Self-Synchronized Drone Swarm Report

Author - Taras Lysun

Project Goal

The main goal of this project was to implement a self-synchronized drone swarm system that uses Orange Pi as the computer “brain” for each drone. The system was designed to demonstrate synchronization and command execution inside the swarm, using machine learning models for object detection. The swarm needed to be capable of avoiding obstacles like trees, using a YOLOv8n object detection model. This system emphasizes decentralized coordination and intelligent decision-making across the swarm's drones.

Description of the Drone Swarm

A drone swarm is a set of autonomous drones that work together on bigger tasks. The advantage of using a swarm over single drones is that you don’t need to have each drone operated by a different pilot, but instead one person can handle multiple drones. In this project, synchronization is very important, as many missions rely on time-based tasks or fixed time periods of task completion, so there needs to be a way to synchronize all the clocks of each drone within the swarm.

Synchronization Approaches

Several approaches to drone swarm synchronization exist, including centralized control, decentralized communication, and leader-follower methods. In this project, I chose the hierarchical model. The swarm is organized into a master drone (MD), cluster heads (CH), and common drones (CD). The synchronization is achieved by passing down a clock from the MD to the CHs, which in turn synchronize with the CDs.

System Architecture

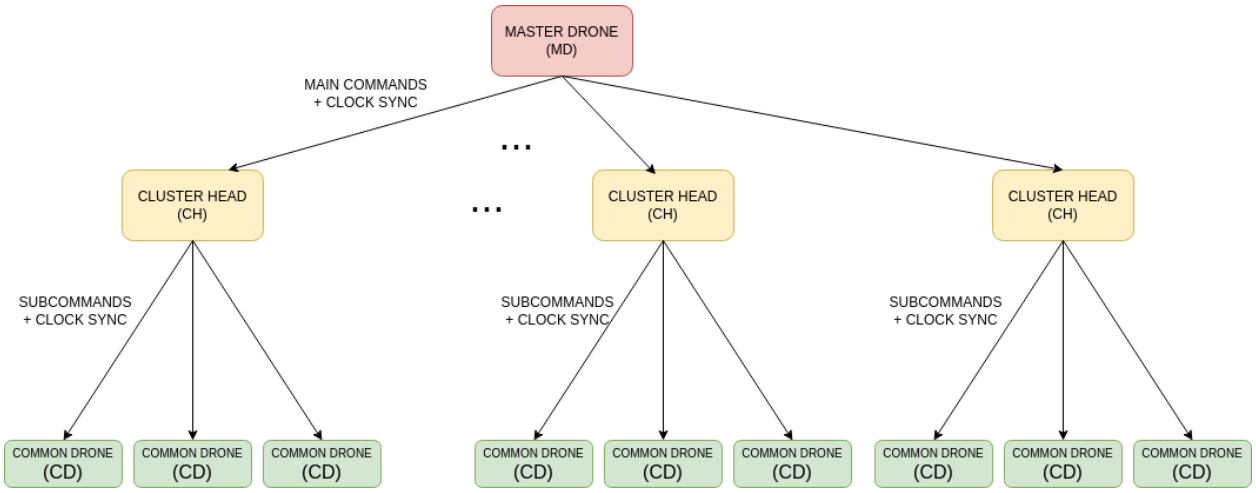
The architecture is built on three types of drones: the master drone (MD), cluster heads (CH), and common drones (CD).

- **Master Drone (MD):** The MD handles high-level operations. It sends commands like "fly forward," "gather around a point," or "synchronize clocks" to the cluster heads. It operates as the leader of the swarm and ensures that the entire system works fine.
- **Cluster Head (CH):** The CHs are the “main workers” as they receive instructions from the MD and manage the common drones in their own

clusters, they decide and communicate with both levels of drones. They also handle object detection and obstacle avoidance.

- **Common Drone (CD):** CDs follow the commands sent from the CHs. They are responsible for executing low-level simple tasks like following their cluster head or forming into clusters.

All communication between drones is handled via UDP, because the speed of information exchange is more crucial than each individual package. This architecture is designed to balance control between centralization (via the MD) and distributed execution (via CHs and CDs).



Object Detection with YOLO

To implement obstacle avoidance, each cluster inferences from a YOLOv8n model, fine-tuned to detect trees in its environment. The model was trained using a dataset of tree images, and the fine-tuning process was designed to improve its accuracy in detecting trees specifically.

The model was then exported to ONNX for efficient inference on the Orange Pi devices running the CHs. The CH drones can identify trees in their flight path, ensuring that they can navigate around obstacles autonomously while moving to their assigned target coordinates.

Visualization Using Unity

A visualization of the drone swarm was created using Unity. In this visualization:

- Each drone is represented as a game object within the Unity environment.
- The cluster heads are equipped with virtual cameras that capture their surroundings.
- The captured images are sent to a backend system via HTTP requests for real-time object detection and obstacle avoidance.

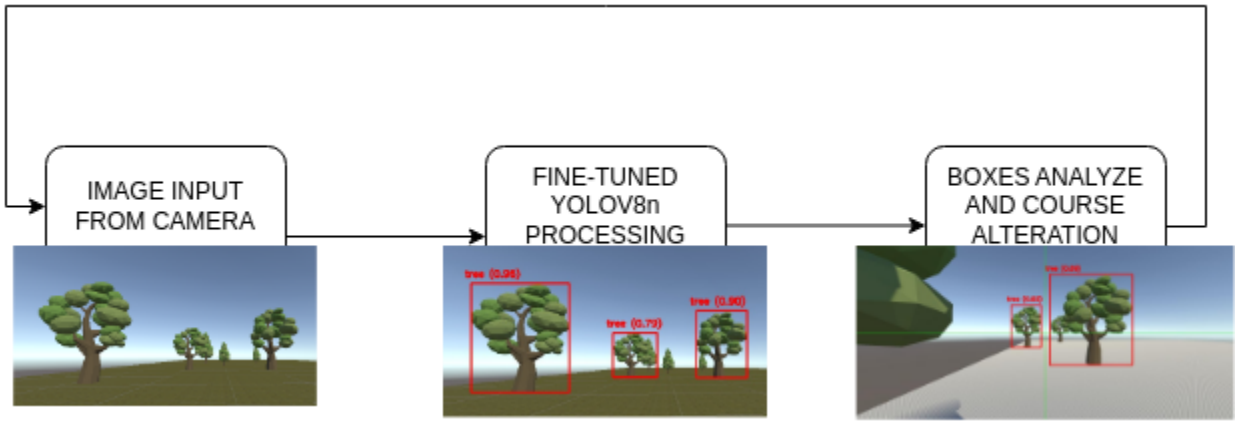
The game objects in Unity receive the drones' coordinates via UDP and position themselves accordingly in the simulated environment. This visual feedback provides a real-time representation of the swarm's behavior and movement.

Pipeline Description

The obstacle avoidance pipeline is executed on the backend as follows:

- **Receive Image:** The CH drones capture images from their surroundings and send them to the backend.
- **Inference:** The YOLOv8n model detects trees in the images. If a tree is detected in the drone's flight path, the system calculates a possible route to pass it.
- **Pass trees and other obstacles:** The system checks if the center of the detected tree overlaps with the center of the image, which represents the drone's current flight direction and decides which direction to use to pass the obstacle.

This process allows the CH to dynamically avoid obstacles while ensuring it still moves towards its target.



7. Conclusion

This project successfully demonstrates a self-synchronized drone swarm system that uses decentralized communication, obstacle detection, and avoidance. Through Unity, I was able to visualize the swarm in real-time, showcasing its behavior in a simulated environment. It was a really interesting experience and gave me a deeper understanding of swarm algorithms and different problem solving approaches.

Sources:

1. Development of Self-Synchronized Drones' Network Using Cluster-Based Swarm Intelligence Approach
2. GitHub
3. Tree Detection datasets:
    - https://universe.roboflow.com/models/object-detection
    - https://universe.roboflow.com/toytopia/tree-bsm8r
4. Video Demonstration of object detection and avoidance