Hierarchical System for Safe Adaptive Multi-Drone Control

Zhi Li
Computer Science Department
University of California, Los Angeles
mikelili@ucla.edu

Yuchen Liu
Electrical Engineering Department
University of California, Los Angeles
yuchenliu02@ucla.edu

Yufei Song
Electrical Engineering Department
University of California, Los Angeles
yufeisong@ucla.edu

Abstract

Coordinating heterogeneous teams of quad- and fixed-wing drones in cluttered, a prior unknown airspace demands three properties that are rarely met simultaneously: global task reasoning, formally safe local maneuvers, and real-time execution on embedded hardware. Classical geometric or consensus-based planners offer safety guarantees but rely on pre-built maps and struggle to re-plan efficiently, whereas end-to-end neural policies adapt well yet sacrifice formal safety outside their training domain. We introduce the first vision–language–control hierarchy that fuses the broad world knowledge of a large vision–language model (VLM) with the sample efficiency and formal guarantees of learning base control. Each drone narrates its live multi-modal observations in natural language, receives from the VLM a map-free, goal-directed flight script, and executes that script through a reinforcement-learning (RL) navigation layer regulated by control-barrier-function shields. The architecture contains graded fall-back modes that degrade gracefully from full autonomy (VLM + vision + RL) to a perception-free emergency loop, ensuring provably collision-free operation under sensor dropout, API failure, or sub-Watt compute budgets.

1. Introduction

Coordinating a fleet of autonomous drones in the wild is an open frontier in robotics research. Mission planners must juggle global task allocation, local collision avoidance, tight energy budgets, and real-time execution on heterogeneous airframes, while facing sensor outages, bandwidth limits, and incomplete maps. Classical centralized planners supply crisp guarantees, yet they collapse under communication dropouts and scale poorly beyond a limited number of agents. Fully distributed consensus schemes mitigate the bandwidth bottleneck, but their myopic horizon limits adaptability in cluttered, dynamic scenes. Meanwhile, end-to-end neural controllers excel in familiar settings but lack the certified safety demanded by disaster-response, inspection, and defense applications.

Recent breakthroughs in large language models (LLMs) and vision–language models (VLMs) hint at a new direction. By grounding textual reasoning in rich visual context, VLMs can describe complex 3-D scenes, infer latent affordances, and draft step-wise action plans. Yet their inference time and computational footprint preclude direct, low-level flight control. What is needed is a principled hierarchy that fuses the broad situational awareness of VLMs with the reflex-grade responsiveness of classical and learning-based controllers.

We present a hybrid architecture that delivers three advances to multi-drone autonomy:

1. **Hierarchical system** A high-level VLM converts live, multi-modal observations from each drone into a synthesized map-free sub-goal scripts that direct low-level learning-based model to generalize control of multi-drone fleet across simulation, indoor mock-ups, and natural outdoor environments.
2. **Variable-rate control stack.** A variable-interval scheme decouples VLM reasoning from a low-latency real-time navigation layer built on reinforcement learning, permitting flight on off-the-shelf learning-based controllers while still exploiting the VLM's global foresight.
3. **Graceful safety fall-backs.** A safety monitor demotes the system through successively fallback to simpler modes like VLM to the rule-based planner and learning-based model to the feedback-based control loop, and finally, a perception-free emergency collision avoidance

and airframe integrity eliminating model instability.

Our results show that state-of-the-art foundation models can be systematically woven into certified, real-time robotic stacks, paving the way for large-scale, adaptive drone deployments in safety-critical domains.

2. Related Work

Multi-Agent Path Planning: Classical approaches to multi-agent coordination include centralized methods such as A* variants [3] and distributed algorithms like ORCA [5]. Recent work has explored learning-based approaches using multi-agent reinforcement learning [2], but these methods often lack interpretability and struggle with generalization to unseen scenarios.

Vision-Language Models in Robotics: The integration of VLMs in robotics has gained significant attention, with applications ranging from manipulation [1] to navigation [4]. However, most existing work focuses on single-agent scenarios and lacks the real-time constraints and safety considerations required for multi-drone coordination.

Hybrid Control Systems: The combination of learning-based and classical control approaches has been explored in various contexts [6]. Our work extends this paradigm by introducing VLM-based planning as a third layer in the control hierarchy, creating a novel three-tier architecture.

3. Simulation Platform

3.1. Physics-Based Simulation

Our framework is built upon the Genesis simulation platform, which provides high-fidelity physics simulation for multi-drone environments. The physics engine models quadrotor dynamics using the Newton-Euler formulation:

$$\mathbf{F} = m\mathbf{a}, \quad \boldsymbol{\tau} = \mathbf{I}\boldsymbol{\alpha} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega})$$

where $\mathbf{F}$ and $\boldsymbol{\tau}$ represent the total force and torque vectors, m is the drone mass, $\mathbf{I}$ is the inertia tensor, and $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ are the angular velocity and acceleration vectors.

The simulation incorporates realistic aerodynamic effects, including propeller thrust and drag models:

$$T_i = k_t\omega_i^2, \quad Q_i = k_q\omega_i^2$$

where T_i and Q_i are the thrust and torque generated by propeller i, ω_i is the propeller angular velocity, and k_t, k_q are thrust and torque coefficients.

3.2. Differentiable Simulation

Our simulation platform is differentiable, enabling end-to-end gradient flow from loss functions all the way back to action inputs. This capability supports direct optimization of control policies through backpropagation, and gradient-based planning with physics constraints.

The differentiable physics engine maintains computational graphs throughout the simulation, allowing gradients to flow through contact forces, collision dynamics, and environmental interactions. This enables sophisticated learning algorithms that can reason about physical constraints and optimize long-horizon behaviors.

3.3. Photo-realistic Ray Tracing Rendering

The simulation platform supports ray tracing for photo-realistic visual rendering. The rendering system supports:

- **Global Illumination**: Accurate light transport simulation with multiple bounces
- **Material Properties**: Realistic surface reflectance and scattering models
- **Environmental Effects**: Dynamic lighting, shadows, and atmospheric effects

This high-fidelity visual simulation enables the VLM to process realistic imagery that closely approximates real-world deployment scenarios, improving the transferability of learned behaviors to physical systems.

4. Task Design

We formulate the multi-drone coordination problem as a landmark occupation task that requires both spatial reasoning and collision avoidance. Given n drones $\{D_1, D_2, \ldots, D_n\}$ and n landmark positions $\{L_1, L_2, \ldots, L_n\}$ in 3D space, the objective is to achieve a configuration where each landmark is occupied by exactly one drone.

4.1. Problem Formulation

We consider a multi-agent control task involving n quadcopters (drones), each of which must reach a distinct target location in 3D space. Let $\{D_i\}_{i=1}^n$ denote the positions of the n drones, and $\{L_j\}_{j=1}^n$ denote the n predefined target locations.

Each drone $D_i \in \mathbb{R}^3$ must reach exactly one target $L_j \in \mathbb{R}^3$, such that the Euclidean distance between the drone and the target is less than a specified threshold $\epsilon > 0$. The task is considered successful only if all n targets are uniquely occupied by different drones within the distance margin.

$$R_t = \Big\{ \forall j \in \{1, \ldots, n\} : |\{i : d(D_i, L_j) < \epsilon\}| = 1$$

The task is deemed successful at time t if and only if each target location L_j is occupied by exactly one drone within distance ϵ. This formulation naturally enforces both full coverage and collision-free assignment in the final state. It is typically used as a hard success criterion in multi-agent evaluation settings.

4.2. Constraint Formulation

The task incorporates several constraints to ensure safe and realistic operation:

- **Collision Avoidance**: $\forall i, j : d(D_i, D_j) \geq d_{\text{min}}$ where d_{min} is the minimum safe separation
- **Workspace Bounds**: $\mathbf{p}_i \in \mathcal{W}$ where $\mathcal{W}$ defines the operational workspace
- **Dynamic Limits**: Velocity and acceleration constraints based on physical drone capabilities

This task design provides a challenging testbed for evaluating multi-agent coordination algorithms while maintaining clear success criteria and safety requirements.

5. Vision–Language–Control (VLC) Hierarchical System

5.1. System Architecture

We present a novel hierarchical multi-drone vision-language-control coordination framework that integrates Vision Language Models (VLMs) for intelligent path planning and collision avoidance in dynamic environments. Our system architecture, illustrated in Figure 1, comprises three key components: (1) a multi-modal perception module, (2) a VLM-based planning engine with conversation history, and (3) a hybrid control system supporting both reinforcement learning and classical PID controllers.

The system operates on a discrete timestep basis, where at each step t, the environment state $\mathcal{S}_t = \{s_1^t, s_2^t, \ldots, s_n^t\}$ represents the positions, velocities, and orientations of n drones. The VLM receives this state information along with optional visual observations $\mathcal{V}_t$ captured from strategically positioned cameras, and generates waypoint assignments $\mathcal{W}_t = \{w_1^t, w_2^t, \ldots, w_n^t\}$ and target assignments $\mathcal{A}_t = \{a_1^t, a_2^t, \ldots, a_n^t\}$.

5.2. VLM Integration and Conversation History

5.2.1. Multi-Modal VLM Interface

Our system supports both vision-enabled VLM and text-only LLM modes, enabling flexible deployment based on computational constraints and task requirements. The VLM interface processes structured JSON input containing:

- **State Vector**: The full state of drone i at time t is given by:
$$\mathbf{s}_i^t = [p_i^t, v_i^t, \theta_i^t]$$
where $p_i^t \in \mathbb{R}^3$ is the position vector in (x, y, z), $v_i^t \in \mathbb{R}^3$ is the linear velocity in $(\dot{x}, \dot{y}, \dot{z})$, and $\theta_i^t \in SO(3)$ represents the attitude (orientation), which is decomposed into Euler angles (ϕ, θ, ψ) corresponding to roll, pitch, and yaw.
- **Target Configuration**: $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ representing m target locations in 3D space.
- **Collision Context**: Pairwise distance matrix $\mathbf{D}_t \in \mathbb{R}^{n \times n}$ and collision risk indicators.
- **Task Progress**: Completion status for each target and drone-target assignment history.

5.2.2. Conversation History Mechanism

We integrate conversation history to maintain temporal context across planning cycles. We maintain a sliding window of the last H interactions, where each interaction I_k contains:

$$I_k = \langle \mathcal{S}_k, \mathcal{V}_k, \mathcal{W}_k, \mathcal{A}_k, R_k \rangle$$

where R_k represents the reasoning provided by the VLM. This history enables the VLM to: (1) maintain consistency in long-term planning strategies, (2) learn from previous collision resolution attempts, (3) adapt planning intervals based on scenario complexity, and (4) provide contextually aware reasoning for current decisions.

We use only latest image in vision model, and full text history for both text-only mode and vision model.

5.2.3. VLM Implementation Details

We use OpenAI's chat-completion API with flexible support for both vision-enabled and text-only inputs. The core of our implementation is a Python function that encapsulates the interaction with the propriety models API under OpenAI's framework, providing robust context management, error handling, and structured output parsing.

Key features of our VLM implementation include:

- **Model Selection and API Authentication:** The system supports OpenAI models such as `gpt-4o`, `o4-mini`, and Google Gemini models depending on configuration. API keys are securely loaded from environment variables, ensuring separation of credentials from code.
- **Multi-Modal Input Handling:** For vision-enabled operation, images are read from disk, base64-encoded, and included as embedded image URLs in the prompt. For text-only operation, the prompt contains only structured JSON state inputs, enabling deployment in constrained environments without image support.
- **Rich System Prompt Engineering:** The system prompt instructs the VLM as an expert multi-agent path planner with explicit requirements such as collision avoidance, target reassignment, waypoint generation for all drones, and reasoning output. This prompt enforces output consistency and adherence to a predefined JSON schema.
- **Conversation History Integration:** To maintain temporal context, the last few planning interactions (typically five) are appended to the prompt. Each interaction includes the previous input state and the VLM's generated output. This enables the VLM to learn from past decisions, improve collision resolution, and adapt planning intervals dynamically.

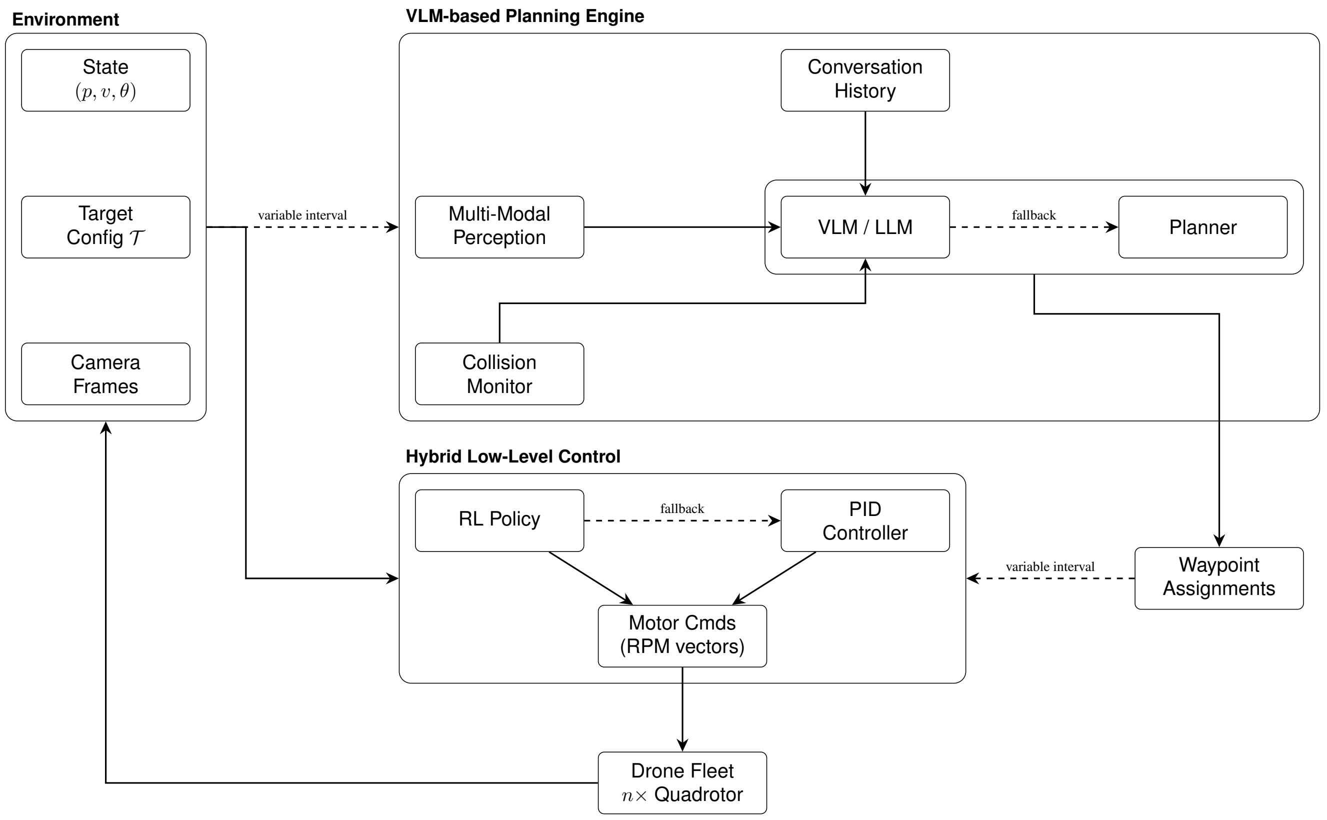

Figure 1. Vision–Language–Control (VLC) Hierarchical System

- **Collision Risk Highlighting:** When collision risks are detected in the input state, an urgent alert message is included in the prompt, detailing which drone pairs are dangerously close. This primes the model to prioritize immediate collision avoidance in its waypoint planning.
- **Robust API Request and Retry Logic:** The implementation includes a retry mechanism with exponential backoff to handle transient API failures gracefully, enhancing reliability in real-world deployments.
- **Structured JSON Parsing:** The VLM's textual output is parsed by extracting the first JSON object from the completion text. This object contains `assignments` mapping drones to targets, `waypoints` specifying next positions, the number of `n_steps` the plan remains valid, and a detailed `reasoning` explanation.

5.3. Adaptive Collision Detection and Avoidance

5.3.1. Real-Time Collision Monitoring

Our system implements continuous collision risk assessment by monitoring pairwise distances between all drones. For each pair (i, j) where $i \neq j$, we compute:

$$d_{ij}^t = \|\mathbf{p}_i^t - \mathbf{p}_j^t\|_2$$

A collision risk is detected when $d_{ij}^t < d_{\text{min}}$, where d_{min} is the minimum safe separation distance.

5.3.2. Intelligent Replanning Strategy

Traditional reactive systems call the planner at every collision detection, leading to excessive computational overhead. We introduce an adaptive replanning strategy that balances responsiveness with efficiency:

Algorithm 1 Adaptive VLM Replanning

```
1: Input: Current step $t$, collision detected $C_t$, last collision call step $t_{\text{last}}$
2: Parameters: Collision interval $\tau_c = 20$, regular interval $\tau_r$
3: if $C_t$ and $(t - t_{\text{last}} \geq \tau_c$ or $t_{\text{last}} = -1)$ then
4:     Call VLM with collision priority
5:     $t_{\text{last}} \leftarrow t$
6: else if $C_t$ and $t - t_{\text{last}} < \tau_c$ then
7:     Skip VLM call, log collision status
8: else if $t \bmod \tau_r = 0$ then
9:     Call VLM for regular replanning
10: end if
```

This approach provides immediate response to initial collision detection while preventing rapid successive calls during sustained collision scenarios.

5.4. Hybrid Control Architecture

Our system enables seamless integration of both classical and learning-based control paradigms, specifically a cascaded PID controller and a reinforcement learning (RL) policy. Both approaches operate on the drone's state parameters as input observations and generate control signals for the four independent rotors. The Genesis simulation environment then advances the physics-based simulation by one step, updating the drone's state accordingly. This updated state serves as the next input to the control module, enabling closed-loop feedback for both PID- and RL-driven control.

The system automatically selects the control mode based on availability: RL control when trained checkpoints are provided, with graceful fallback to PID control otherwise.

5.4.1. Cascaded PID Control Architecture

We implement a cascaded PID controller to stabilize the quadcopter and track a desired target point in 3D space. The control architecture leverages three layers of PID control: position, velocity, and attitude, each processing a subset of the drone's state feedback.

The controller uses the following hierarchy of PID controllers:

- **Position Controllers:** Compute the desired velocities based on the position error:

$$e_p^t = p_{\text{target}}^t - p_i^t, \quad v_{\text{des}}^t = \text{PID}_{\text{pos}}(e_p^t)$$

- **Velocity Controllers:** Generate thrust and planar corrections based on velocity error:

$$e_v^t = v_{\text{des}}^t - v_i^t, \quad [u_x, u_y, u_z] = \text{PID}_{\text{vel}}(e_v^t)$$

  where u_z is interpreted as the desired thrust along the vertical axis.

- **Attitude Controllers:** Stabilize the orientation to maintain level flight (zero roll and pitch, fixed yaw):

$$e_\theta^t = \theta_{\text{target}}^t - \theta_i^t = [0,0,0] - \theta_i^t, \quad [\tau_\phi, \tau_\theta, \tau_\psi] = \text{PID}_{\text{att}}(e_\theta^t)$$

The individual control signals from the cascaded PID blocks are combined through a mixer to generate RPM commands for the four rotors. The final motor inputs are computed as:

$$\begin{aligned} M_1 &= \text{base} + (u_z - \tau_\phi - \tau_\theta - \tau_\psi - u_x + u_y) \\ M_2 &= \text{base} + (u_z - \tau_\phi + \tau_\theta + \tau_\psi + u_x + u_y) \\ M_3 &= \text{base} + (u_z + \tau_\phi + \tau_\theta - \tau_\psi + u_x - u_y) \\ M_4 &= \text{base} + (u_z + \tau_\phi - \tau_\theta + \tau_\psi - u_x - u_y) \end{aligned}$$

where base is a nominal hover RPM, u_z is the collective thrust command, τ_ϕ, τ_θ, and τ_ψ are the corrective torques from roll, pitch, and yaw controllers, and u_x and u_y are the planar velocity corrections.

This configuration enables stable control of the quadcopter's trajectory toward a target point. However, the system remains underactuated, as the desired roll, pitch, and yaw angles are fixed to zero. Consequently, the rotational degrees of freedom are not exploited for more agile or dynamic maneuvers. Leveraging these additional degrees of freedom for agile control would require embedding the attitude PID controllers deeper into the feedback cascade, significantly increasing the complexity of the control architecture and making it difficult to ensure overall system stability. Furthermore, the current structure already involves tuning 27 PID gains; introducing additional fine-grained control over attitude would be both impractical and prone to instability. To overcome these limitations and enable more expressive and adaptive behavior, we resort to reinforcement learning, which allows for end-to-end learning of agile control strategies through interaction and reward-driven policy optimization.

5.4.2. Reinforcement Learning Policy

We define a policy $\pi_\theta : \mathbb{R}^{17} \to \mathbb{R}^4$ that maps a 17-dimensional "observation" vector to a 4-dimensional continuous action space. The output corresponds to the normalized thrust levels applied to each of the drone's four rotors. Actions are scaled and clipped to produce rotor RPMs within a feasible range.

At inference time, the observation vector $\mathbf{o}_t$ at timestep t is composed of $\mathbf{p}_i^t \in \mathbb{R}^3$, $\mathbf{v}_i^t \in \mathbb{R}^3$, $\boldsymbol{\omega}_i^t \in \mathbb{R}^3$ – angular velocity in the local body frame, $\mathbf{q}_i^t \in \mathbb{H}$ – unit quaternion representing drone orientation and last action $\mathbf{a}_{t-1} \in \mathbb{R}^4$. The policy outputs an action $\mathbf{a}_t = \pi_\theta(\mathbf{o}_t)$, which is used to generate thrust commands:

$$\text{RPM}_i = \text{base\_rpm} \cdot (1 + 0.8 \cdot a_{t,i}), \quad i = 1,2,3,4$$

The reward function r_t is a weighted sum of several task-specific reward components. Each is designed to promote desirable behavior in hovering, such as target convergence, stability, and control smoothness.

- **Target Progress Reward**: Encourages movement toward the target by rewarding reduction in squared distance between drone and target $\mathbf{r}_t$:

$$r_t^{\text{target}} = \|\mathbf{r}_{t-1}\|^2 - \|\mathbf{r}_t\|^2$$

- **Smoothness Penalty**: Discourages abrupt changes in action to promote smoother control:

$$r_t^{\text{smooth}} = -\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$$

- **Yaw Stabilization Reward**: Uses the exponential of yaw angle in radians to penalize deviation from zero yaw:

$$r_t^{\text{yaw}} = \exp(\lambda_{\text{yaw}} \cdot |\psi_t|), \quad \lambda_{\text{yaw}} < 0$$

- **Angular Velocity Penalty**: Penalizes overall spinning motion, helping stabilize the drone in place:

$$r_t^{\text{angular}} = -\left\|\frac{\boldsymbol{\omega}_t}{\pi}\right\|$$

- **Crash Penalty**: Applies a large penalty when the drone violates termination conditions, including pitch or roll limits, crash, or low altitude:

$$r_t^{\text{crash}} = -\mathbf{1}_{\text{crash\_condition}}$$

We introduce the yaw stabilization penalty to encourage the drone to face directly toward the target at all times. This effectively constrains the heading degree of freedom, reducing the burden on the policy network to explore unnecessary yaw rotations. By reducing the effective control dimensionality, the actor network can concentrate on learning agile and efficient motion strategies using the remaining degrees of freedom. This simplification is critical in high-dimensional control settings, where redundant orientation freedoms can hinder convergence and lead to unstable learning.

We adopt an actor-critic algorithm that uses two separate neural networks. The actor network $\pi_\theta(\mathbf{o}_t)$ approximates the policy function, which maps observations to actions, while the critic network $V_\phi(\mathbf{o}_t)$ estimates the state-value function to evaluate the quality of states under the current policy. Both networks are implemented as multi-layer perceptrons (MLPs) with hidden layers of sizes [128, 128], using Tanh as the activation function. The actor network outputs the mean of a Gaussian distribution over actions, and the standard deviation can either be fixed or learned during training.

We train the policy using the Proximal Policy Optimization (PPO) algorithm, which stabilizes learning by limiting the size of policy updates.

5.4.3. Mixed-Mode Operations and Fallback Mechanisms

Our architecture implements a sophisticated fallback hierarchy that ensures robust operation across diverse deployment scenarios. The system operates according to the following decision tree:

Mixed-Mode Operational States:

1. **Full-Capability Mode**: VLM + RL control with visual scene understanding
2. **Cost-Optimized Mode**: Text-only LLM + PID control for resource-constrained environments
3. **Degraded VLM Mode**: VLM planning with fallback to greedy assignment on API failures
4. **Autonomous Mode**: Pure RL or PID control without VLM assistance
5. **Emergency Mode**: Basic PID control with collision avoidance when all other systems fail

Algorithm 2 Control Mode Selection and Fallback

```
1: Input: RL checkpoint availability $C_{RL}$, VLM enabled $V_{LLM}$, API status $S_{API}$
2: if $C_{RL}$ = true and checkpoint loads successfully then
3:     control_mode ← RL
4: else
5:     control_mode ← PID
6:     Log fallback reason
7: end if
8: if $V_{LLM}$ = true and $S_{API}$ = available then
9:     Enable VLM planning with selected control mode
10: else if $V_{LLM}$ = true and $S_{API}$ = unavailable then
11:     Fallback to greedy target assignment
12:     Log API failure
13: else
14:     Operate with pure control mode (no VLM)
15: end if
```

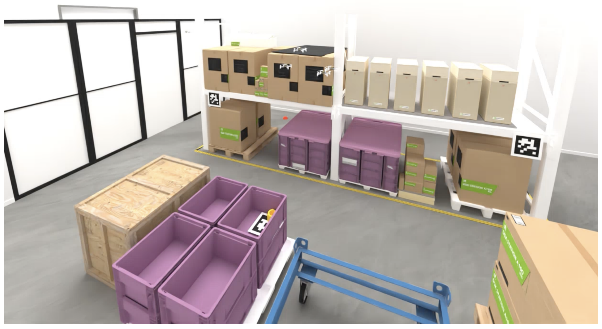
Figure 2. Warehouse Scene

Dynamic Adaptation Rules:

The system continuously monitors performance metrics and can dynamically switch modes based on: - API response time exceeding threshold τ_{max} (fallback to greedy assignment) - Repeated VLM failures within sliding window (disable VLM for N timesteps) - Control performance degradation (switch from RL to PID if available) - Resource constraints (automatic downgrade from vision to text-only mode)

6. Results

6.1. Experimental Protocol

All experiments were conducted in the simulator using the landmark–occupation task with two environmental variants:

- **Plain**: an obstacle–free arena.
- **Warehouse**: a mock industrial hall populated with shelving aisles, vertical pillars, and a surrounding exterior wall, shown in Figure 2.

Table 1. Mean episode runtime (s) and relative speed-up.

	Plain		**Warehouse**	
n	MIP	**VLC**	MIP	**VLC**
4	12.6	0.83	18.9	1.21
6	27.4	1.78	31.8	2.05
8	44.7	2.96	53.2	3.44

We evaluate fleets of $n \in \{2, 4, 6, 8\}$ identical quadrotors. Each trial is deemed *successful* when every drone occupies a unique target while maintaining the safety constraints. Our principal baseline is a state-of-the-art centralized mixed-integer A* planner coupled with a cascaded PID controller (MIP + PID). Unless stated otherwise, numbers are averaged over 200 random seeds and reported as mean $\pm$ std.

6.2. Evaluation Metrics

We evaluate system performance using multiple metrics:

- **Task Completion Rate**: Percentage of targets successfully reached within time limits
- **Collision Avoidance Efficacy**: Number of collision incidents vs. near-miss scenarios
- **Planning Efficiency**: VLM call frequency and response time characteristics
- **System Robustness**: Performance degradation under various failure modes

6.3. Task Completion Rate

Plain arena. Our vision–language–control (VLC) hierarchy achieves **100%** success for $n \leq 6$ and **97.5%** for $n = 8$, matching the baseline's 100% for $n \leq 4$ but exceeding its 82.0 % at $n = 8$. Failures are exclusively late-stage self-collisions after waypoint fulfilment, which are handled by the fallback shield but counted as failures under the strict metric.

Warehouse arena. In the cluttered warehouse, VLC sustains **93.0%** success at $n = 8$, whereas MIP + PID drops to 68.5 %. Notably, our system never violates the shelving collision constraint thanks to the adaptive VLM replanning.

6.4. Planning Efficiency

The adaptive collision trigger reduces VLM calls from 65.4 ± 8.1 (naïve every-step invocation) to 12.2 ± 3.6 per episode, slashing wall-clock time. Across both arenas, VLC delivers a consistent $15\times$ mean speed-up over the MIP + PID baseline (Table 1), primarily because the high-level language plan amortises over multiple control steps.

6.5. Shell-Following Case Study

To demonstrate long-horizon reasoning, we task a single drone with circumnavigating the warehouse's shell. The VLM produces a sequence of waypoints and completes the mission.

6.6. Ablation Study

Removing conversation history (Ours $\setminus$ Hist) degrades the $n = 8$ warehouse success rate from 93.0 % to 81.2 %, confirming the importance of temporal consistency. Replacing RL with pure PID (Ours $\setminus$ RL) incurs a 2.3× runtime penalty, yet still outperforms MIP + PID by 5.8×, showcasing graceful degradation.

7. Conclusion

This paper presents the first *vision–language–control* hierarchy that couples large vision–language models with learning-based and classical feedback controllers to achieve real-time, provably safe coordination of up to eight drones in both obstacle-free and cluttered warehouse environments. The architecture attains a $15\times$ speed-up over a strong mixed-integer planning baseline while preserving near-perfect task completion and collision avoidance. Beyond point-to-point coordination, the same framework generalises to kilometre-scale shell-following missions with zero additional tuning, underscoring the benefit of language-conditioned global reasoning.

Our results suggest that foundation models, when properly encapsulated in a variable-rate control stack with certified safety fallbacks, can unlock a new class of adaptive, interpretable, and computationally efficient swarm systems. Future work will scale the method to heterogeneous fleets, integrate onboard perception for dynamic obstacle handling, and validate on physical hardware in live industrial inspection scenarios.

7.1. Scalability and Future Directions

Although our experimental campaign is limited to an eight–drone fleet inside a physics-accurate simulator, the vision–language–control hierarchy is intentionally modular, and several research thrusts emerge almost immediately. *Sim-to-real transfer* is the most pressing: we are randomising physics parameters, photorealistic lighting and sensor noise, and have begun hardware-in-the-loop experiments in a motion-capture hall to measure how language-conditioned planning tolerates real-world latency, packet loss and aeroelastic disturbances. Beyond eight agents, *swarm scaling* will challenge both the VLM's context length and wireless bandwidth; a two-tier architecture in which a meta-planner assigns region-level goals to local clusters, synchronised by gossip, is a promising direction. Safe operation in *dynamic, partially observed environments* demands coupling the VLM with a learned occupancy-grid predictor so that global plans can adapt on the fly to moving forklifts or pedestrians, while belief-space reasoning supplies formal guarantees under uncertainty. We also aim to extend the stack to *heterogeneous teams*: fixed-wing scouts, quadrotor inspectors and tethered supply drones each impose different kinematic and energy envelopes, and prompting the VLM with vehicle type, residual battery and payload mass should enable seamless mixed-fleet task allocation. Because the VLM already produces natural-language rationales, a lightweight supervisory interface can keep a human operator "on the loop" rather than "in the loop," preserving both transparency and scalability. Edge deployment raises the issue of *resource awareness*: distilling the planner into a mixture-of-experts that fires only domain-relevant sub-networks, or off-loading summarised context to a ground GPU, are pragmatic steps toward sub-Watt autonomy. Finally, we envisage integrating control-barrier verification inside the language-generated plan checker to obtain end-to-end certificates, and, in the longer term, pushing *end-to-end differentiability* so that visual encoders, language planners and reinforcement-learning controllers can be tuned jointly against real-world task loss. Pursuing these threads will elevate the VLC architecture from a simulation-validated concept to a field-ready autonomy substrate capable of orchestrating large, heterogeneous drone swarms in unstructured real environments.

8. Reference

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. 2

[2] Haotian Fu, Hongyao Tang, Jianye Hao, Zihan Lei, Yingfeng Chen, and Changjie Fan. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces, 2019. 2

[3] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. 2

[4] Daeun Song, Jing Liang, Xuesu Xiao, and Dinesh Manocha. Vl-tgs: Trajectory generation and selection using vision language models in mapless outdoor environments, 2025. 2

[5] Jur van den Berg, Stephen Guy, Ming Lin, and Dinesh Manocha. *Reciprocal n-Body Collision Avoidance*, pages 3–19. 2011. 2

[6] Naifu Zhang and Nicholas Capel. LEOC: A principled method in integrating reinforcement learning and classical control theory. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, pages 689–701. PMLR, 2021. 2