# Multi Drone RL Strategy for Swarm Formation, Survival and Narrow-Alley Navigation

Supervisor
Dr Sujit P.B.

Presented by
Ananya Gandhi
20319

## Motivation

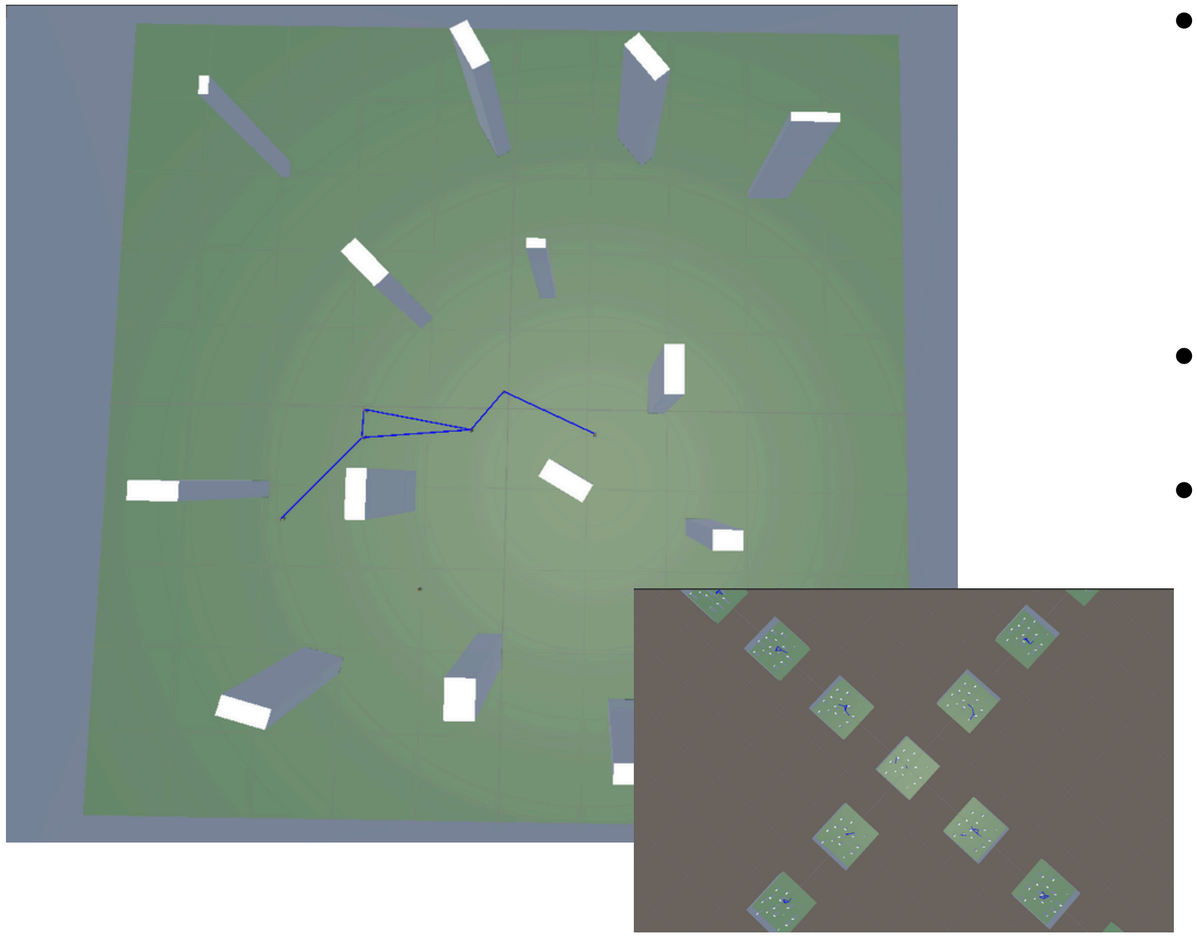
To develop a robust, scalable and adaptive swarm like structure of UAVs where the drones can be operated by a human as well as take control to carry out operations autonomously for various missions like search-and-rescue, emergency food delivery where the payload is such that a single drone cannot carry the whole payload and it has to be distributed among multiple drones.

## Problem Statement

- Maintaining stability even if the drone collapses or fails.
- Collision avoidance with each other and in narrow alleys.
- Navigating through densely built environments pose significant challenges.
- Managing multiple drones can be complicated with human operation of one drone.

- Customized Environment
- Reward Shaping
- convergence of algorithm
- exponential reduction in loss

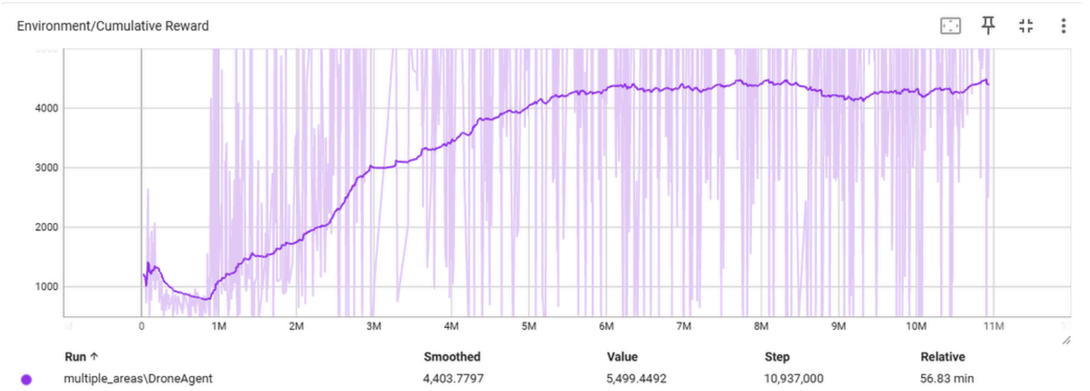
# Customized Training Environment



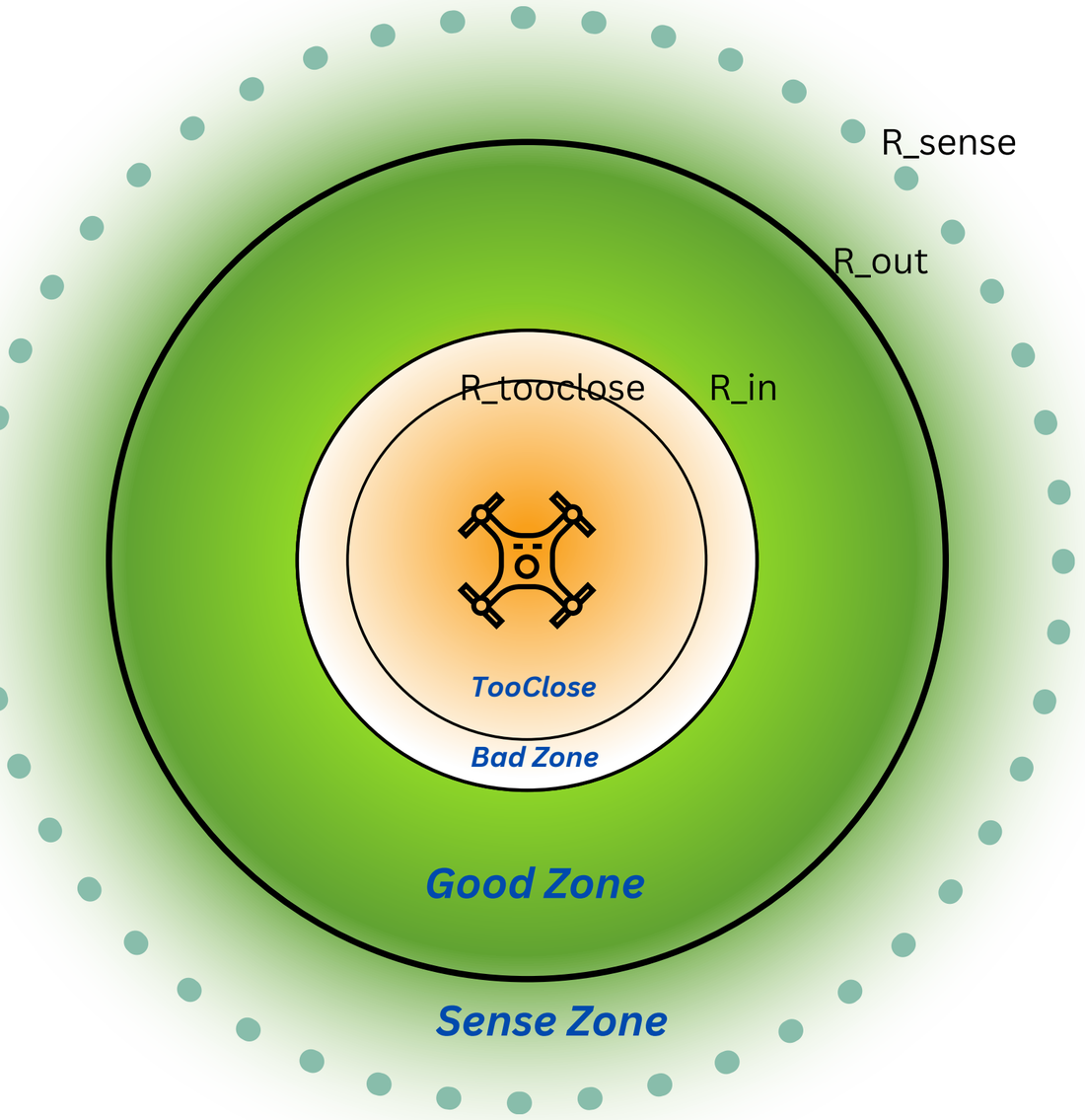
- Randomised obstacles
- Multiple Training Areas: lead to faster Training and Convergence

## Methodology Outline:

- We will use the Unity ML-Agents library to train this model as the base model using Proximal policy optimization (PPO), we now introduce randomised waypoints in the environment which the leader drone is meant to follow.
- this should train the follower drones to maintain the swarm around the leader drone during motion.
- once this model is trained, it should give us freedom to manually control the leader drone expecting the follower drones to maintain their swarm formation.



# Reward Shaping




```
// Reward Valuations in High to Low Order
private readonly float swarmationReward = 200.0f;
private readonly float insideGoodRegionReward = 40.0f;
private readonly float insideSensingZoneReward = 10.0f;
private readonly float intraSwarmCollisionPenalty = -80.0f;
private readonly float obstacleCollisionPenalty = -60.0f;
private readonly float boundaryCollisionPenalty = -60.0f;
private readonly float tooClosePenalty = -30.0f;
private readonly float insideBadRegionPenalty = -10.0f;
private readonly float rr_factor = 0.5f;

private bool collidedWithObstacle = false;
private bool collidedWithDrone = false;
private bool collidedWithBoundary = false;
```

## Types of Rewards:

1. TooClosePenalty
2. insideGoodRegionReward
3. insideBadRegionPenalty
4. droneProximityReward
5. insideSensingZoneReward

1. ObstacleCollisionPenalty
2. ObstacleProximityPenalty
3. BoundaryCollisionPenalty
4. Intra-SwarmCollisionPenalty

1. survivalReward
2. lostPenalty
3. SwarmationReward
4. Intra-Swarm Rebound Reward Component

- When a drone encounters another drone, it receives **a Swarmation Reward.**

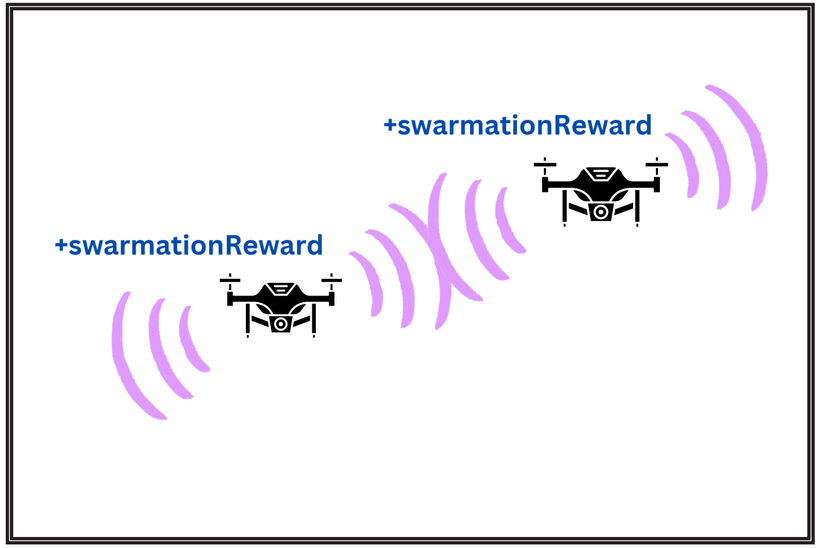
- Additionally, **a Rebound reward** is applied to every drone in the swarm when another drone receives a reward. **This promotes swarm formation and exploration.**

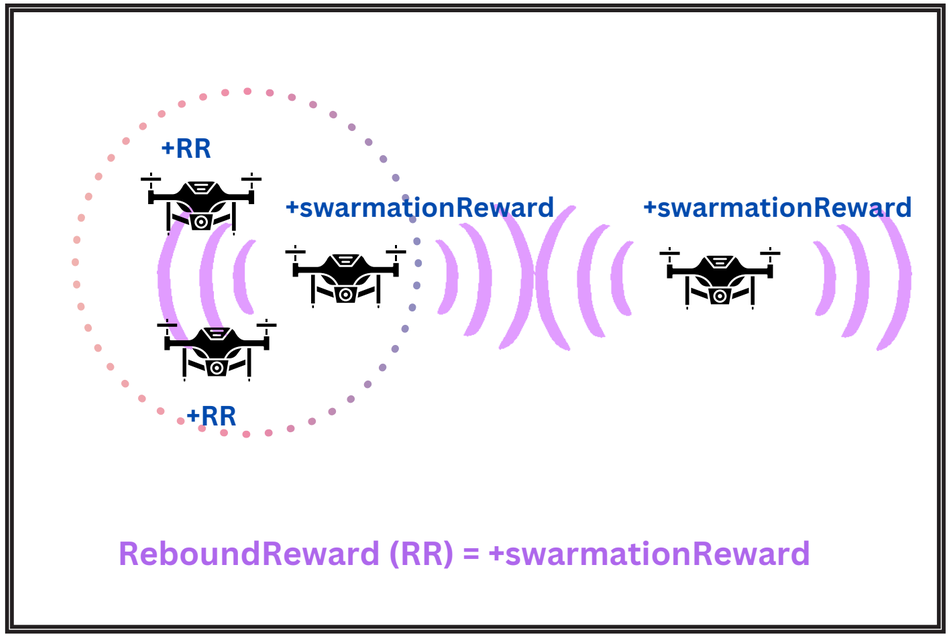
The **Swarmation Reward** is set to a very high value, much greater than the **Collision Penalty** and **Bad Zone Penalty.**

**|Swarmation Reward| > |Collision Penalty| > |Good Region Reward|**
so that the swam prioritises disintegration over the loss of its member drone.

- To ensure proper navigation in narrow alleys, we set the penalties as follows:
**Bad Zone Penalty < Too Close Penalty < Obstacle Collision Penalty ≪ Intra-Swarm Collision Penalty**

The **much greater intra-swarm collision penalty** as compared to the **drone-obstacle collision penalty** ensures that the drone rather taking just itself out rather than along with another drone in the swarm in case of an extreme situation.



# Novelties Introduced

## The Novelty of the Rebound Reward Component

- The Intra-Swarm Rebound Reward Component not only **ensures that the swarm stays intact** but **also give exploration powers to an already formed swarm** and **natural Leadership take-overs in case of a missing/death of a Leader.**

- This is the most important component, ensuring the swarm stays intact while allowing exploration. It also facilitates natural leadership takeovers in case of the leader's absence or failure.

## The Novelty of the Function private void AddDroneObservationsRandomly(VectorSensor sensor)

## Key Problems faced:

1. Dynamicity of number of SensedDrones for each DroneAgent
2. Neural networksrequire fixed-sized inputs. Directly feeding a variable number of observations into the network is problematic. Filling empty inputs with zeroes act as misleading values

PTO

# Solution

Since, Neural Networks work on relativistic weights, we exploit this property by giving zeroes for randomized inputs.

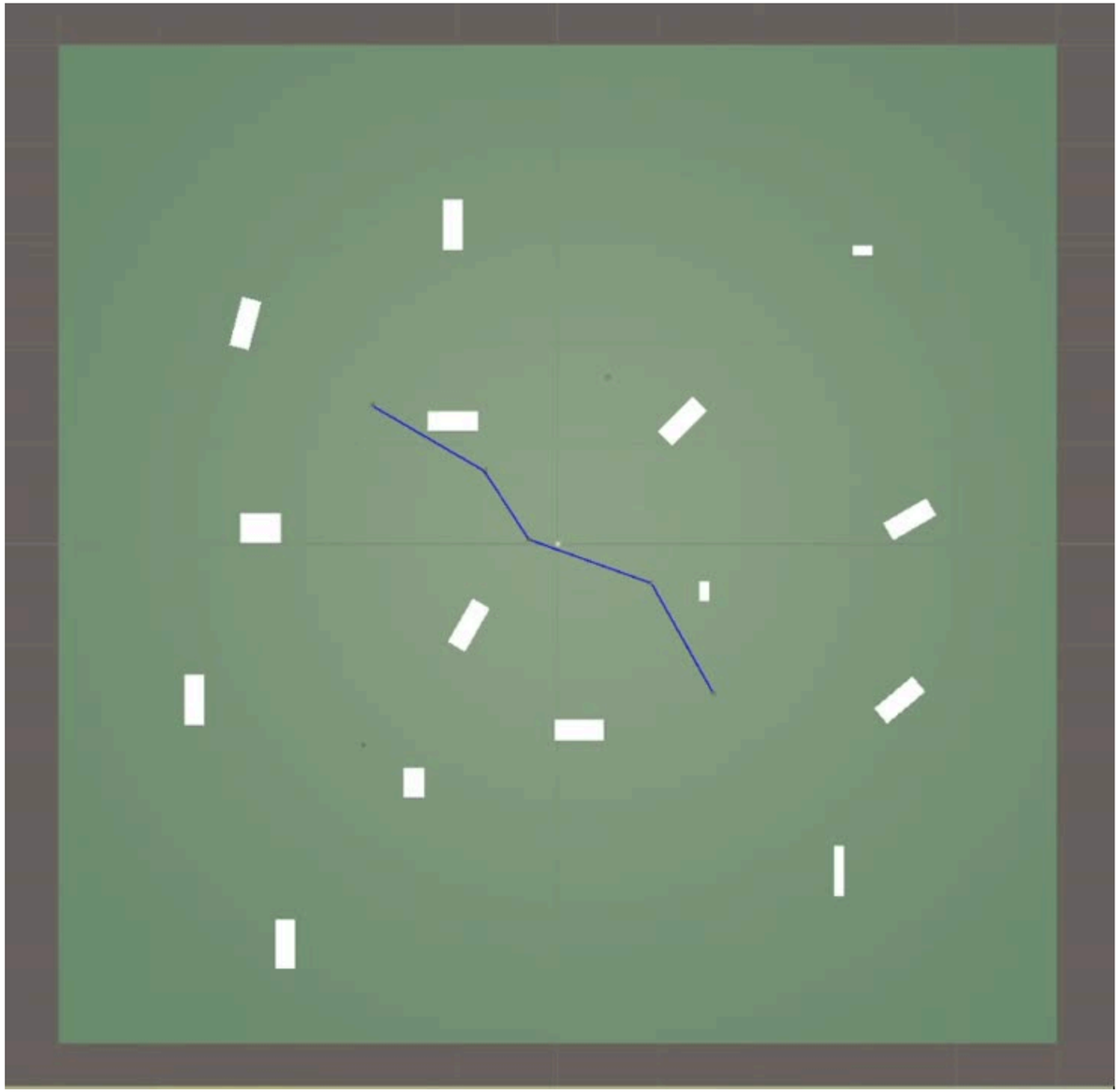
This way, the Zeroes are by default considered of trivial significance and ignored, thus bypassing them being undesirably picked up by the NN as spatial values for sensedDrones.

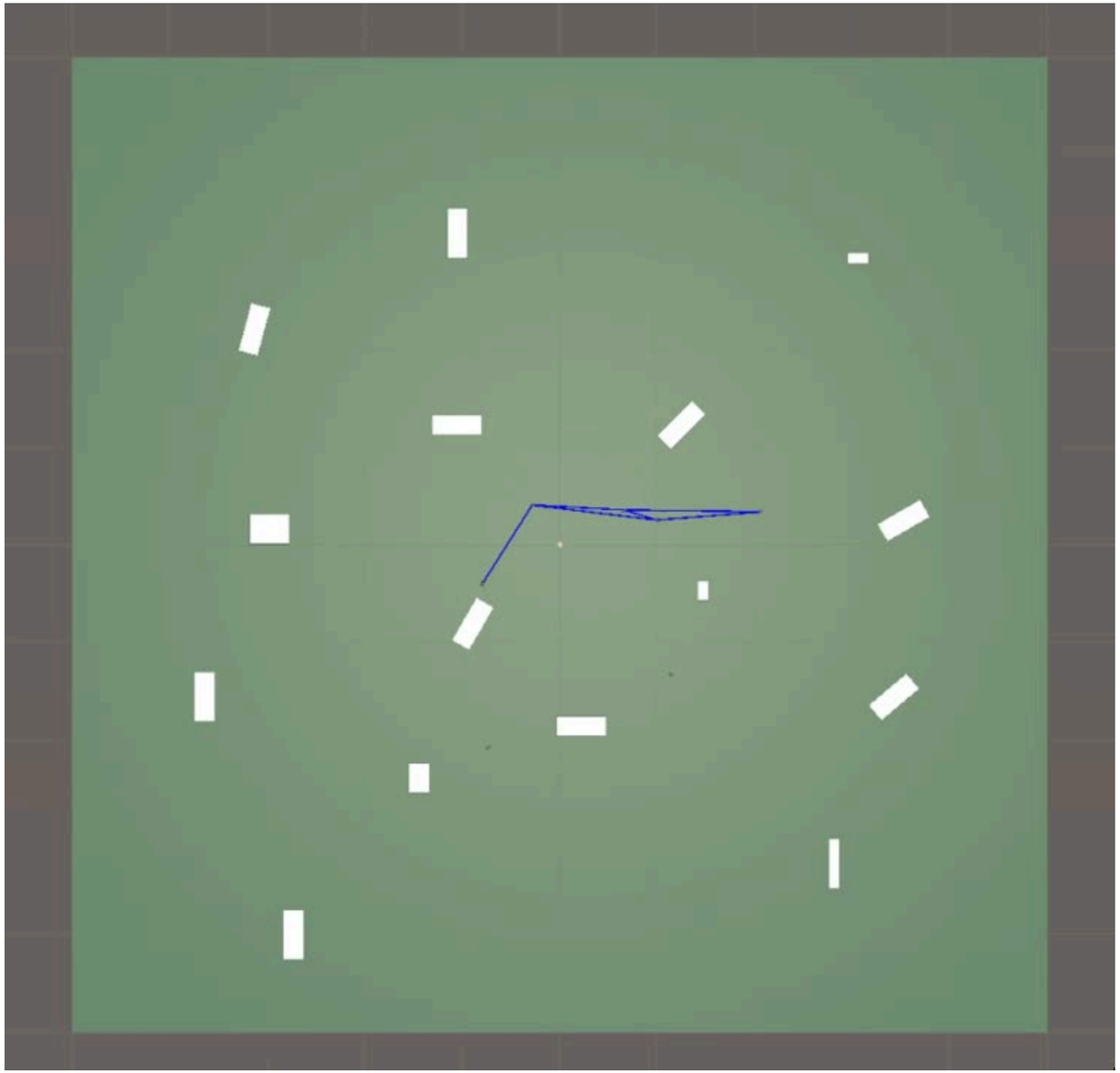
```
1 reference
private void AddDroneObservationsRandomly(VectorSensor sensor)
{
    int totalSlots = Drone_Values.NumberDrones - 1;
    Dictionary<int, Vector2> observationSlots = new Dictionary<int, Vector2>();

    // Fill the dictionary with Vector2.positiveInfinity for all slots
    for (int i = 0; i < totalSlots; i++)
    {
        observationSlots[i] = Vector2.zero;
    }

    // Randomly assign sensed drone observations
    foreach (var drone in sensedDrones)
    {
        int index;
        do
        {
            index = random.Next(totalSlots);
        } while (observationSlots[index] != Vector2.zero);

        Vector3 droneRel = transform.InverseTransformPoint(drone.transform.position);
        observationSlots[index] = Vector3to2(droneRel / Drone_Values.R_sense); // Normalising
    }

    // Add observations to the sensor in order
    for (int i = 0; i < totalSlots; i++)
    {
```

Novelty of this approach:

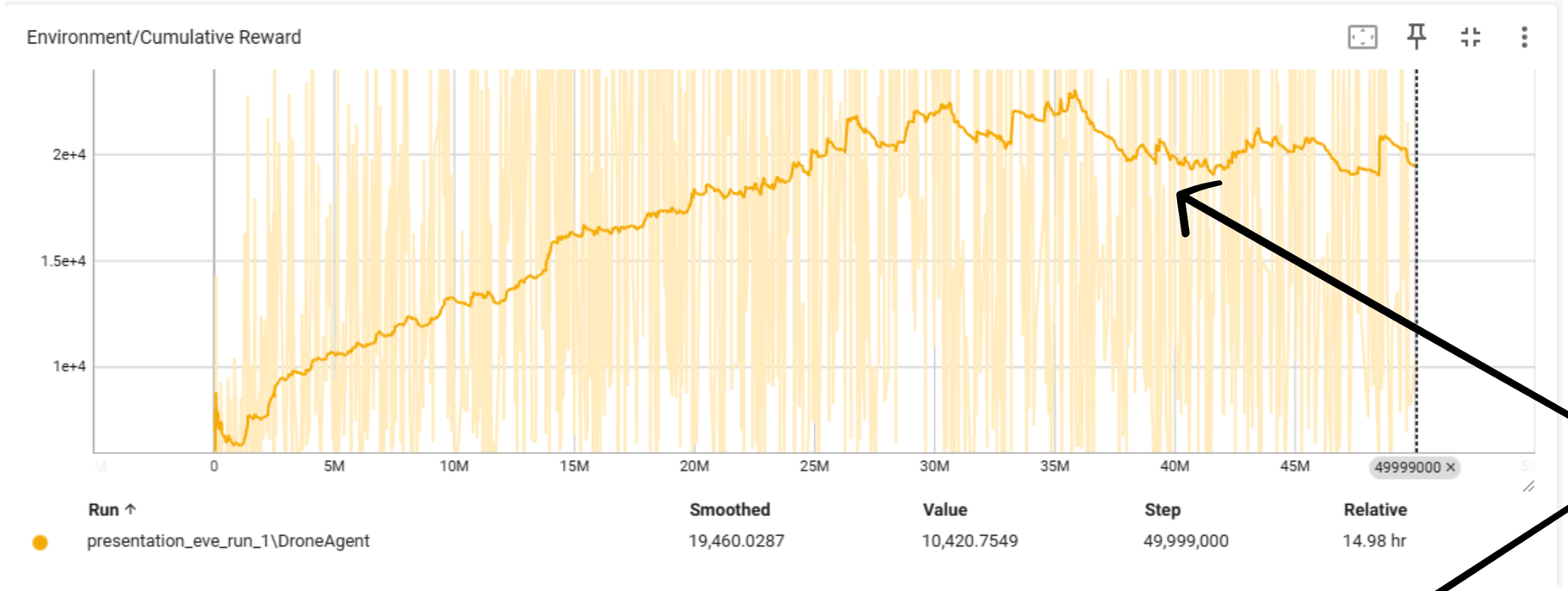
1. This solves a rather infamous and bothersome issue in dynamic observation cases for NN in a rather simplistic way, handling a variable number of sensed drones while maintaining a consistent input format.
2. allows the network to scale seamlessly to varying numbers of sensed drones while using a fixed input size
3. Removes bias induced in the NN due to input ordering

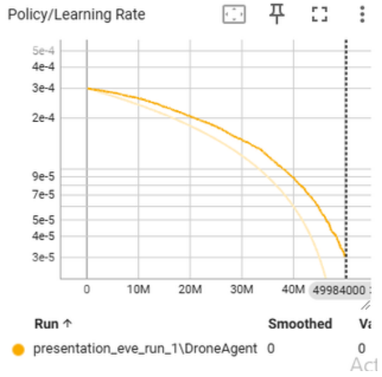
# Glimpses of the Stages in Training





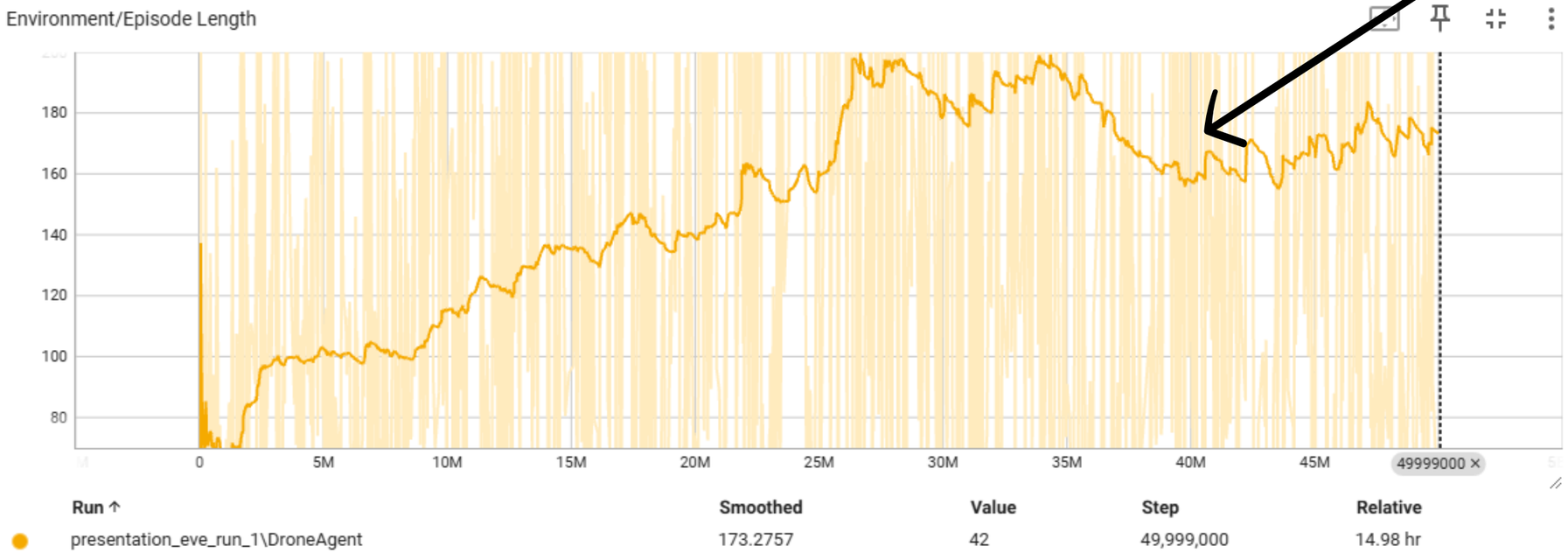
Central Stationary Leader drone trains the Self-Formation property of the Swarm.

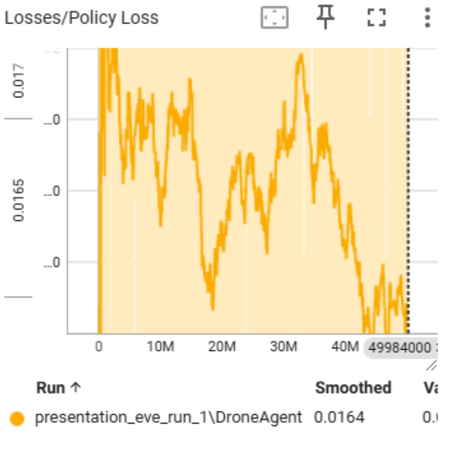



The Reason for Dips here:

Better Policy Discovered and Undertaken

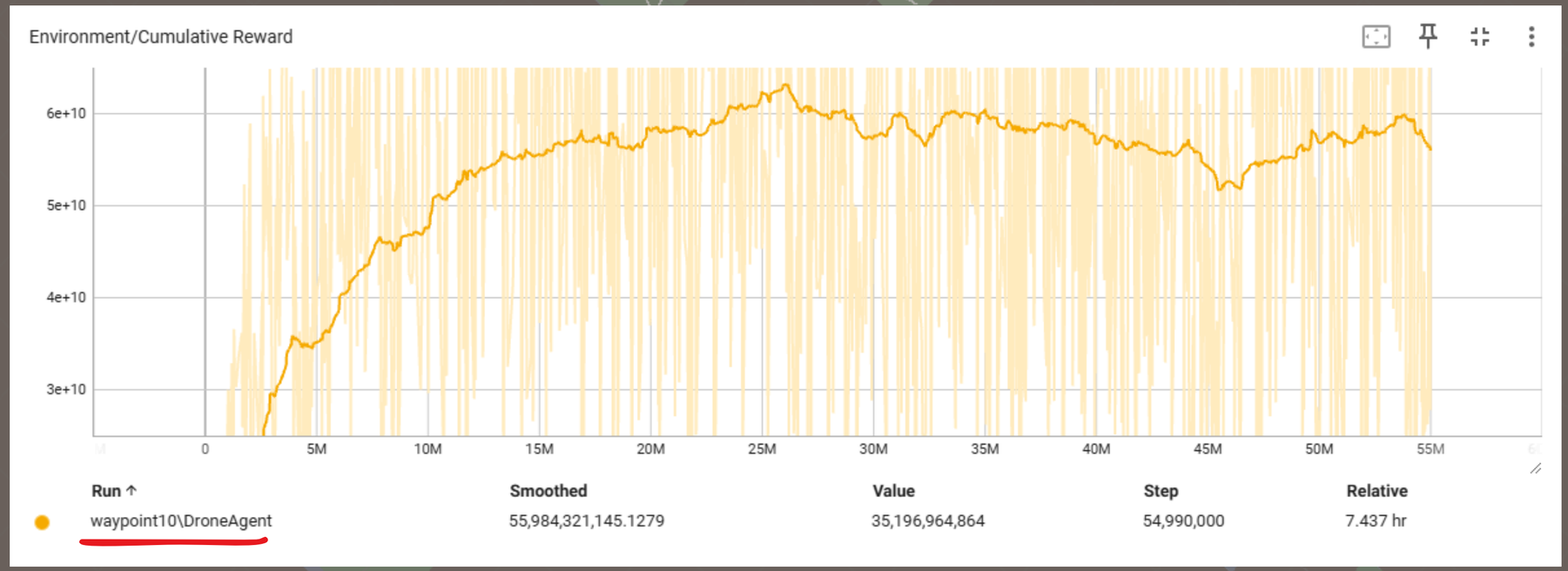


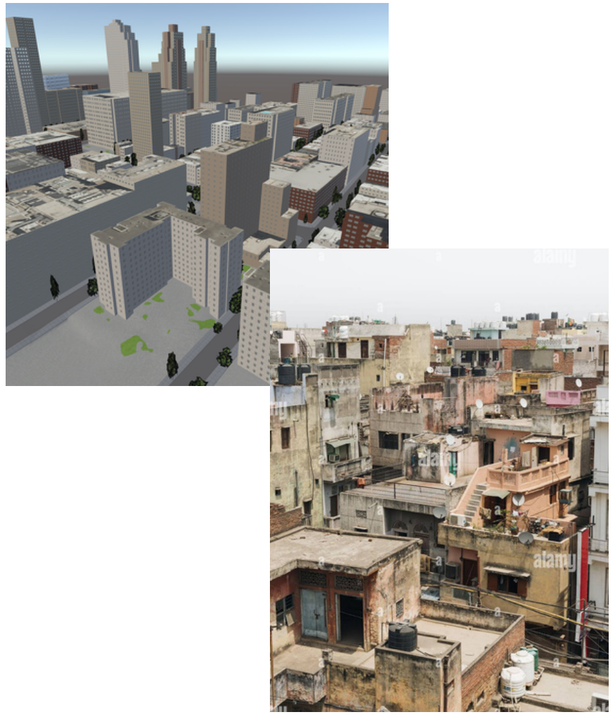
# Strategy

Once this base model is trained, we then introduce waypoints in our Training Environment which will further train the swarm to maintain a cohesive formation while following the Leader Drone in motion

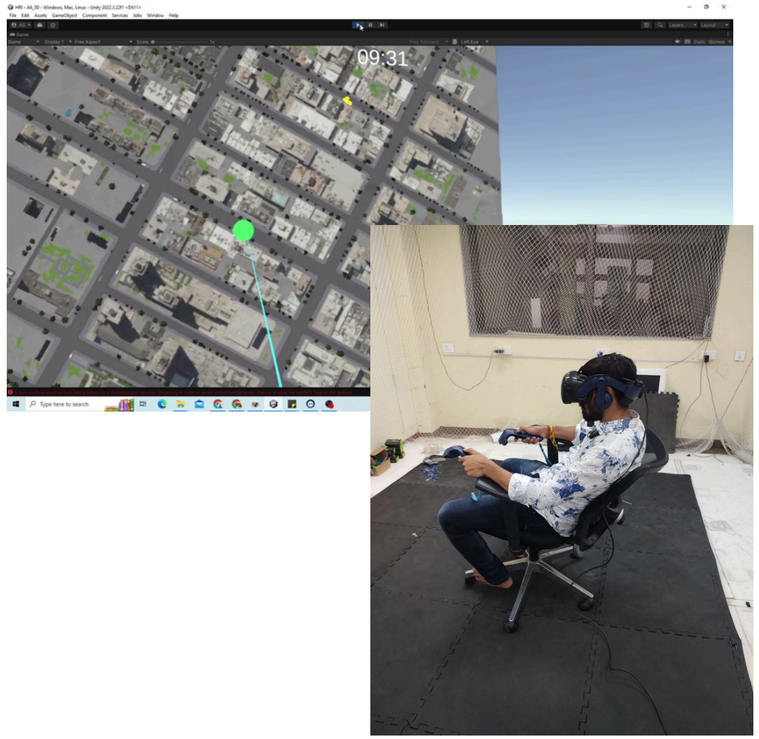
Once, this model trains well, Swarm is robust whilst Leader Drone being User Controlled



# Future Works



**Ensuring Model Robustness**
Adaptability for diverse environments.




**Narrow-Alley Setup & VR Integration**
Enhanced user-controlled leader drone experience.

**CrazyFlies Hardware Implementation**
(If time permits.)

# THANK YOU