Université Paris-Saclay

Université d'Évry

Multi-UAVs consensus formation control

Mission coordination project

Viet Khanh Nguyen 20252887@etud.univ-evry.fr

M2 Smart Autonomous and Aerospace Systems (SAAS)

Department: UFR de Sciences et Technologies

Évry, France, December 2, 2025

Multi-UAVs consensus formation control

Student name: Nguyen Viet Khanh
Student ID: 20252887
Major: M2 Smart Aerospace and Autonomous System - SAAS
Course: SAAS Master Program

1. *Report title:* Multi-UAVs consensus formation control
2. *Report keyword:* Quadcopter control, Formation control, Consensus Algorithm, A* path planning, Pure pursuit.
3. *Time of project received: November 2025*
4. *Time of project deadline:02 December 2025*

Contents

Preface

This project was completed during the first semester of my Master 2 program in Smart Aerospace and Autonomous Systems, shortly after I arrived in France. I have been interested in formation control for a long time, but only recently had the opportunity to study it in depth. Therefore, this project was developed both as a way for me to practice consensus control and to gain a complete understanding of the control pipeline for drones.

Most of the formation control knowledge I acquired comes from the book [1], while the translation–scaling formation control approach is based on [2]. All the control systems I designed were implemented in Python and simulated using MuJoCo. The project code will be made public on my GitHub here.

1 Introduction

Autonomous multi-quadcopter systems are gaining increasing interest due to their potential in surveillance, inspection, search-and-rescue, and cooperative transportation. A core challenge in these applications is enabling multiple aerial robots to navigate safely in complex environments while maintaining a desired formation. This project addresses that challenge by integrating global planning, local tracking, quadcopter control, and multi-agent coordination into a unified simulation framework.

The work combines an A* global planner for generating collision-free paths, a Pure Pursuit tracker for smooth trajectory following, and a hierarchical quadcopter controller that stabilizes both position and attitude. On top of this, a consensus-based formation control strategy allows multiple agents to maneuver cohesively, while collision-avoidance mechanisms ensure safe operation. All components are implemented and tested in a realistic MuJoCo simulation environment.

The aim of the project is to provide a complete workflow—from high-level planning to low-level control—for studying and validating formation-flight strategies for quadcopter teams.

2 Overview: Global Planning and Local Tracking

The navigation stack is divided into two parts: **A* (A-star)** handles the global path generation, and **Pure Pursuit** handles the local execution of that path.

2.1 A* Path Planning: The Global Planner

The A* algorithm serves as the **global path planner**. It finds the optimal sequence of waypoints ($\mathcal{P}$) from the start to the goal in a cost map by minimizing the function

$$f(n) = g(n) + h(n)$$

- **Role:** Generates the desired route (self.waypoints), ensuring the path is shortest (or lowest cost) and avoids known obstacles.
- **Output:** An ordered, static set of waypoints which form the path $\mathcal{P}$.

2.2 Pure Pursuit: The Local Tracker

The **Pure Pursuit** algorithm is a geometric technique used to follow the path $\mathcal{P}$. It steers the vehicle towards a lookahead point ($\mathbf{p}_{\text{filtered}}$) placed a fixed distance (L_d) ahead of the current vehicle position along the path.

2.3 Key Mechanism and Tuning

1. **Lookahead Point Selection:** The method determines the raw target point on $\mathcal{P}$ a distance L_d away from the drone.
2. **Filtering (α):** A crucial addition is the application of a low-pass filter (coefficient self.alpha) to the raw reference point:

   $$\mathbf{p}_{\text{filtered}}(k) = \alpha \mathbf{p}_{\text{filtered}}(k-1) + (1-\alpha)\mathbf{p}_{\text{raw}}(k)$$

   This step ensures the commanded target position shifts smoothly, preventing rapid jumps in the control commands which could destabilize the inner-loop PID controllers.

3 Quadcopter control

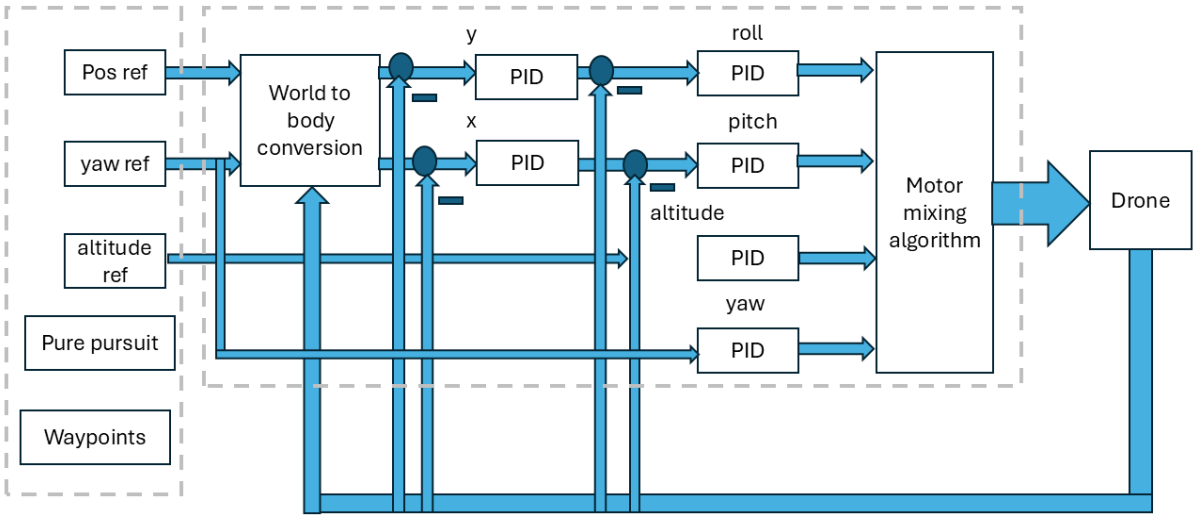


Figure 1: Quadcopter control architecture

3.1 Outer Loop: Position and Guidance Control

The outer loop is responsible for navigating the drone to a target location and following a predefined path. It operates on the world frame coordinates and generates attitude setpoints for the inner loop.

1. **Guidance Inputs:** The system takes references: Pos ref (desired X, Y coordinates), yaw ref (desired heading ψ), and altitude ref (desired Z coordinate). Waypoints from A* and Pure pursuit algorithms handle high-level path generation.
2. **World to Body Conversion:** The X and Y position from body frame is converted to world frame
3. **PID position control:** transform the position error into the commanded attitude angles (desired ϕ_{des} and θ_{des} for roll and pitch, respectively) necessary to tilt the thrust vector:
$$\begin{bmatrix} \phi_{\text{des}} \\ \theta_{\text{des}} \end{bmatrix} = f(e_X, e_Y)$$
4. **Altitude Control:** The altitude error e_Z is fed into a **PID controller** to calculate the required total **Thrust Command** (T_{des}).

3.2 Inner Loop: Attitude and Rate Stabilization

The inner loop operates at a much higher frequency than the outer loop. Its primary function is to stabilize the drone and ensure it accurately tracks the attitude and thrust commands generated by the outer loop.

1. **Inputs:** The setpoints are the desired attitudes (ϕ_{des}, θ_{des}, ψ_{des}) and the desired altitude thrust (T_{des}) from the outer loop.
2. **Attitude Tracking:** Dedicated **PID controllers** are used for the three rotational axes (roll ϕ, pitch θ, yaw ψ). These controllers minimize the attitude error:

$$\begin{cases} e_\phi = \phi_{\text{des}} - \phi_{\text{actual}} \implies \text{Roll Moment Command}(M_\phi) \\ e_\theta = \theta_{\text{des}} - \theta_{\text{actual}} \implies \text{Pitch Moment Command}(M_\theta) \\ e_\psi = \psi_{\text{des}} - \psi_{\text{actual}} \implies \text{Yaw Moment Command}(M_\psi) \end{cases}$$

   The output of these PID controllers is the commanded **torque** (moment) around the respective body axes.

3.3 Motor Mixing and Output

The final control stage synthesizes the four main commands into individual motor signals:

1. **Inputs to Mixer:** Total Thrust Command (T_{des}) and the three Moment Commands (M_ϕ, M_θ, M_ψ).
2. **Motor Mixing Algorithm:** This block calculates the individual speed for each motor (ω_i) required to simultaneously produce the desired total thrust and the three corrective torques. In this report, I use this simple linear mapping:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} T_{\text{des}} \\ M_\phi \\ M_\theta \\ M_\psi \end{bmatrix} = A_{mma} \begin{bmatrix} T_{\text{des}} \\ M_\phi \\ M_\theta \\ M_\psi \end{bmatrix}$$

   where A_{mma} is the mixer matrix derived from the drone's geometry.

3.4 Key Control Elements: PID Controllers

The Proportional-Integral-Derivative (PID) controller is the core component in this architecture. The control output $u(t)$ is defined as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

- K_p (**Proportional**): Provides control action proportional to the current error $e(t)$.
- K_i (**Integral**): Eliminates steady-state errors by accumulating past errors.
- K_d (**Derivative**): Dampens oscillations and improves transient response by reacting to the rate of change of the error.

4 Formation control

This formation control just my summary from the orginal paper [2]. For more detail please read this original paper for more infomation.

4.1 Formation maneuver control problem

Consider a formation of n agents in $\mathbb{R}^d$ $(n \geq 2,\ d \geq 2)$. Let $\mathcal{V} = \{1, \ldots, n\}$. Denote $p_i(t) \in \mathbb{R}^d$ and $v_i(t) \in \mathbb{R}^d$ represent the position and velocity of agent $i \in \mathcal{V}$. We denote the first n_l agents are leaders and the rest n_f agents are followers. Let $\mathcal{V}_l = \{1, \ldots, n_l\}$ and $\mathcal{V}_f = \{1, \ldots, n_f\}$. We modeled the agent as double-integrator

$$\begin{cases} \dot{p}_i(t) &= v_i(t) \\ \dot{v}_i(t) &= u_i \end{cases} \text{where } i \in \mathcal{V}_f$$

$u_i(t) \in \mathbb{R}^d$ is the acceleration input to be designed. We denote $p_l = \left[p_1^T, \ldots, p_{nl}^T\right]^T$, $p_f = \left[p_1^T, \ldots, p_{nf}^T\right]^T$, $v_l = \left[v_1^T, \ldots, v_{nl}^T\right]^T$, $v_f = \left[v_1^T, \ldots, v_{nf}^T\right]^T$. Let $p = \left[p_l^T,\ p_f^T\right]^T$ and $v = \left[v_l^T,\ v_f^T\right]^T$. The underlying information flow among the agents is described by a fixed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the edge set. Then information could flow between agent i and its neighbor set $\mathcal{N}_i$ $(\mathcal{N}_i = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\})$. We assume that the information flow between any two followers is bidirectional. The bearing of agent j relative to agent i is described by the unit vector

$$g_{ij} = \frac{p_j - p_i}{\|p_j - p_i\|} (= -g_{ji})$$

The orthogonal projection matrix of g_{ij} is

$$P_{g_{ij}} = I_d - g_{ij} g_{ij}^T$$

Definition 1. *The target formation denoted by $\mathcal{G}(p^*(t))$ is a formation that satisfies the following constraints for all $t \geq 0$:*

$$\begin{cases} \dfrac{p_j - p_i}{\|p_j - p_i\|} &= g_{ij}^* \forall (i, j) \in \mathcal{E} \\ p_i(t) &= p_i^*(t) \forall i \in \mathcal{V}_l \end{cases}$$

Given the target formation $\mathcal{G}(p^*(t))$ is unique, if the real formation position $p(t)$ converges to the target formation position $p^*(t)$, the desired formation maneuver and formation pattern can be simultaneously achieved. Define the position and velocity errors for the followers as

$$\delta_p(t) = p_f(t) - p_f^*(t), \quad \delta_v(t) = v_f(t) - v_f^*(t)$$

Control objective: Design control laws for the followers to drive $\delta_p(t) \to 0$ and $\delta_v(t) \to 0$ as $t \to \infty$. We define the Bearing Laplacian Matrix $\mathcal{B}(\mathcal{G}(p^*)) \in \mathbb{R}^{dn \times dn}$

$$[\mathcal{B}(\mathcal{G}(p^*))] = \begin{cases} 0_{d\times d}, & i \neq j,\ (i,j) \notin \mathcal{E}, \\ -P_{g^*_{ij}}, & i \neq j,\ (i,j) \in \mathcal{E}, \\ \sum_{k\in\mathcal{N}_i} P_{g^*_{ik}}, & i = j,\ i \in \mathcal{V}, \end{cases}$$

This matrix could be partitioned into

$$\mathcal{B} = \begin{bmatrix} \mathcal{B}_{ll} & \mathcal{B}_{lf} \\ \mathcal{B}_{fl} & \mathcal{B}_{ff} \end{bmatrix}$$

where $\mathcal{B}_{ll} \in \mathbb{R}^{dn_l \times dn_l}$, $\mathcal{B}_{lf} \in \mathbb{R}^{dn_l \times dn_f}$, $\mathcal{B}_{fl} \in \mathbb{R}^{dn_f \times dn_l}$, $\mathcal{B}_{ff} \in \mathbb{R}^{dn_f \times dn_f}$. From the property $\mathcal{B}p^* = 0$, that we yield

$$p^*_f(t) = -\mathcal{B}^{-1}_{ff}\mathcal{B}_{fl}p_l(t), \quad v^*_f(t) = -\mathcal{B}^{-1}_{ff}\mathcal{B}_{fl}v_l(t)$$

To describe the translation and scaling maneuvers, we define the centroid $\bar{p}(p^*(t))$ and the scale $s(p^*(t))$

$$\bar{p}(p^*(t)) = \frac{1}{n}\sum_{i\in\mathcal{V}} p^*_i(t) = \frac{1}{n}(1_n \otimes I_d)^T p^*(t)$$
$$s(p^*(t)) = \sqrt{\frac{1}{n}\sum_{i\in\mathcal{V}} \|p^*_i(t) - \bar{p}(p^*(t))\|^2} = \frac{1}{\sqrt{n}}\|p^*(t) - 1_n \otimes \bar{p}(p^*(t))\|$$

The desired maneuvering dynamics of the centroid and scale of the target formation are given by

$$\dot{\bar{p}}(p^*(t)) = v_c(t), \quad \dot{s}(p^*(t)) = \alpha(t)s(p^*(t))$$

To control both the translation and scaling maneuvers, we need at least two leaders to have the desired formation.

4.2 Formation Maneuvering with Time-Varying Leader Velocity

The proposed control law $u_{consensus}$ for n_f followers is

$$u_i = -K_i^{-1}\sum_{j\in\mathcal{N}_i} P_{g^*_{ij}}\left(k_p(p_i - p_j) + k_v(v_i - v_j) - \dot{v}_j\right)$$

where $K_i = \sum_{j\in\mathcal{N}_i} P_{g^*_{ij}}$. We could prove that under the proposed control law, the control objective $\delta_p(t) \to 0$ and $\delta_v(t) \to 0$ as $t \to \infty$ could be achieved.

Proof. Multiplying K_i on both sides

$$\sum_{j\in\mathcal{N}_i} P_{g^*_{ij}}(\dot{v}_i - \dot{v}_j) = \sum_{j\in\mathcal{N}_i} P_{g^*_{ij}}\left(k_p(p_i - p_j) + k_v(v_i - v_j) - \dot{v}_j\right)$$

rewritten this equation in matrix form

$$\mathcal{B}_{ff}\dot{v}_f + \mathcal{B}_{fl}\dot{v}_l = -k_p\left(\mathcal{B}_{ff}p_f + \mathcal{B}_{fl}p_l\right) - k_v\left(\mathcal{B}_{ff}v_f + \mathcal{B}_{fl}v_l\right)$$

Since

$$\mathcal{B}_{ff}\delta_p(t) = \mathcal{B}_{ff}\left(p_f(t) - p_f^*(t)\right) = \mathcal{B}_{ff}p_f(t) + \mathcal{B}_{fl}p_l(t)$$
$$\mathcal{B}_{ff}\delta_v(t) = \mathcal{B}_{ff}\left(v_f(t) - v_f^*(t)\right) = \mathcal{B}_{ff}v_f(t) + \mathcal{B}_{fl}v_l(t)$$
$$\mathcal{B}_{ff}\dot{\delta}_v(t) = \mathcal{B}_{ff}\left(\dot{v}_f(t) - \dot{v}_f^*(t)\right) = \mathcal{B}_{ff}\dot{v}_f(t) + \mathcal{B}_{fl}\dot{v}_l(t)$$

Therefore we have

$$\mathcal{B}_{ff}\dot{\delta}_v(t) = -k_p\mathcal{B}_{ff}\delta_p(t) - k_v\mathcal{B}_{ff}\delta_v$$

which equivalent

$$\dot{\delta}_v(t) = -k_p\delta_p(t) - k_v\delta_v$$

rewrite the whole error of the system in matrix form

$$\begin{bmatrix}\dot{\delta}_p\\ \dot{\delta}_v\end{bmatrix} = \begin{bmatrix}0 & I\\ -k_pI & -k_vI\end{bmatrix}\begin{bmatrix}\delta_p\\ \delta_v\end{bmatrix}$$

The error matrix of the system is Hurwitz, with the eigenvalue $\lambda = -k_v \pm \sqrt{k_v^2 - 4k_p}$. Then the global and exponential convergence result is proven. □

4.3 Multi-Agent Collision Avoidance

I will design a potential field to avoid the collision between agents. Denote $\beta_{ij}(p(t))$ is the potential function which satisfy

$$\begin{cases}\beta_{ij}(p(t)) \in [0,1]\\ \beta_{ij}(p(t)) = 0,\ \|p_i - p_j\| = 0\\ \beta_{ij}(p(t)) = 1,\ \|p_i - p_j\| \geq d\end{cases}$$

In this project I choose

$$\beta_{ij}(p(t)) = \left(1 - \mu\frac{\left(\|p_i - p_j\|^2 - d^2\right)^2}{1 + \left(\|p_i - p_j\|^2 - d^2\right)^2}\right)^{\rho}$$

where $\mu = \frac{1+d^4}{d^4}$ and $\rho = \frac{1}{2}\left(1 - \mathrm{sgn}(\|p_i - p_j\|^2 - d^2)\right)$. Choose $\beta = \prod_{i,j\in\mathcal{V},i\neq j}\beta_{ij}(p(t))$. Then we choose the potential field for collision avoidance is

$$\Phi_a(p(t)) = ln(\beta(p(t))) = \sum_{i,j\in\mathcal{V},i\neq j}\beta_{ij}(p(t))$$

Then the control law u_{avoid} is given as

$$u = \nabla_p\Phi_a(p(t))$$

where the control law for the agent i is

$$u_i = \sum_{j\in\mathcal{N}_i}\frac{\dot{\beta}_{ij}}{\beta_{ij}} = \sum_{j\in\mathcal{N}_i}\frac{-4\mu\left(\|p_i - p_j\|^2 - d^2\right)\left(p_i(t) - p_j(t)\right)}{1 + \left(\|p_i - p_j\|^2 - d^2\right)^2 - \mu\left(\|p_i - p_j\|^2 - d^2\right)^2}$$

Final consensus control law

The final control law for agent i is the combination between the Formation Control Law and the Collision Avoidance

$$u_i = u_{consensus,i} + u_{avoid,i}$$

This final control law ensure the multi-agent system follow the desired formation maneuver while avoid collision.

5 Mujoco simulation

All of my control algorithm in this project is written in Python. In addition, I choose Mujoco to validate the proposed control architecture. There are plenty of open source robot model in Mujoco. In this project I choose the Skydio X2 model, which the original source could be founded here.

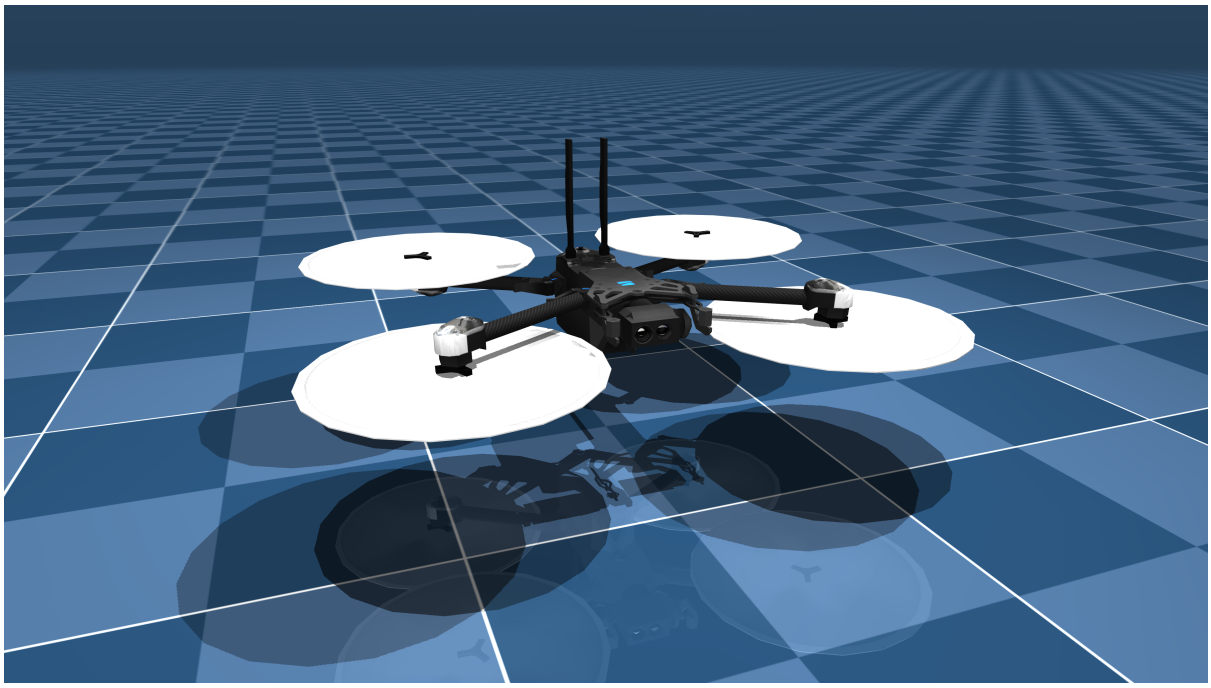
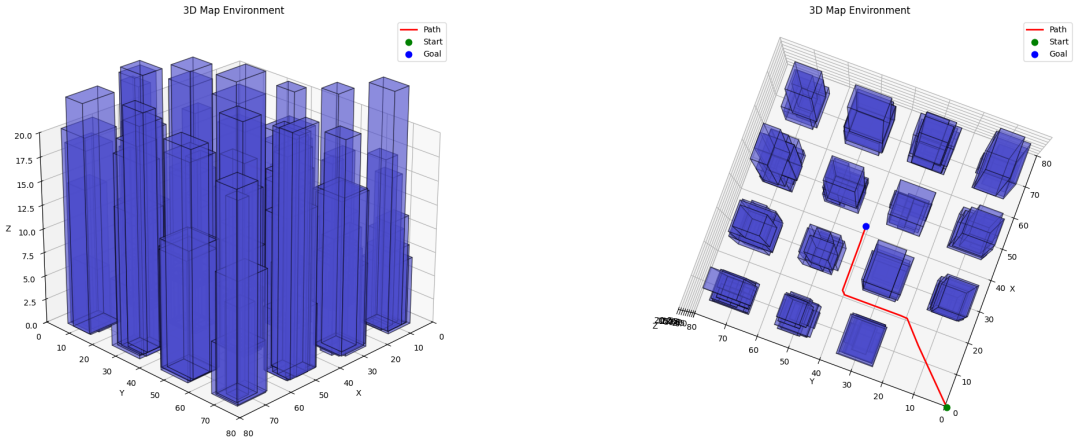


Figure 2: Skydio X2

Simulation scenario: After take off, the UAVs must go from the start position to the predefined goal position, then landing after. Both take off and landing operation must be taken into account. In the mean time, while our UAVs must form a icosahedron in 3D space, and scaling down to avoid the static buildings (obstacles). The safty criteria must be met which no drone is allow to collide with each other.

5.1 Map

I will build a xml file environment to simulate multi-UAVs in a city-like environment.



(a) City map (b) A* path

Figure 3: City map and computed A* path

The starting point is $(0, 0, 5)$ and the goal is $(40, 40, 15)$. I inflate the obstacles so that we could have a safer waypoints away from the obstacle.

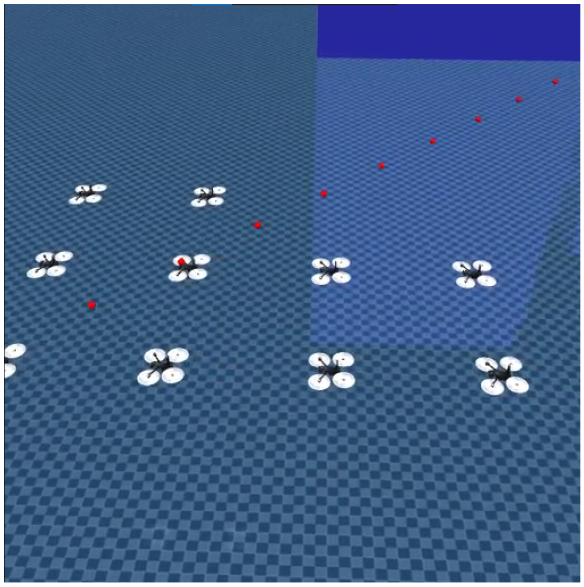
5.2 Mission Stages

The provided code defines a **Finite-State Machine (FSM)** that dictates the operational sequence for a multi-UAV mission. This architecture ensures complex, sequential, and safety-critical tasks—such as formation changes and path tracking—are executed only when strict control and state criteria are met. The core control input for the horizontal axes is the velocity reference - v_{ref}, which is primarily derived from `formation_controller`.

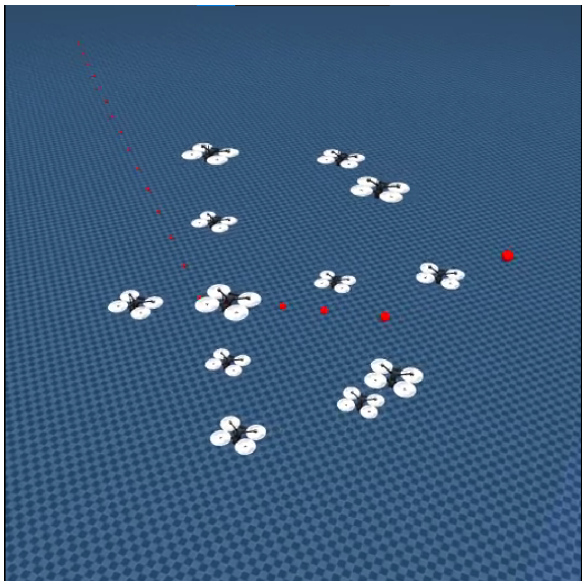
```
switch operation_mode:
        case "take_off":
                v_ref = 0
                altitude_ref = 5
                if np.all(all_uav_height < 0.05):
                        operation_mode = "formation_icosahedron"

        case "formation_icosahedron":
                v_ref = formation_controller
                altitude_ref += v_ref(z)*dt
                if e_bearing_norm < e_bearing_tol:
                        operation_mode = "cruise"

        case "cruise":
                a_ref = formation_controller
                v_ref += a_ref*dt
```

```
                altitude_ref += v_ref(z)*dt
                if (path_tracking.goal_flag) and
                (e_bearing_norm < e_bearing_tol):
                        operation_mode = "formation_rectangle"

        case "formation_rectangle":
                v_ref = formation_controller
                altitude_ref = \int_{t_3}^{t} v_ref_z dt
                if e_bearing_norm < e_bearing_tol:
                        operation_mode = "landing"

        case "landing":
                v_ref = 0
                altitude_ref = 0
```

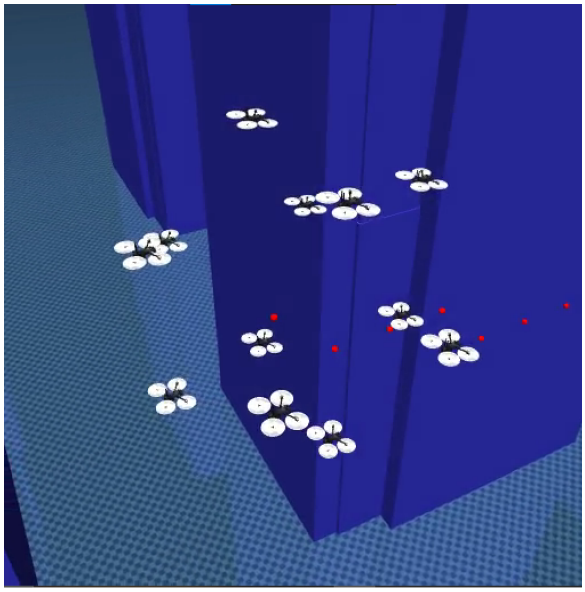
5.3 Result



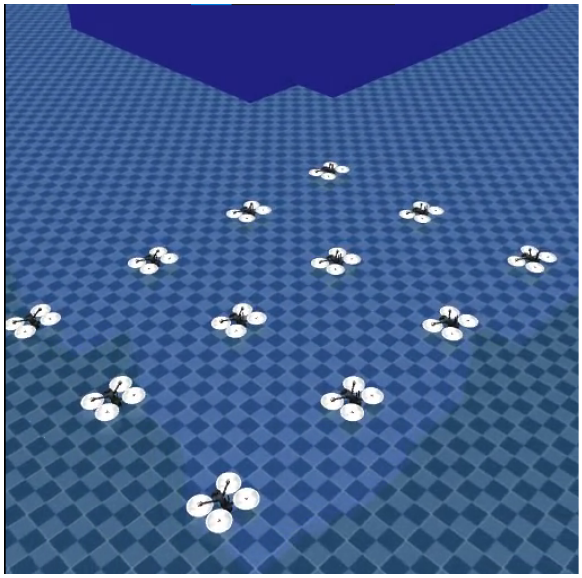
(a) Operation: Take off



(b) Operation: Cruise



(c) Operation: Cruise - Goal reached



(d) Operation: Landing

Figure 4: Visualization of Mujoco simulation results

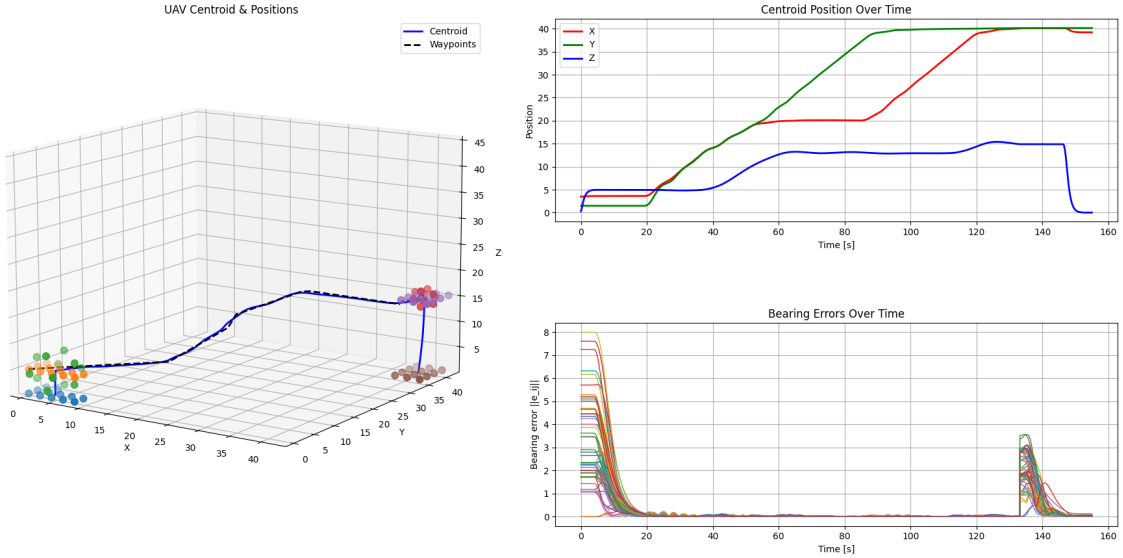


Figure 5: Multi-UAVs trajectory

The figures illustrate the performance of a control system guiding a UAV centroid along a desired trajectory defined by waypoints.

Trajectory Adherence (3D Plot): The **Centroid** of my multi-agents system (blue dashed line) successfully tracks the **Waypoints** (black dashed line). The system executes a smooth, multi-stage maneuver involving changes in all three spatial coordinates (X, Y, Z).

Position Control Over Time: It take 155 seconds for the system from the take-off mode to the landing mode. When reaching the goal $(40, 40, 15)$, all drones start landing on the rectangle area around $(40, 40, 0)$. That is the reason for the centroid slightly driff to $(38, 40, 0)$

Control Stability (Bearing Errors): In the take-off mode, a large initial errors (up to ~ 8 units) are observed during the startup and initial acceleration phase, which is typical for control systems. The control system achieves **near − zero** bearing error (Error ≈ 0), showing excellent stability and tracking during the main mission phase in cruise mode. A spike in error occur when the system switching to landing mode, but exponentially converge to 0 afterward.

6 Conclusion and Deliverables

This project implemented a complete pipeline for autonomous multi-quadcopter formation control, combining global A* path planning, Pure Pursuit local tracking, formation control to low level quadcopter thrust control. The proposed frame work is validated through realistic simulation in MuJoCo. The integrated system enabled each drone to follow a safe global path, accurately track trajectories through well-tuned position and attitude controllers, and maintain coordinated formation and prevent collision avoidance using consensus-based laws. The simulation results confirm that the proposed framework performs reliably and can execute complex formation maneuvers in city environments. Overall, the project provides a solid foundation for future work on more advanced control strategies and real-world deployment.

References

[1] Trinh Hoang Minh, Nguyen Minh Hieu, "Dieu khien he da tac tu," 2021.

[2] S. Zhao and D. Zelazo, "Translational and scaling formation maneuver control via a bearing-based approach," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 3, pp. 429–438, 2017.