defined in earlier sections