IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 August 2025, accepted 14 September 2025,
date of publication 17 September 2025, date of current version 23 September 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3611336

RESEARCH ARTICLE

LLM-Driven Pareto-Optimal Multi-Mode Reinforcement Learning for Adaptive UAV Navigation in Urban Wind Environments

JIAHAO WU, HENGXU YOU, BOWEN SUN, AND JING DU
Department of Civil and Coastal Engineering, University of Florida, Gainesville, FL 32611, USA

Corresponding author: Jing Du (eric.du@essie.ufl.edu)

This work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Grant FA9550-22-1-0492, and in part by the Nvidia AI Technology Center (NVAITC) under Grant 00133684.

ABSTRACT Autonomous drones in complex urban wind environments must balance speed, safety, and energy efficiency under highly variable conditions. Traditional single-policy reinforcement learning controllers often perform poorly when exposed to scenarios beyond their training. We introduce a Pareto-optimal multi-mode framework that trains three specialized unmanned aerial vehicle (UAV) policies (aggressive, balanced, and cautious) via proximal policy optimization (PPO) with specific reward scalings, yielding controllers that collectively span the speed-safety-energy trade-off surface. To automate mode selection, we fine-tune a large language model (LLM) on 30,000 simulation-derived environment-performance tuples, allowing it to predict the optimal policy from building density, wind speed and orientation, battery state, and recent flight history. In a Unity-based Manhattan simulation with computational fluid dynamics (CFD) wind fields across four headings and 10 speed levels, the LLM-driven decision maker reduces average flight time by 16%, lowers the collision rate by 50%, and saves 18% energy compared to any single mode, while preserving nondominated trade-off performance. The decision maker also generalizes to unseen wind patterns and layouts without handcrafted heuristics, demonstrating the promise of combining Pareto-optimal reinforcement learning (RL) with LLM-based meta-decision making for UAV autonomy.

INDEX TERMS Autonomous drone, large language model, meta-decision making, pareto optimality, reinforcement learning, urban wind simulation.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs), commonly known as drones, have undergone extraordinary transformations over the past decade. Evolving from rudimentary, ground-controlled platforms, the current sophisticated autonomous drone system is capable of dynamic decision-making in complex real-world environments, which has been applied in agriculture, disaster management, and the construction industry [1], [2], [3]. Early applications of UAVs were confined to military reconnaissance and aerial photography, but rapid advancements in microelectromechanical systems (MEMS) technology and inertial navigation have broadened their utility across commercial, scientific, and recreational domains [4]. The integration of high-precision GPS, visual-inertial odometry, and simultaneous localization and mapping (SLAM) algorithms has enabled drones to maintain stable flight paths and navigate unfamiliar areas without human intervention. Advances in lightweight composite materials, brushless electric propulsion, and high-density lithium-polymer batteries have dramatically extended flight endurance from mere minutes to well over an hour in specialized platforms [5]. Concurrently, breakthroughs in miniaturized sensor suites, including high-resolution LiDAR, time-of-flight depth cameras, multispectral and thermal imagers, and advanced inertial measurement units, provide rich environmental feedback for real-time obstacle detection, terrain mapping, and aerial surveying [6], [7], [8]. At the

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

same time, the proliferation of compact AI accelerators, field-programmable gate arrays (FPGAs), and multicore embedded processors has brought machine learning inference on board, enabling complex perception and control loops to execute within tight latency constraints. Machine learning techniques, most notably reinforcement learning (RL), have capitalized on these hardware innovations, framing flight control as a sequential decision-making problem and leveraging physics-based simulators and digital twins to discover robust control policies [9]. Landmark successes, such as RL-based obstacle avoidance in cluttered environments, wind-aware trajectory planning in turbulent conditions, and zero-shot sim-to-real transfers for agile maneuvers, have validated the promise of end-to-end learning approaches and driven further interest in RL for UAV autonomy. While these advances have dramatically expanded UAV capabilities, they also highlight persistent weaknesses of monolithic controllers in dynamic urban conditions, motivating a new approach.

Despite this remarkable progress, most reinforcement learning-driven UAV controllers remain trained as single, generalized policies intended to perform across a wide range of conditions [10]. While capable of demonstrating impressive results within their training distribution, such monolithic controllers often exhibit brittle behavior when environmental parameters deviate from those encountered during training. Domain shift, such as varying air densities at different altitudes, can lead to dramatic performance degradation, control oscillations, or mission failure. The reliance on a single reward function to encode multiple objectives (e.g., speed, stability, and energy efficiency) forces practitioners to engage in extensive reward shaping and hyperparameter tuning, yet still sacrifices context-specific adaptability. Further, standard RL benchmarks for UAVs typically evaluate policies in static, well-defined scenarios, offering limited insight into real-world uncertainties such as sensor noise, varying wind, and dynamic obstacles. As a result, even state-of-the-art RL policies may struggle with generalization and robustness when deployed beyond simulated environments. Beyond purely technical limitations, the lack of explicit behavioral modes complicates human-drone interaction: operators cannot predict whether a single-mode agent will prioritize speed over safety in cluttered urban settings or stability under turbulent winds, eroding trust. Regulatory bodies such as the Federal Aviation Administration (FAA) and European Union Aviation Safety Agency (EASA) have noted that opaque, single-mode autonomy poses challenges for certification and airworthiness assessments, which demand demonstrable reliability across operational niches [11], [12].

In this work, we propose a multi-mode reinforcement learning architecture for autonomous drone navigation, leveraging a large language model (LLM) as the central decision-making agent for runtime mode selection. Specifically, we train three specialized RL policies: cautious, balanced, and aggressive, and combine them at runtime to form a Pareto-optimal solution across speed, safety, and energy-efficiency trade-offs under varying environmental conditions. The cautious policy maximizes stability and collision avoidance in obstacle-rich or turbulent scenarios, sacrificing speed when necessary. The aggressive policy, by contrast, emphasizes swift traversal and energy efficiency in low-risk settings. The balanced policy offers an intermediate trade-off between safety and performance. To coordinate these policies, we design an on-board LLM-based module that processes real-time sensor inputs, such as wind magnitude and direction, obstacle density metrics from depth cameras, and mission parameters, and generates a mode-selection decision at each time step. The LLM is trained via supervised fine-tuning on simulated flight logs labeled with the optimal policy under corresponding conditions, enabling it to infer the best-suited mode dynamically. This approach removes reliance on manual thresholding or handcrafted heuristics, allowing the decision maker to generalize across a wide range of unseen scenarios. We implement and evaluate our framework in a Unity-based simulation environment enhanced with urban computational fluid dynamics (CFD) wind models, variable building densities, and sensor noise profiles to approximate real-world uncertainties. Experimental results demonstrate that our LLM-driven multi-mode system achieves significant improvements over single-policy baselines: up to 16% reduction in average flight time, 50% decrease in collision incidents, and 18% energy savings. Moreover, the LLM decision maker exhibits robust generalization, successfully adapting to novel wind profiles and obstacle configurations without additional retraining. The main contributions of this paper are: 1. A Pareto-optimal multi-mode reinforcement learning framework for UAV navigation, integrating three specialized RL policies for distinct performance objectives. 2. An LLM-based decision-making module for context-aware mode selection, trained through supervised fine-tuning on simulated flight data. 3. A high-fidelity Unity simulation platform with urban CFD wind modeling for comprehensive policy training and evaluation.

II. LITERATURE REVIEW

A. REINFORCEMENT LEARNING FOR UAV NAVIGATION

Classical UAV controllers, which include PID, LQR, and MPC, have long provided reliable performance in well-understood operating envelopes [13]. However, these classical controllers face challenges in complex, unstructured, or highly stochastic scenarios. They rely on precise tuning of gains or cost weights, struggle to incorporate high-dimensional sensor inputs directly, and require human intervention to adapt to new payloads or evolving environmental conditions. In contrast, reinforcement learning formulates control as a sequential decision-making problem, using trial-and-error interaction to discover policies that map raw data directly to control commands [14]. Rather than depending on an explicit aerodynamic model, RL-based controllers learn to exploit nonlinear dynamics and adapt to unmodeled disturbances, enabling performance surpassing hand-tuned controllers operating in novel or highly variable environments.

The literature has extensively explored the promise of RL for UAV navigation, demonstrating its ability to learn end-to-end perception-to-action mappings, optimize multiple objectives simultaneously, and generalize across uncertain conditions. Reinforcement learning methods leverage deep networks to directly map raw, multi-modal observations to control actions, eliminating the need for manual feature engineering and hand-tuned gains. For example, convolutional architectures process depth images or LiDAR point clouds for obstacle avoidance, while recurrent modules handle time-correlated IMU data, enabling unified perception and control loops. To capitalize on these unified perception-action mappings, recent studies have implemented end-to-end deep-RL frameworks for UAV control. Cetin, et al. [15] applied a Deep Q-Network-based DRL framework, using a Joint Neural Network to fuse front-camera depth images with scalar features for fully autonomous UAV navigation in a simulated suburban environment with static and dynamic obstacles. The results showed that multimodal input accelerated training convergence and yielded near-perfect navigation performance, achieving 100% success in two scenarios and 98% in a three-drone test. Akhloufi, et al. [16] implemented a dual-framework integrating CNN-based deep reinforcement learning to predict probabilistic control actions, iteratively guiding a follower UAV toward the target. They found that this integrated approach enables accurate positioning and robust pursuit-evasion tracking performance using only onboard vision data. Arshad, et al. [17] proposed a data-driven deep CNN to preserve multi-level regional homogeneity and edge details in congested scenes. The network merged these features to learn texture variations that discriminated targets from background and avoided obstacles, and generated steering angles and collision probabilities for real-time UAV control. They validated the model on urban vehicle and bicycle datasets and demonstrated strong generalization and robust autonomous navigation performance in dynamic environments. Fei, et al. [18] proposed FRDDM-DQN, which combined an optimized Faster R-CNN for obstacle extraction with a novel Data Deposit Mechanism replay memory to improve Deep Q-Network training. They evaluated the method in a 3D simulation and demonstrated that it performs better in autonomous navigation and collision avoidance. Tong, et al. [19] developed a distributed DRL framework with two LSTM-based sub-networks, which demonstrated faster convergence compared to end-to-end networks. Moreover, integrating LSTM modules allowed the framework to capture drifting obstacle movements effectively, thereby enhancing UAV navigation safety in dynamic environments.

Building on these perception-driven advances, recent research has shifted toward designing composite reward structures and multi-objective RL algorithms that balance flight efficiency, energy consumption, and safety, ensuring UAV controllers can meet diverse mission requirements under real-world constraints. Ramezani Dooraki and Lee [20] developed a self-trained deep reinforcement learning controller that ingested depth and RGB images with multiple reward signals to produce continuous 3-DoF actions for autonomous UAV navigation in static and dynamic environments. Wu, et al. [21] introduced MONRL, a multi-objective reinforcement learning algorithm with memory architecture that learns safe path planning and wind compensation from building-layout imagery, obviating aerodynamic sensors. They validated it in a virtual New York City model, demonstrating robust obstacle avoidance and wind adaptation. Song, et al. [22] employed an evolutionary multi-objective RL algorithm to tackle the trajectory control and task offloading problem. Simulation results showed their method produced superior Pareto-optimal policies compared to two other basic evolutionary algorithms. Ramezani, et al. [23] developed a reinforcement learning framework augmented with Analytic Hierarchy Process and similarity-based experience replay to optimize UAV trajectories for SAR missions, explicitly balancing mission efficiency with human comfort and safety. Millar, et al. [24] formulated a convex multi-objective path-planning problem, which included maximizing data collection and minimizing risk. They applied Bayesian-AHP risk assessment, incorporating flight-time constraints and safety, and introduced a low-complexity MORL algorithm with convergence guarantees for efficient optimal movement decisions. Zhang, et al. [25] developed a DRL-based multi-objective path planning method for UAV-assisted sensor data collection. They decomposed the problem into subproblems solved via actor-critic and a modified pointer network to maximize data gathered and minimize flight time, yielding Pareto-optimal solutions with improved convergence, solution diversity, and robustness to changing sensor counts.

Although deep RL has enabled end-to-end control and decision-making, most studies still optimize a single, monolithic policy via a scalarized objective, which obscures inherent multi-objective trade-offs and often proves brittle under distribution shift across tasks, sensors, and environment dynamics. Empirical evidence shows deep RL agents can overfit training environments and underperform out of distribution, raising concerns for deployment in urban winds with sensor noise and clutter [10], [26], [27]. In multi-objective settings, scalarization further obscures how weights shape behavior, and most papers do not report systematic procedures for choosing reward weights or validating them with ablations [28]. This motivates a reproducible multi-objective alternative that exposes the trade-offs explicitly and remains robust across varying wind strengths/orientations and urban densities.

B. MULTI-POLICY REINFORCEMENT LEARNING

The literature on multi-policy control for UAVs has grown substantially, with research converging on two main approaches: hierarchical option-based frameworks and expert-ensemble models. These architectures promise improved adaptability and safety by decomposing complex navigation tasks into specialized modes or by blending multiple policies based on real-time conditions.

Hierarchical reinforcement learning frameworks introduce a top-level controller that selects among a set of lower-level skills or ''options,'' each encoded as a temporally extended policy [29]. Options comprise initiation sets, termination criteria, and intra-option policies learned to accomplish sub-tasks such as stabilized hovering, obstacle circumnavigation, or precision landing. By training the option-critic architecture jointly, agents automatically discover and refine useful behaviors, reducing manual task decomposition efforts. Liu, et al. [30] introduced a hierarchical deep reinforcement learning framework for UAV navigation, leveraging a multi-level policy architecture to decompose long-horizon tasks into manageable sub-tasks, augmented by averaged value estimation and temporal attention in recurrent networks. They demonstrated superior performance over DrQ in realistic simulations and real-world tests. Ramezani and Amiri Atashgah [31] proposed a hierarchical reinforcement learning algorithm that integrates an LSTM-based battery consumption predictor into a high-level controller for setting energy-efficient goals and a low-level controller augmented with hindsight experience replay to boost sample efficiency. AlKhonaini, et al. [32] implemented a multi-level policy network trained with the REINFORCE algorithm and an entropy regularization term to extract and interpret RF signal features for intruding UAV detection and identification. The system achieved 99.7% detection accuracy, demonstrated superior cumulative return, and reduced loss, establishing its efficacy for RF-based UAV surveillance. Nguyen, et al. [33] developed a deep hierarchical RL framework in which a UAV acts as an aerial sheepdog, using a curriculum of DRL-trained lessons to shepherd UGV swarms via hierarchical output fusion. They validated the method in ROS-based simulations and indoor trials, demonstrating sim-to-real and scale generalization. While option hierarchies exploit temporal abstraction, alternative methods blend multiple experts in parallel to handle diverse flight conditions. Mixture-of-Experts architectures train several specialized expert networks, each optimized for particular operating conditions, alongside a gating network that dynamically weights expert outputs based on current sensor inputs [34]. This soft selection enables smooth interpolation between behaviors, avoiding abrupt transitions that can destabilize the UAV. Blending experts tuned for energy-efficient flight and high-speed traversal can yield substantial improvements in energy consumption.

To further increase adaptability, meta-learning techniques have been employed so that the multi-policy system can incorporate new modes or rapidly change its gating network after a single or a few novel trials. Zhao, et al. [35] introduced a meta-learning-enhanced multi-agent DDPG algorithm that initializes networks via automatically weighted MAML trajectories and employs a Reward-TD prioritized replay mechanism for improved efficiency. They demonstrated in UAV swarm scenarios that this approach significantly boosts task success rate, average reward, and robustness. Yel and Bezzo [36] presented a meta-learning approach for UAV trajectory tracking that adapts at runtime to unseen actuator faults and external disturbances. The method integrates runtime monitoring with data pruning and feedback-based reference adjustment, all without modifying the controller. O'Connell, et al. [37] proposed an online composite adaptation method that treats deep neural network outputs as basic functions for representing variable wind conditions, optimized via meta-learning. This enabled fast onboard updates by integrating the network-derived basis functions into an adaptive control loop for UAVs. Eldeeb and Alves [38] introduced a few-shot meta-offline RL algorithm that integrates conservative Q-learning with MAML to train UAV trajectory and scheduling offline, minimizing age-of-information and transmission power without online interaction. It converges faster than DQN and CQL, achieves optimal AoI-power trade-offs from limited data, and adapts resiliently to new environments. Soorki, et al. [39] developed a UAV-assisted LoRa gateway control policy through deep RL framed as a POMDP and a deep meta-RL extension that adapts using prior SAR experiences. In campus, plane, and canyon tests, meta-RL reduced SAR time slots by over 65% and decreased UAV energy consumption significantly against baseline RL methods.

Prior multi-policy approaches, such as options/HRL and mixtures-of-experts, show that specialized skills can outperform a single generalist, but two gaps remain. First, most works do not demonstrate coverage of the multi-objective Pareto frontier in realistic dynamic conditions; results are typically reported on static or simplified settings without wind-aware performance trade-offs [29], [30], [31]. Second, the selector is usually a small gating network [34], which can be effective but offers limited interpretability and OOD robustness, and is rarely accompanied by ablation studies that quantify the selector's contribution versus single-policy baselines. A runtime specialization mechanism is missing that is transparent, data-driven, and validated against wind-aware, city-scale variability.

C. AI REASONING AND DECISION MAKING

In modern autonomous systems, AI reasoning frameworks form the cognitive backbone that interprets complex sensory data, models environmental uncertainty, and orchestrates multi-step decision processes. By integrating probabilistic inference, neural reasoning modules, and domain knowledge, these systems can dynamically adapt their behavior to evolving mission goals and safety requirements.

Extensive research into AI reasoning frameworks for robotics has shown how these systems integrate diverse sensor data, apply probabilistic inference for context-sensitive decision-making, and generate interpretable explanations that improve transparency and resilience in unpredictable environments. Zhou, et al. [40] introduced a framework named LKIRF that combined large language models with offline knowledge graphs built from human motion and environmental data and online inference graphs. They found that LKIRF substantially improved prediction accuracy across diverse

scenarios and increased transparency and explainability of the intent-inference process. Sobrín-Hidalgo, et al. [41] extended their RAG-based LLM explanation system by incorporating Vision-Language Models to fuse onboard camera imagery with textual logs. They demonstrated that this multimodal approach significantly improved the precision and clarity of generated explanations. Liu, et al. [42] introduced an end-to-end VLA model that integrated the Mamba state space model with co-trained vision and language representations. The results showed that the model could achieve strong reasoning and pose accuracy while running inference three times faster than prior VLA models. Sun, et al. [43] presented an autonomous office assistant that harnessed a LLM within a multi-agent collaboration framework to perform sophisticated reasoning, anticipate challenges, and retrieve pertinent information from memory. Huang, et al. [44] proposed an end-to-end autonomous driving framework that fused multimodal LLM reasoning with trajectory planning under a novel alignment constraint, ensuring the model's chain-of-thought directly guides its actions. Evaluated on nuScenes and DriveLM-nuScenes, this framework achieved record-low trajectory errors and collision rates, outperforming existing methods. Kannan, et al. [45] presented SMART-LLM, a framework that employed LLM-driven reasoning via few-shot prompting to decompose high-level instructions, form robot coalitions, and allocate tasks, converting commands into executable multi-robot plans. This framework was evaluated on a benchmark covering four instruction categories, demonstrating robust reasoning-driven planning performance across simulation and real-world evaluations.

In aerial drone applications, AI reasoning agents orchestrate mission planning in response to environmental changes, enhancing autonomy and safety [46]. AI reasoning enables the UAV to plan contextual actions, such as trajectory adjustments, sensor reconfiguration, and real-time policy switching. Without manual intervention, this agent can improve autonomy and resilience in unpredictable environments. Building on this capability, recent work has integrated LLM-based reasoning directly into UAV autonomy architectures. Zhao and Lin [47] introduced a hardware-software co-design for LLM-empowered UAVs. Their prompt framework integrated LLM reasoning with UAV autonomy stacks and was validated on diverse real-world missions, demonstrating effective onboard planning and perception. Wang, et al. [48] developed a modular prompt framework for LLM-driven drone control comprising guidelines, skill APIs, constraints, and examples to enhance reasoning and ensure code compliance. This adaptable structure was shown to improve coherence and reliability of LLM-generated control code across varied UAV platforms and missions. Lin, et al. [49] built up an ACP-based urban UAV framework integrating an MLLM agent with a human-in-the-loop feedback loop for autonomous flight and interaction. It instructed fine-tuned MLLMs on a 3D spatial command dataset to imbue spatial reasoning and used prompt fine-tuning to decompose complex UAV tasks in simulation. Chen, et al. [50] employed real-time language models to convert situational context and obstacle data into sub-task instruction sequences for autonomous urban path planning in a 3D simulation. Gao, et al. [51] proposed an end-to-end zero-shot aerial VLN framework leveraging an LLM agent to predict navigation actions from natural language instructions and visual cues. They designed a Semantic-Topo-Metric Representation that extracts semantic masks of landmarks, projects them into a top-down map, and encodes distance metrics as prompts to bolster spatial reasoning.

Recent work suggests language-conditioned reasoning modules can improve skill selection and generalization in robotics [52]. However, to our knowledge, LLM-based meta-decision makers have not been systematically evaluated as runtime policy selectors in robotics under realistic, dynamic-rich conditions with explicit multi-objective trade-offs and reproducible weighting/ablation protocols. Existing LLM-for-UAV studies demonstrate feasibility but stop short of connecting language-guided decisions to Pareto-optimal RL behaviors under rigorous ablation and grid-search protocols [47], [48]. This leaves an opportunity to pair specialized RL policies with an LLM selector that is trained on environment-performance tuples and validated for robustness and interpretability.

To address these gaps, we replace scalarized single-policy control with three specialized RL policies that explicitly span the speed-safety-energy Pareto frontier, and we introduce an LLM-based meta-controller that performs runtime policy selection from compact, sensor-derived context to improve robustness under dynamically varying conditions. We further ensure reproducibility and interpretability by publishing a comprehensive grid search for reward weights, providing a formal algorithmic specification with defined variables, and conducting ablation studies that quantify the contributions of policy specialization and the meta-controller.

Table 1 contrasts recent hierarchical/multi-policy RL and LLM-guided control methods across domain, architecture, objectives, selection mechanism, and outcomes. Unlike most prior work centered on manipulation or locomotion in laboratory settings, our study targets the safety-critical UAV navigation domain in dense urban wind fields, where decisions must trade off time versus success under sensor noise and latency. Rather than learning skills and a selector jointly or having an LLM emit continuous actions, we use the LLM strictly as a discrete meta-selector over a fixed set of Pareto-trained flight modes (aggressive/balanced/cautious). This separation of multi-objective design from execution yields transparent, auditable switch rules with crisp, data-driven decision boundaries (over wind speed and building density), while the pre-verified low-level controllers enable safety gating and simpler certification than end-to-end policies. The result is a domain-ready, interpretable, and data-efficient selection mechanism with explicit sim-to-real assumptions, complementary but distinct from prior HRL and LLM-action approaches.

TABLE 1. Comparison with recent related work on policy selection and hierarchical control.

Work (year)	Domain	Architecture	Objectives	Policy/skill selection	Outcomes/notes
H-REIL (RSS 2020) [53]	Autonomous driving	High-level RL switches among low-level imitative policies	Safety-efficiency trade-off	PPO meta-policy chooses mode per interval	Switching outperforms IL and single-mode baselines
RMA (2021) [54]	Legged locomotion	Base locomotion policy + online adaptation module	Robust walking under terrain/payload changes	Latent context conditions policy	Works across rocks, stairs, sand; adaptation critical
Option-Critic (AAAI 2017) [55]	General RL	End-to-end learning of options, terminations, and policy-over-options	Maximize return with temporal abstraction	Learned policy over options	Foundational HRL framework
HIRO (NeurIPS 2018) [56]	General RL	Two-level HRL with off-policy correction	Long-horizon control & sample efficiency	High-level sets goals for low-level	Strong long-horizon performance
BUDS (RA-L 2022) [57]	Manipulation	Discovered skill library + meta-controller (hierarchical BC)	Long-horizon imitation from demos	Meta-controller selects skill + subgoal	66% sim success; 56% real-kitchen success
Policy Stitching (CoRL 2023) [58]	Manipulation	Modular policy with latent alignment; compose robot/task modules	Transfer to new robot-task combos	Compose pretrained modules ("stitching")	Zero/few-shot transfer across tasks
PaLM-SayCan (2022) [52]	Mobile manipulation	LLM plan proposals grounded by value/affordance model	Long-horizon instruction following	LLM proposes: value model filters	Successful long-horizon tasks in the real world
VoxPoser (2023) [59]	Manipulation	LLM + VLM compose 3D value maps; model-based planner	Language-conditioned trajectory synthesis	Planner selects actions from value maps	Zero-shot closed-loop trajectories
RT-2 (CoRL 2023) [60]	Manipulation	Vision-Language-Action transformer (web + robotics data)	Semantic generalization in control	End-to-end action token generation	Improved generalization to novel objects/instructions
This work (LLM selector)	UAV navigation	LLM meta-controller selects Pareto-trained flight modes	Minimize time while maintaining success and safety	Fine-tuned LLM chooses mode based on environmental factors	Interpretable switching boundaries; complements HRL/adaptation

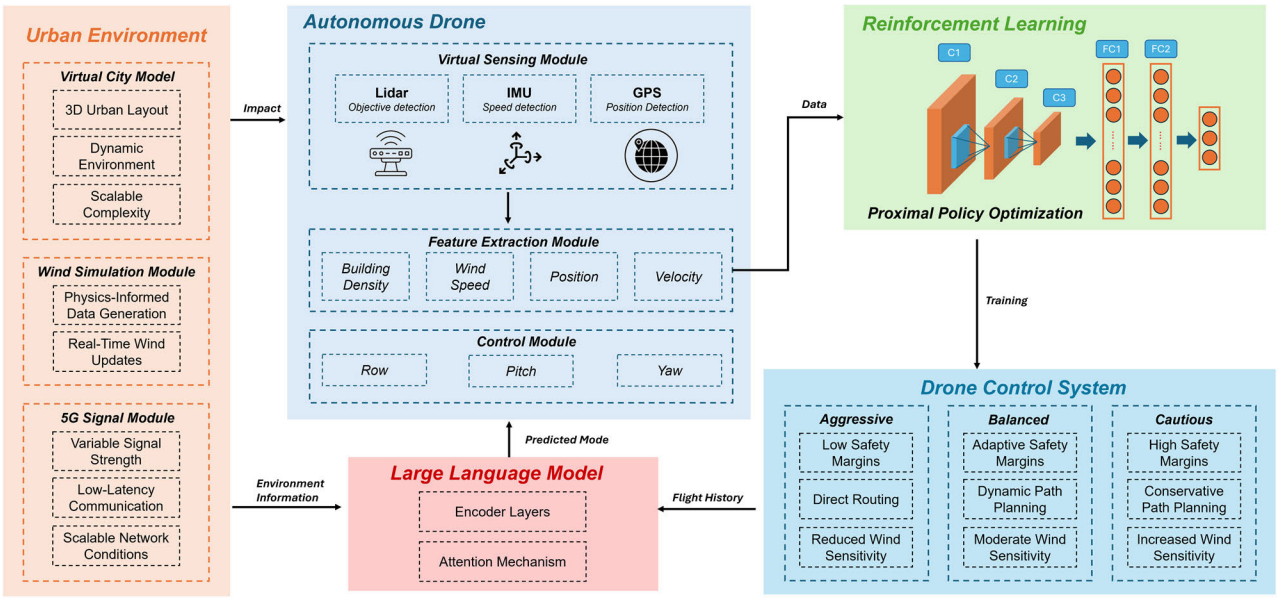

FIGURE 1. The integrated system architecture of the proposed MMRL framework.

III. METHODOLOGY

A. OVERVIEW

Our multi-mode reinforcement learning framework integrates high-fidelity urban wind simulation, specialized RL policy training, and an onboard LLM for real-time mode selection. First, we generated a virtual Manhattan district with a 2,000 m × 2,000 m footprint, populated with nearly 500 high-rise buildings and precomputed steady-state CFD wind fields for four principal headings and ten different speed levels. Using Unity 3D, we simulated a DJI-Mavic-inspired quadrotor in

each 400 m grid cell under 40 distinct wind conditions and trained three separate RL agents, which were aggressive, balanced, and cautious, via Proximal Policy Optimization (PPO) with a modular, sub-objective reward function.

We adopt PPO due to its clipped surrogate objective and strong empirical stability/robustness on continuous-control benchmarks, while retaining simple implementation and favorable wall-clock performance [61]. Concretely, PPO's on-policy actor-critic updates with generalized advantage estimation, minibatch Stochastic Gradient Descent (SGD) over multiple epochs, and ratio-based clipping limit destructive policy moves and make it straightforward to implement and tune in practice. In contrast, Deep Q-Learning (DQN) is designed for discrete action spaces and must be discretized or replaced for continuous control [62], which either explodes the action grid or sacrifices control resolution, and typically requires target networks/replay plus reward clipping to remain stable. Continuous-action methods like deep deterministic policy gradient (DDPG) address this but are known to be more sensitive to hyperparameters and exploration noise [63]. Soft Actor-Critic (SAC) is a strong off-policy alternative with excellent sample efficiency, but it introduces additional replay/temperature-tuning complexity and sensitivity to design choices that were not advantageous under our single-GPU, Unity-based training loop [64]. Given these constraints, PPO provided the most reliable convergence for our continuous-action task with a predictable compute footprint and minimal tuning overhead.

In our design, each policy learned a unique trade-off between flight time, energy use, and safety. Once trained, we evaluated all three agents on 10,000 trials per mode, covering every grid, wind orientation (along, against, perpendicular), and wind speed multiplier. We then extracted the fastest mode in each scenario to form a supervised dataset of (density, wind speed, orientation) for the best mode. Finally, we fine-tuned a pre-trained LLM on 80 % of these examples, prompting it with environmental descriptors and recent flight history so that at runtime it can infer the optimal policy mixture. And we used the remaining 20% of data to validate the LLM's generalization.

Fig. 1 presents the whole system for this proposed framework. On the left, the urban environment module synthesizes three components: a 3D virtual city model with real building layouts and obstacles, a physics-informed wind simulation that provides spatially varying gust fields, and a 5G signal emulator for variable communication conditions. These environmental factors drive the autonomous drone stack at the center: a virtual sensing module (Lidar, IMU, GPS) collects raw data, which can generate environmental descriptors. These descriptors serve two parallel purposes. First, they feed into our reinforcement learning pipeline, where PPO, with mode-specific reward functions, trains three distinct agents embodying aggressive, balanced, and cautious personalities. After training, each agent is evaluated across the simulated grid and wind scenarios to produce detailed flight-history logs. Second, the same descriptors and recent flight history

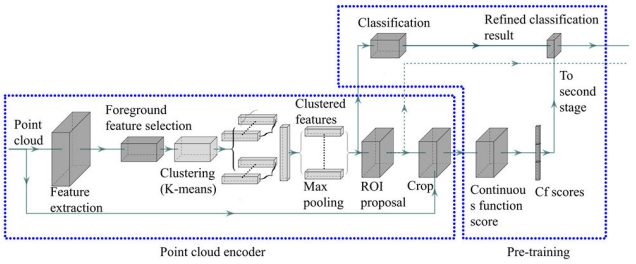

FIGURE 2. Two-stage point-cloud encoder and classification pipeline.

are provided as prompts to a large language model, which learns via supervised fine-tuning to predict the optimal mode label for any new condition. At runtime, the LLM ingests the current feature vector and the most recent flight data to output a predicted mode.

B. LIDAR DATA PROCESSING

The point cloud encoder used in our work adopts the structure illustrated in Fig. 2, originally developed for 3D object detection. The network begins with a feature extraction module, which processes raw point cloud data to generate dense point-wise representations. These features are then passed through a foreground feature selection stage that isolates informative spatial regions. To improve recognition and semantic grouping, a K-means clustering module is applied to group the extracted features based on local geometric similarity. These clustered feature representations provide a structured and semantically enriched input to subsequent modules in our framework. This is followed by a max pooling operation over each cluster to obtain compact clustered features. These features are fed into an ROI proposal module, which predicts candidate object regions. The proposals are evaluated by a classification module, and the results are further refined through a continuous function branch that adjusts the classification scores, resulting in refined predictions.

In our application, this network is modified and used purely as a point cloud encoder, adapted from PointNet++ [65]. The encoder was pretrained on the KITTI 3D object detection dataset, leveraging the full pipeline described above [66]. During downstream training in our model, only the encoder is used, and its parameters are kept frozen to preserve the learned 3D geometric representations.

C. BASIC REWARD FUNCTION

In our multi-mode reinforcement learning (MMRL) framework, the reward function serves as the primary mechanism for shaping agent behavior through trial-and-error interactions. Building on the MMRL paradigm, we decompose the total reward into a collection of interpretable sub-objectives that reflect critical aspects of drone navigation: progress toward the goal, mission completion, environmental hazard avoidance, safety, and temporal efficiency. This modular approach enables clear alignment between system-level

requirements and learned behaviors, while the absence of hard-coded weights in the base formulation ensures flexibility for future tuning. We first define the overarching reward structure and then describe each component in detail, followed by the mode-specific scaling factors that produce aggressive, balanced, and cautious behaviors. The reward system that we designed in this experiment is as in (1).

$$R = r_{distance} + r_{arrival} + r_{\text{avoidance}} + r_{collision} + r_{wind} + r_{timeout} + r_{time} \tag{1}$$

where R represents the total reward received by the drone at each timestep. This scalar is composed of multiple interpretable sub-objectives that align with core navigation requirements: $r_{distance}$ quantifies stepwise progress toward the target, $r_{arrival}$ provides a discrete bonus upon safe arrival at the goal, $r_{\text{avoidance}}$ rewards successful avoidance of designated strong-wind regions, $r_{collision}$ imposes a penalty and terminates the episode when a collision occurs, r_{wind} applies a small penalty for remaining in high-disturbance or looping zones, $r_{timeout}$ penalizes failure to complete the mission within the allotted time budget, and r_{time} enforces temporal efficiency through a minor per-step cost. In this base formulation, each sub-objective contributes equally, without additional weighting, simplifying the initial design and ensuring that learned behaviors reflect the raw trade-offs inherent in the task.

The first reward function is based on the distance between the target and the drone. This continuous incentive measures the reduction in distance to the target at each timestep, normalized by the episode's starting separation. By rewarding every meter of progress rather than absolute proximity, the drone is guided along direct, efficient trajectories and discouraged from oscillating or idling near the goal. Normalization ensures consistency across missions of different lengths, providing dense feedback that accelerates learning in long-horizon tasks. We express this distance-reduction term in (2).

$$r_{distance} = \sum_{t=1}^{T} \frac{d_{t-1} - d_t}{d_0} \times 100 \tag{2}$$

where $r_{distance}$ measures the normalized decrease in distance to the target at each timestep; d_t and d_{t-1} are the distances at steps t and t-1, and d_0 is the initial episode distance.

For the second reward function, a large, one-time bonus is awarded when the drone first enters the goal region, clearly distinguishing genuine task success from mere proximity. Without this discrete reward, agents might "dance" around the boundary to exploit the continuous distance term. By setting the bonus above any possible cumulative incremental gains, policies learn that safe arrival is the highest-value outcome, preventing indecisive hovering. This function is formalized in (3).

$$r_{arrival} = \begin{cases} +10 & \textit{if arrives target} \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

Strong wind areas introduce instability and higher energy use. To instill proactive hazard avoidance, the drone earns a positive reward when it plans and executes a path that skirts around predefined wind polygons without entering them. Instead of penalizing intrusion, this design encourages the agent to recognize and prefer safer corridors, mirroring real-world routing strategies. Such an incentive enhances both stability and battery life. The wind-avoidance term is detailed in (4).

$$r_{\text{avoidance}} = \begin{cases} +10 & \textit{if avoids strong wind zone} \\ 0 & \textit{otherwise} \end{cases} \tag{4}$$

Collisions can cause severe real-world consequences. We therefore impose a substantial negative penalty and terminate the episode immediately upon any impact. This creates a strong deterrent against reckless shortcuts through cluttered environments and forces the policy to internalize obstacle layouts. Because no further rewards accrue post-collision, the agent quickly learns that safety is paramount. The collision penalty is specified in (5).

$$r_{collision} = \begin{cases} -10 & \textit{if a collision occurs} \\ 0 & \textit{otherwise} \end{cases} \tag{5}$$

To prevent the drone from getting "stuck" in high-disturbance or looping regions, we apply a small penalty per timestep spent inside designated disturbance zones. The loss scales with proximity to the zone center, gently nudging the agent toward open airspace without overwhelming primary objectives. This fosters exploration and dynamic rerouting when conditions warrant. The functional form appears in (6).

$$r_{wind} = \begin{cases} -0.1 \times \dfrac{d_{zone}}{D_{max}} & \textit{per step} \\ 0 & \textit{otherwise} \end{cases} \tag{6}$$

where r_{wind} penalizes each timestep spent in a disturbance region; d_{zone} and D_{max} denote current and maximum zone distances.

Exceeding the maximum timestep before reaching the target model's finite battery life or mission windows. A fixed negative penalty prevents policies from favoring overly cautious behaviors that never complete the mission. By balancing against the arrival bonus, agents learn to trade off safety and timeliness under resource constraints. This timeout cost is encoded in (7).

$$r_{timeout} = \begin{cases} -10 & \textit{if step reaches maxmium} \\ 0 & \textit{otherwise} \end{cases} \tag{7}$$

Lastly, a minor per-step cost encourages the drone to minimize idle time and reduces the incentive to loiter in safe zones. Although small relative to other terms, this continuous penalty smooths the learning landscape and complements the timeout cost by providing ongoing pressure for efficient progress. Its expression is given in (8).

$$r_{time} = -0.001 \textit{ per step} \tag{8}$$

D. MODE-SPECIFIC SCALING FACTORS

While the base reward components establish a common foundation for all policies, our MMRL framework requires distinct navigational "personalities" to span the Pareto frontier of speed, safety, and energy efficiency. To this end, we introduce three multiplicative scaling factors (k_1, k_2, k_3) that adjust the distance-reduction reward, the per-step time penalty, and an additional conditional penalty, respectively:

$$r^{mode}_{distance} = k_1 \times r_{distance} \tag{9}$$

$$r^{mode}_{time} = k_2 \times r_{time} \tag{10}$$

$$r^{mode}_{wind} = k_3 \times r_{wind} \tag{11}$$

where k_1 scales the urgency of forward progress: a larger value rewards every meter of closing distance more heavily, pushing the drone toward direct, high-speed trajectories; k_2 scales time pressure: increasing it makes each additional timestep more costly, encouraging faster mission completion; setting it to zero removes any explicit rush; k_3 scales the penalty for disturbance-zone exposure: higher values enforce stronger avoidance of gusty or unstable regions, while lower values permit greater tolerance of such conditions.

To justify our choice of scaling factors k_1, , k_2 and k_3, we conducted a coarse grid search over five distance-reward levels (0.5, 1.0, 1.5, 2.0, 3.0), three time-penalty levels (0, 1, 2), and three disturbance-penalty levels (0, 1, 2). These candidate values were informed by our previous research [21] and reflect an empirically driven tuning approach. Each triple was trained and evaluated in three urban scenarios (low, medium, high building density) under varying wind conditions, with ten independent runs per setting to ensure statistical robustness. We evaluated every candidate triple by training an agent and measuring its flight time, collision rate, and energy consumption; by plotting these three metrics we constructed the Pareto frontier and then selected the three non-dominated configurations that together span the full trade-off surface, Aggressive (3, 2, 0) for the fastest flights, Cautious (0.5, 0, 2) for the safest, and Balanced (1.5, 1, 1) as the intermediate compromise and most energy-efficient behavior. This data-driven procedure, rather than heuristic guessing, ensures reproducibility and yields clearly distinct modes along the speed, safety, and efficiency dimensions. The complete grid-search results are given in **Table 6. (Appendix A)**.

In the aggressive configuration, we set $k_1 = 3$, $k_2 = 2$, $k_3 = 0$. Tripling the progress-to-goal reward makes each meter of forward movement extraordinarily valuable, compelling the drone to follow the most direct trajectory toward its destination. Doubling the per-step penalty further punishes any hesitation or detour, reinforcing a high-speed flight profile. By zeroing out the disturbance-zone penalty, the agent effectively ignores turbulent regions, allowing it to cut corners and skim close to hazard boundaries without cost.

During training, this combination drives the policy to exploit open corridors and minimize travel time, often at the risk of reduced safety margins. Consequently, the aggressive agent achieves the fastest completion times and lowest energy consumption in benign conditions, albeit with a higher collision rate when navigating dense clutter or sudden wind gusts.

For the balanced mode, we use $k_1 = 1.5$, $k_2 = 1$, $k_3 = 1$. A 1.5 multiplier on the distance reward still encourages efficient route planning, while preserving sufficient flexibility to reroute around obstacles. The standard time penalty maintains a healthy sense of urgency without penalizing cautious maneuvers, and the base disturbance penalty continues to nudge the drone away from high-wind zones. This middle-ground tuning fosters a policy that naturally negotiates trade-offs between speed and safety: it slows down modestly to avoid tight passages or strong turbulence but never lingers long enough to jeopardize mission timeliness. Empirically, the balanced agent achieves consistently low collision rates while completing missions in a reasonable time, making it well-suited for environments with mixed risk levels, such as semi-urban areas with intermittent gusts or moderate obstacle density.

In the cautious setting, k_1 is reduced to 0.5, k_2 is set to 0, and k_3 is increased to 2. Halving the distance-reduction reward greatly diminishes the incentive for direct flight, encouraging the drone to take longer but safer routes. Removing the time penalty eliminates any rush, allowing the agent to loiter if necessary, such as waiting out gusts or conducting thorough scans of the environment. Doubling the disturbance-zone penalty creates a strong aversion to high-wind areas or loop-prone regions, effectively turning them into near "no-fly" zones. During training, the cautious policy learns to maintain large safety buffers around obstacles and to hover briefly to reassess when encountering unexpected conditions. Consequently, this mode achieves the lowest collision rates and highest stability, at the expense of longer flight durations and increased energy use, which is ideal for highly cluttered, wind-swept, or otherwise hazardous operational contexts.

By tuning these three scalar parameters, our framework produces specialized RL agents that collectively span the multi-objective trade-off surface, allowing dynamic selection of the optimal navigation strategy in response to changing mission requirements and environmental conditions.

E. LLM FINE-TUNING

We use an LLM-based selector to leverage high-level reasoning, compositional generalization, and interpretability in long-horizon decision making. Prior work shows that language models can plan, ground language in robotic affordances, and transfer across tasks and embodiments, such as Zero-Shot Planners, SayCan, PaLM-E, and RT-2 [52], [60], [67], [68]. Compared with fixed thresholds, finite-state machines, or small MLP gating, an LLM can fuse heterogeneous context (numeric descriptors, categorical flags, and short histories of data), reason over mission constraints stated in natural language, and adapt its selection rule via prompt edits or few-shot examples without retraining. It also

supports compositional decision logic (combining wind, density, and battery cues with soft constraints) and produces operator-facing rationales (e.g., "selected cautious due to strong crosswind and low battery"), which improve auditability, debugging, and trust. Practically, this yields more robust mode selection under distribution shift, while parameter-efficient fine-tuning (LoRA) keeps training and inference overhead modest on a single GPU [69].

We use Llama 3.1 8B (base) as the real-time decision maker. The model is trained via supervised fine-tuning on 10,000 labeled examples generated by running the three RL policies (aggressive, balanced, cautious) across diverse urban and wind conditions. Each example consists of: An **environment descriptor** vector (d, v, o, b, t), where $d \in [0, 40]\,\%$ is the local building density, $v \in [0, 20]$ m/s is the instantaneous wind speed, $o \in [along, against, perpendicular]$ indicates wind orientation relative to the planned heading, $b \in [0, 100]\,\%$ is the remaining battery state, and t is the corresponding flight time in seconds. A **mode label** $l \in [0, 1, 2]$ corresponding to the policy that achieved the minimum total flight time under those exact conditions (0 = aggressive, 1 = balanced, 2 = cautious). We convert each example into a natural-language prompt of the form: ***"density = 22.5%, wind = 9.8 m/s (perpendicular), battery = 65%, time = 52.3, mode = 0".*** At inference time, using Llama-3.1-8B as a real-time selector imposes a tight latency/memory budget, so we constrain prompts to short, structured strings and a small discrete action set, avoiding long chain-of-thought or multi-turn deliberation. Finally, labels are simulation-derived and may encode simulator biases; although our evaluation shows strong performance across diverse routes, closing any residual sim-to-real gap will require targeted data collection and lightweight, incremental fine-tuning on real flights, an avenue we plan to pursue with active sampling of under-covered wind–density regimes.

The 10,000 examples are shuffled and split 80/10/10 into training, validation, and test sets, ensuring no grid-wind combination appears in both train and test. To reduce computational and memory overhead, we use LoRA (parameter-efficient adapters) with FP16 on a single NVIDIA RTX 4090 (24 GB VRAM); we set a context length of 512 tokens and batch size 32. Training minimizes cross-entropy between the model's three-way output and the ground-truth label, using AdamW (learning rate $= 2 \times 10^{-5}$, weight decay $= 0.01$) with a linear decay schedule to zero over 10 epochs, gradient clipping (max-norm = 1.0), and early stopping after two consecutive non-improvements in validation accuracy; the best checkpoint is selected by validation accuracy. These settings reflect the single-GPU compute budget and were sufficient for stable convergence.

To mitigate overfitting on a modest dataset, we limit trainable capacity via LoRA (freezing the backbone), use weight decay (0.01), apply adapter dropout ($p = 0.10$) and label smoothing ($\varepsilon = 0.05$), and introduce prompt-level randomness (randomized field order during training) to discourage

Algorithm 1 Fine-Tuning LLAMA 3.1 8B with LoRA for Drone Mode Selection

Input: Dataset $\mathcal{D} = \{(d_i, v_i, o_i, b_i, t_i, l_i)\}_{i=1}^{10,000}$
where d_i = building density, v_i = wind speed, o_i = wind orientation, b_i = battery state, t_i = flight time, l_i = mode label (0, 1, or 2)
Prompt format: "density = …, wind = …, battery = …, time = …, mode = …"
Split: 80% train, 10% val, 10% test (no grid-wind pair overlap)
Batch size $B = 32$, context length $C = 512$ tokens, `FP16` precision
AdamW optimizer: learning rate $\eta = 2 \times 10^{-5}$, weight decay $\lambda = 0.01$
Linear decay schedule (annealed to zero over 10 epochs)
LoRA parameter-efficient adapters enabled
Gradient clipping $g_{\max} = 1.0$
Max epochs $E = 10$, early stopping patience $\tau = 2$
Hardware: NVIDIA RTX 4090 (24GB VRAM, fits 8B model + LoRA)
Output: Best fine-tuned model $\mathcal{M}^*$ for drone mode selection

$\mathcal{M} \leftarrow$ ***LoadPretrained***(LLAMA 3.1 8B, LoRA enabled)
opt $\leftarrow$ ***AdamW***($\mathcal{M}$, η, λ)
$bestAcc \leftarrow 0$, $patience \leftarrow 0$
Format each example as a prompt–label pair using fixed template
Randomly split data: 80% train, 10% val, 10% test
(ensure no grid-wind overlap between train and test sets)
for $e = 1$ **to** E **do**
  for each batch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$ of size B **do**
    $\mathbf{y}_{\text{pred}} \leftarrow \mathcal{M}(\mathcal{B}_{\text{inputs}})$
    $\ell \leftarrow$ ***CrossEntropyLoss***($\mathbf{y}_{\text{pred}}$, $\mathcal{B}_{\text{labels}}$)
    ℓ.***backward***()
    ClipGradNorm($\mathcal{M}$.`parameters`(), $g_{\max}$)
    opt.***step***()
    opt.***zero_grad***()
  end for
  (ℓ_v, a_v) $\leftarrow$ ***Evaluate***($\mathcal{M}$, $\mathcal{D}_{\text{val}}$)
  if $a_v > bestAcc$ **then**
    $bestAcc \leftarrow a_v$
    $\mathcal{M}^* \leftarrow$ ***Copy***($\mathcal{M}$)
    $patience \leftarrow 0$
  else
    $patience \leftarrow patience + 1$
  end if
  LogMetrics(e, ℓ_v, a_v)
  if $patience \geq \tau$ **then**
    break
  end if
end for
(ℓ_t, a_t) $\leftarrow$ ***Evaluate***($\mathcal{M}^*$, $\mathcal{D}_{\text{test}}$)
return $\mathcal{M}^*$

FIGURE 3. Algorithm for supervised fine-tuning of Llama 3.1 8B.

lexical memorization. The train/validation curves remained aligned; validation accuracy peaked around epochs 7-8, and early stopping typically halted training at epoch 8. On the held-out test set, accuracy and F1 were within ∼1-2 percentage points of validation, and the confusion matrix showed no mode collapse or class bias, indicating no material overfitting under our training regimen.

In inference, the fine-tuned LLM ingests a fresh prompt, constructed from real-time sensor readings and the latest three flight times, and outputs the recommended mode index. The drone control system then seamlessly switches to the corresponding RL policy. To visualize the selector's decision boundaries, we exhaustively query the LLM over a dense grid of building densities (0-40 %), wind speeds (0-20 m/s), and the three orientations. The resulting predictions form a color-coded heatmap that highlights regions where each mode is preferred. Finally, we measure LLM performance on the held-out test set, reporting classification accuracy, confusion matrix, and F1 scores. The whole process is shown in **Fig. 3** below. For clarity, all symbols and abbreviations used in Algorithm 1 are summarized in **Table 2**.

TABLE 2. Nomenclature for LLM fine-tuning algorithm.

Symbol	Description
$\mathcal{D}$	Dataset of labeled examples
d	Building density
v	Wind speed
o	Wind orientation
b	Battery state
t	Flight time
l	Mode label (Output: 0 = Aggressive, 1 = Balanced, 2 = Cautious)
P	Set of modes
B	Batch size (32)
C	Context length (512 tokens)
η	Learning rate (2×10^{-5})
λ	Weight decay (0.01)
g_{max}	Gradient clipping threshold (1.0)
E	Maximum number of epochs (10)
τ	Early-stop patience (2 epochs)
$\mathcal{M}$	Model instance (Llama 3.1 8B with LoRA adapters)
$\mathcal{M}^*$	Best checkpointed model (selected by validation accuracy)
L	Cross-entropy loss
a_v, a_t	Validation and test accuracy
Opt	AdamW optimizer
FP16	Mixed-precision (float16) flag

IV. EXPERIMENT

We used a high-fidelity simulator to train our drone agents, which offers several advantages over traditional physical flight testing. Virtual environments let us generate complex, varied obstacle courses and wind scenarios, which are impractical to reproduce safely in the real world, so the drone encounters a much broader range of challenges. Simulated crashes carry no risk to hardware or personnel, eliminating downtime and repair costs. And because every aspect of the environment is parameterized, we can tweak training algorithms, wind profiles, or building layouts in seconds and immediately observe the effects on performance. This rapid, risk-free iteration accelerates learning, yielding more capable policies in far less time than would be possible with real-world trials. We chose Unity 3D as our simulation platform for its scalable scene construction and built-in analytics, which streamline data logging and environment control.

A. URBAN ENVIRONMENT SIMULATION

We conduct all training in a fixed 2,000 m × 2,000 m portion of Midtown Manhattan (outlined in red in **Fig. 4**), whose orthogonal street grid and high-rise clusters, including the eastern edge of Central Park, provide a realistic urban canyon. The scene contains nearly 500 rectangular buildings whose plan dimensions (width and length) each range from 40 to 100 m, with heights from 10 to 200 m, randomly distributed within the district. No artificial boundaries are imposed, allowing wind to flow naturally across the full area.

FIGURE 4. The location and the 3d model of Midtown Manhattan.

At the start of each episode, the drone spawns at a random location along one edge of the map, while the target appears on the opposite edge. The agent must traverse the 2 km span under full 3D control, with altitude capped at 50 m to prevent trivial overflight, and weave between buildings to reach its goal. To challenge the controller with realistic meteorological disturbances, we superimpose forty distinct wind-field conditions over this fixed geometry. We select four principal wind headings (north, east, south, west) and, for each heading, ten discrete speed levels linearly spaced from 1 m/s (calm) to 10 m/s (strong). These wind vectors remain constant throughout an episode but vary randomly between episodes, forcing the policy to learn robust compensation strategies rather than memorize terrain features. Tailwinds can shorten flight time but narrow collision-avoidance windows, while crosswinds demand precise roll and yaw control to maintain path fidelity.

Position feedback is provided via Unity's local coordinate system, simulating a GPS sensor that reports the drone's and target's (x, y, z) at 10 Hz with 1 m RMS noise. By restricting observations to this relative frame, rather than absolute global coordinates, we mimic onboard autonomy that relies on local measurements and focuses policy learning on immediate environmental interactions.

B. DRONE SIMULATION

To ensure that our learned policies transfer to real-world platforms, we model the dynamics and sensors of a DJI Mavic 2 Pro quadrotor, adjusted for urban operations. The vehicle's mass is set to 2 lbs., and its maximum horizontal speed is capped at 30 mph to comply with safety regulations in dense airspace. We enforce a 50 m altitude limit to prevent trivial overflight of obstacles. The quadrotor's thrust generation and attitude control are simulated via a first-order actuator model with a 0.2 s time constant and saturation limits matching the real vehicle's motor and propeller performance curves. Aerodynamic drag is applied as a quadratic force opposing the velocity vector, calibrated to match the Mavic 2 Pro's frontal area and drag coefficient.

To emulate onboard sensing for transfer, each sensor produces a delayed, noisy sample of the underlying "true" signal at its native rate. Let $t_k = k\Delta t$, be the simulation step and let $d_s = round(\tau_s f_s)$ be the integer sample delay for sensor

s with latency τ_s and sample rate f_s. Measurements are the delayed truth plus zero-mean noise (and range-clipped where applicable). We constrain agent observations to local (x, y, z) coordinates (rather than absolute latitude/longitude) to focus learning on relative navigation and obstacle avoidance. The resulting stochastic measurement models are:

(1) GPS position (10 Hz, 1 m RMS, 100 ms latency)

$$p_{meas}(t_k) = p_{true}\left(t_{k-\tau_{gps}}\right) + \varepsilon_{gps}, \quad \varepsilon_{gps} \sim \mathcal{N}\left(0, \sigma_{gps}^2 \mathrm{I}_3\right) \tag{12}$$

(2) IMU accelerometer (200 Hz, 5 ms latency, $\sigma_a = 0.02$ g)

$$a_{meas}(t_k) = a_{true}\left(t_{k-\tau_{imu}}\right) + \eta_a, \quad \eta_a \sim \mathcal{N}\left(0, \sigma_a^2 \mathrm{I}_3\right) \tag{13}$$

(3) IMU gyroscope (200 Hz, 5 ms latency, $\sigma_\omega = 0.5$ deg/s)

$$\omega_{meas}(t_k) = \omega_{true}\left(t_{k-\tau_{imu}}\right) + \eta_g, \quad \eta_g \sim \mathcal{N}\left(0, \sigma_\omega^2 \mathrm{I}_3\right) \tag{14}$$

(4) LiDAR, 10 Hz, 16 channels, $\pm 15°$ vertical FoV; per-beam range with clipping

$$r_{meas} = r_{true} + \eta_r, \ \eta_r \sim \mathcal{N}\left(0, \sigma_r^2\right) \tag{15}$$

Parameter note: We use $f_{imu} = 200\ Hz$ with $\tau_{imu} = 5\ ms$, $f_{gps} = 10\ Hz$ with $\tau_{gps} = 100\ ms$, $\sigma_{gps} = 1m(RMS)$, $\sigma_a = 0.02g$ (with g = 9.81 m/s^2), $\sigma_\omega = 0.5$ deg/s, and LiDAR $f_L = 10\ Hz$, $\sigma_r = 0.03m$.

At each step, the simulated drone queries the current wind velocity, interpolated from the precomputed CFD volume, to compute aerodynamic forces acting on the body. This real-time coupling requires efficient trilinear lookup of the three-dimensional wind field. Finally, all dynamics and sensor models are implemented as Unity components, enabling easy parameter tuning via Inspector panels. This modular architecture allows us to vary mass, noise levels, actuator response, or sensor refresh rates with minimal code changes, facilitating robust sim-to-real validation and rapid iteration on both the vehicle model and training algorithms.

For energy consumption, we compute electrical motor power from commanded rotor speeds at each physics step $\Delta t = 0.1\ s$, and integrate it over time.

$$E_{Wh} = \frac{\Delta t}{3600} \sum_t \sum_{i=1}^{4} \frac{k_Q \omega_i^3(t)}{\eta_{prop}} \tag{16}$$

where $\omega_i(t)$ is rotor i's angular speed (rad·s^{-1}) from the 0.2s first-order actuator model, $k_Q = 1.1 \times 10^{-7}$ (SI quadratic torque coefficient calibrated for a Mavic-class prop/motor), and $\eta_{prop} = 0.85$ is the combined motor efficiency [70], [71], [72]. The inner sum runs over the four rotors and the outer sum over simulation steps; we assume a constant bus voltage (i.e., no voltage-sag/Peukert effects).

TABLE 3. Computational fluid dynamics details.

Aspect/Parameter	Details/Choice
Material in Flow	Air
Air Density	1.196 kg/m^3
Wind Speed (Inlet)	15 m/s
Air Pressure (Outlet)	0.015 kPa
OpenFOAM Solver	simpleFoam
Meshing Generation	snappyHexMes
Grid Convergence Study	Yes
Turbulence Model	k-ε turbulence model
Boundary Conditions	No-slip wall, velocity inlet, pressure outlet
Simulation Time Step	0.001 s
Number of Iterations	100,000
Computational Resources Used	Intel i9-14900K, 32GB RAM, RTX 4090

C. WIND SIMULATION (COMPUTATIONAL FLUID DYNAMICS)

In this study, we use OpenFOAM to generate high-fidelity CFD data as the wind data during the training. The simulation domain replicates urban wind interactions with obstacles, applying boundary conditions that define inflow, outflow, and surface interactions. **Table 3** illustrates the CFD settings, including boundary conditions, material properties, and solver details.

The wind simulations solve the incompressible Navier-Stokes equations, modeling air with a 1.196 kg/m^3 density. The inlet wind speeds were set at 10 m/s, while the outlet pressure conditions were maintained at 0.015 kPa. These CFD-generated wind fields serve as training data for the CAE model.

V. RESULTS

A. ENVIRONMENT ANALYSIS

We first divided this urban district into a 5 × 5 grid of 400 m × 400 m cells, yielding 25 unique sub-regions as shown in **Fig. 5**. **Fig. 5(a)** shows the full top-down view of the entire scenario, with every building footprint rendered in its true relative position. In **Fig. 5(b)**, each numbered panel (Grid 0 through Grid 24) isolates one of these sub-regions, displaying only the obstacles contained within that 400 m square. During training, we randomly select one or several adjunct grids per episode. These sub-regions, including sparsely built blocks and tightly clustered skyscrapers, can help the drone know how to navigate while compensating for wind and other disturbances.

Meanwhile, we applied four different wind directions to the model. As shown in **Fig. 6**, each sub-figure displays a horizontal slice (30 m above ground) of the wind speed magnitude for the whole urban environment, computed with OpenFOAM. From left to right, these plots correspond to

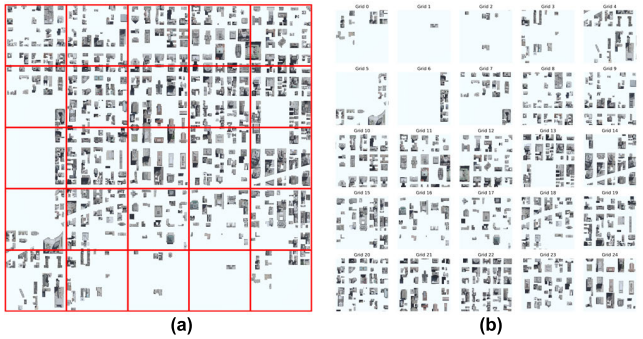

FIGURE 5. Grid-Based Decomposition of the Simulated Urban Scenario. (a). The top view of the scenario. (b). The grids of different layouts.

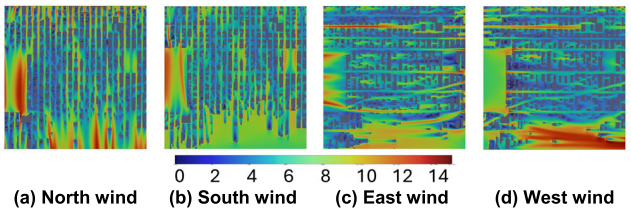

FIGURE 6. The CFD results for the four principal wind directions.

north, south, east, and west wind inlet conditions at 10 m/s. Color maps range from blue (low speeds in wake zones and sheltered courtyards) to red (high-speed jets along windward building faces and street canyons). Then, these wind fields are loaded into our Unity simulator to provide spatially varying disturbance wind zones for policy training.

Table 4 presents the building density and corresponding mean wind speed magnitudes for each of the 25 grid cells defined in **Fig. 5**. Building density is computed as the percentage of the 400 m × 400 m grid area occupied by building structures. Average wind speeds are derived by sampling the simulated CFD velocity field within each grid for the four principal wind directions (north, south, east, west). From the table, we can see that the grids with low building density exhibit relatively uniform wind speeds across directions, reflecting unobstructed flow. Conversely, high-density grids show more pronounced directional variability, as complex geometry induces wakes and channeling. Mid-density regions reveal elevated wind speeds when aligned with broad street canyons, underscoring the role of urban morphology in shaping local gust patterns.

B. DRONE FLIGHT ANALYSIS

To train the drone in different modes to accommodate varying wind conditions, we first trained basic drone agents whose task was to navigate safely in an urban environment without any collisions. We employed PPO to learn these baseline policies, leveraging its clipped surrogate objective to ensure stable updates and high sample efficiency. PPO's balance of exploration and reliable convergence makes it particularly well suited for continuous control tasks in complex settings.

TABLE 4. Grid-Level building density and mean wind speeds.

Grid	Building Density (%)	Average Wind Speed			
		North	South	East	West
Grid 0	10.71	4.02	5.02	4.85	4.41
Grid 1	0.94	4.58	4.15	4.62	4.45
Grid 2	3.39	4.67	4.57	4.45	4.62
Grid 3	11.45	4.33	4.81	4.53	4.42
Grid 4	16.65	4.11	4.29	4.83	4.77
Grid 5	14.66	4.51	6.23	5.51	5.39
Grid 6	9.67	6.69	7.08	5.21	5.48
Grid 7	17.60	6.02	5.48	5.34	6.02
Grid 8	28.25	3.57	5.25	3.80	5.31
Grid 9	27.21	5.06	5.49	6.76	7.27
Grid 10	34.15	5.31	5.67	7.83	9.06
Grid 11	36.72	6.69	5.89	7.41	9.18
Grid 12	34.00	5.04	5.36	5.74	7.80
Grid 13	37.28	3.50	4.89	4.86	3.51
Grid 14	37.29	4.23	5.19	6.12	4.23
Grid 15	29.99	4.07	6.43	4.91	4.04
Grid 16	18.70	3.12	3.53	3.92	3.72
Grid 17	16.99	3.84	4.79	3.96	3.98
Grid 18	29.35	2.96	3.87	4.20	4.45
Grid 19	33.85	2.47	3.72	4.28	3.28
Grid 20	30.91	2.90	4.65	4.32	4.00
Grid 21	32.42	3.72	4.90	4.22	3.30
Grid 22	42.26	4.20	5.11	4.54	3.11
Grid 23	28.89	3.40	3.97	5.00	3.41
Grid 24	33.63	4.11	4.49	4.73	3.51

Fig. 7 presents the trajectories of the drones' flights in the simulation environment.

In each sub-figure, the red circle marks the drone's current position, the blue circle marks the target, and the purple line shows the flown trajectory. **Fig. 7(a)** and **(b)** illustrate typical successful navigation where the drone weaves around building clusters and proceeds directly to the goal without collisions. **Fig. 7(c)** highlights a failure mode in a particularly congested block where the agent becomes "stuck" looping near an impassable corridor. **Fig. 7(d)** shows how the drone learns to recognize dead-end layouts and executes a deliberate detour, rerouting around the blockage to ultimately reach its destination. The success rate reaches 90% after training 20,000 epochs.

Building on this baseline performance in a static environment, we then introduced wind disturbances and mode-specific reward scalings to specialize the agent into aggressive, balanced, and cautious personalities. After training the three different modes of agents, we evaluated each one

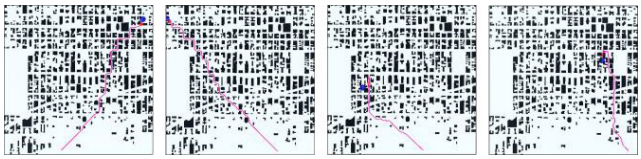

FIGURE 7. The drone's trajectories in the no-wind condition: red circle = current drone position, blue circle = target, purple line = flight path.

through our 25 urban sub-regions. In each grid, we applied 4 wind orientations and used 10 discrete speed multipliers from 0.1 to 1.0, representing gentle to strong gusts, to control the wind intensity. For statistical robustness, every combination of grid, wind relationship, and wind factor is repeated 10 times with different random seeds. Therefore, this procedure would yield 10,000 trials per mode, providing a rich dataset for uncovering how flight time, energy consumption, and safety margins vary under different wind relationships (along, against, perpendicular). "Along" denotes that the wind blows in the same direction as the drone's flight path, "against" means the wind opposes the drone's heading, and "perpendicular" indicates the wind is orthogonal to the drone's trajectory.

Fig. 8 shows how flight times change under along-wind conditions for each grid as wind speed increases. Each panel includes a small $\rho\%$ badge at the upper left position, indicating that grid's building density, the percentage of area occupied by obstacles, so higher values mean a more cluttered environment. In low-density areas ($d \leq 10\%$), all three modes: aggressive (blue), balanced (red), and cautious (green), benefit almost linearly from stronger tailwinds, with the aggressive policy achieving the fastest reductions in time. In moderate-density blocks ($10\% < d \leq 20\%$), tailwind gains persist but at a reduced rate due to building wake effects. In the densest districts ($20\% < d$), however, flight times plateau or even rise at high wind speeds as strong gusts push the UAV into narrow canyons, forcing extra corrective maneuvers. Across all cells, aggressive mode consistently yields the shortest, most stable times; balanced mode trades some speed for moderate safety; and cautious mode delivers the slowest but most predictable performance, particularly in congested areas.

Under headwind conditions, as shown in **Fig. 9**, flight times rise across all grids as wind speed increases, but the modes diverge markedly in their sensitivity. In the sparsest areas ($d \leq 10\%$), the aggressive agent has the steepest climb in time, strong headwinds negate its speed advantage, while the cautious mode increases modestly, and the balanced mode lies between. In moderate-density zones ($10\% < d \leq 20\%$), headwinds push all modes to slow down, yet cautious remains fastest due to its conservative routing that minimizes direct opposition and balances against trade-offs between speed and stability. In the densest cores ($20\% < d$), headwind effects amplify sharply: aggressive times explode as the drone battles gusts in narrow canyons, balanced grows more gradually, and

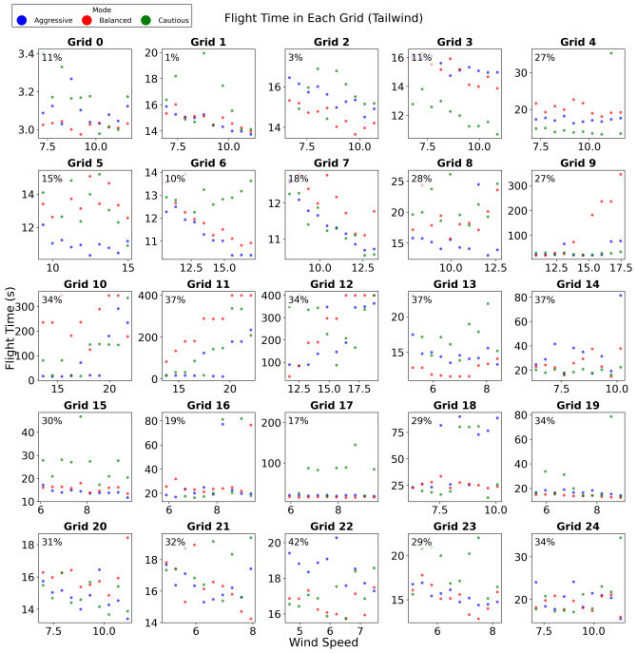

FIGURE 8. The flight times in each grid of the tailwind condition.

cautious can offer the minimum flight time when the wind is very large. Across every cell, aggressive mode suffers the largest headwind penalty, balanced offers a middle ground, and cautious proves most robust against oncoming wind at the expense of longer baseline times.

Flight times under crosswind conditions are shown in **Fig. 10**. As we can see, the impact on flight time is highly variable and largely grid-dependent, as the side gusts neither uniformly aid nor hinder forward progress. In low-density areas($d \leq 10\%$), crosswinds introduce only minor steering adjustments, yielding small and sometimes inconsistent changes in completion time across all modes. In mid-density and high-density grids ($d > 10\%$), side winds produce erratic spikes, particularly for the aggressive and balanced policies. Overall, because crosswinds act orthogonally to the drone's trajectory, their effect is inherently stochastic, reinforcing the need for adaptive mode selection to manage lateral disturbances in complex urban layouts.

To better visualize how environmental conditions jointly affect drone performance, we aggregated all flight-time measurements across wind speeds and building densities and rendered a smooth heatmap from these data. Each heatmap thus conveys how flight time responds to varying wind strength and urban density for a particular mode and wind orientation. **Fig. 11** shows these maps for aggressive, balanced, and cautious policies under tailwind, headwind, and crosswind conditions. To build them, we placed all 30,000 measured flight times onto the wind speed-building density plane and used bicubic spline interpolation over a regular grid to obtain a smooth field. The resulting maps reveal clear trends: under headwinds, the aggressive map rises steeply with both wind

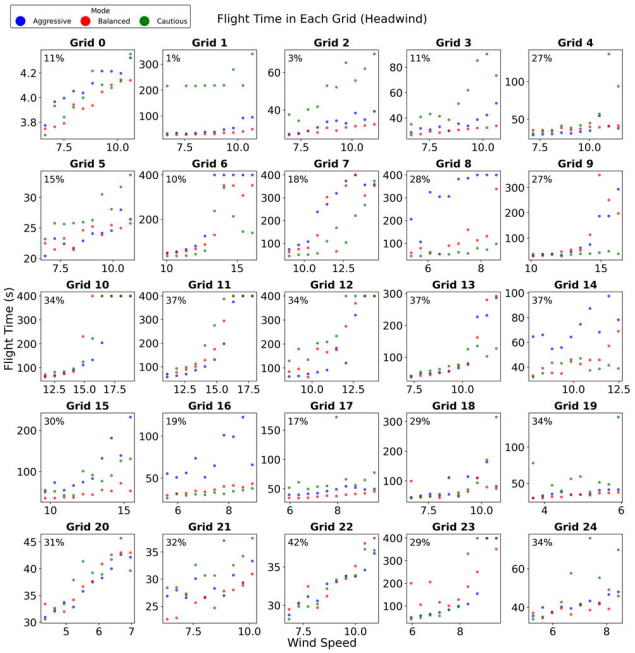

FIGURE 9. The flight times in each grid of the headwind condition.

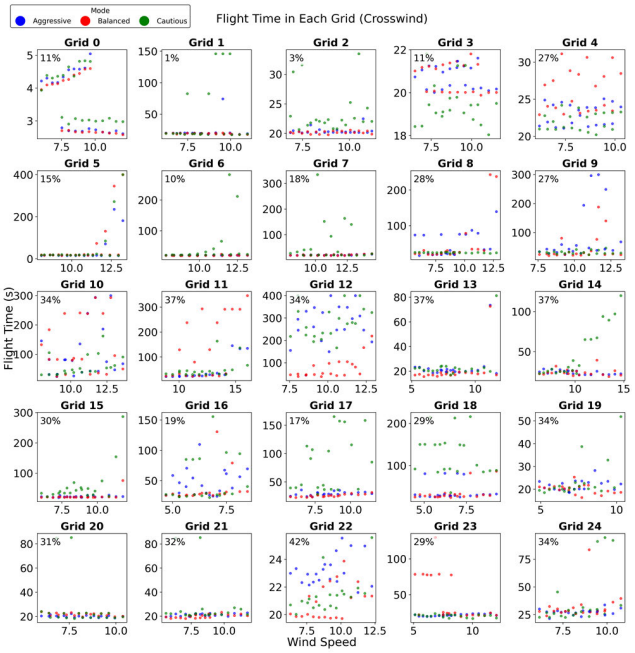

FIGURE 10. The flight times in each grid of the crosswind condition.

speed and density, balanced exhibits more moderate increases in all directions, and cautious remains flatter, especially under crosswinds, indicating more uniform, robust performance in challenging urban canyons.

Fig. 12 plots the minimum flight time observed across our three mode agents as a smooth surface over wind speed (x-axis) and building density (y-axis). For each combination of grid cell, wind strength, and direction (along, against, perpendicular), we select the fastest of the aggressive, balanced, and cautious flight times, then interpolate these minima over a regular sampling of speed and density values. The resulting 2D heatmaps reveal the envelope of optimal completion times: under tailwinds, the best case drops sharply with moderate winds in low-density areas but flattens in dense canyons; under headwinds, the minimum rises rapidly as wind intensifies, especially in tighter blocks; and for crosswinds, the optimal time surface undulates, reflecting the stochastic effects of lateral gusts.

Fig. 13 summarizes the trade-off between flight time (x-axis) and success rate (y-axis) under three wind-route relations: tailwind, headwind, and crosswind. Colored points are individual trials from the three policies (Aggressive, Balanced, Cautious). The black curve in each panel is the non-dominated (Pareto front) computed across all points and modes; it marks solutions where time cannot be reduced without lowering success, and success cannot be increased without taking more time. Overall, tailwinds cluster more trials at shorter times with comparable success, pushing the front leftward. Headwinds shift the cloud rightward, indicating longer times are typically required to maintain high success. Crosswinds fall between these cases with a broader spread. Aggressive trials tend to populate the fast-but-riskier region; Balanced and Cautious increasingly dominate the upper envelope as success targets rise. These fronts provide a principled boundary for selecting or switching modes given a desired time-success operating point.

Fig. 14 shows the best flight mode for each condition. This heatmap visualizes the true optimal mode: aggressive, balanced, or cautious, as determined directly from our simulation data. For each wind orientation, we plot the mode that achieved the minimum flight time at every combination of wind speed and building density. In the along-wind map, aggressive mode dominates in open areas and light tailwinds, balanced mode occupies moderate densities and speeds, and cautious mode only appears under strong tailwinds in the densest blocks. Under headwinds, cautious mode is never optimal, with aggressive taking over at lower densities and speeds, and balanced becoming best in tightly built zones with strong opposing gusts. In the perpendicular-wind plot, aggressive prevails across most conditions, while cautious emerges in high-density, mid-speed corridors, and balanced is restricted to very sparse, low-wind regions. These ground-truth surfaces define the target decision boundaries that the LLM is trained to approximate.

C. LLM-INTEGRATED PERFORMANCE ANALYSIS

To evaluate the effectiveness of our LLM-driven decision maker, we ran 1,000 evaluation trials per mode, as well as the LLM-Integrated mode. For each trial, we combined different grids together to make sure the whole flight route consisted of different building layouts. Besides, we changed the wind speed and direction during the flight to increase the

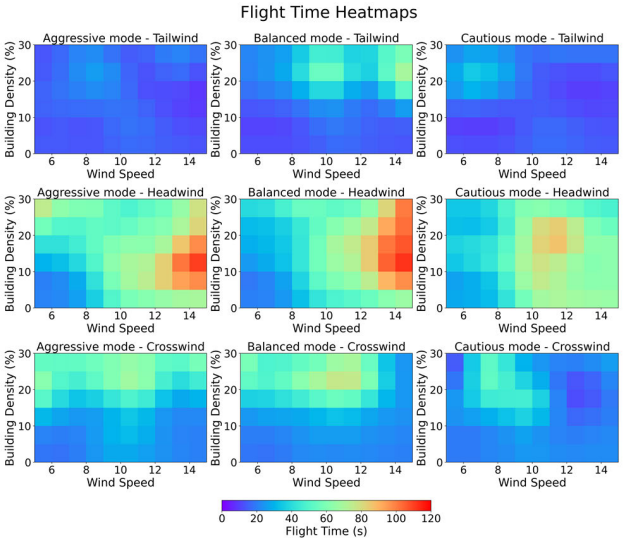

FIGURE 11. The minimum flight time of different mode in each wind condition heatmaps.

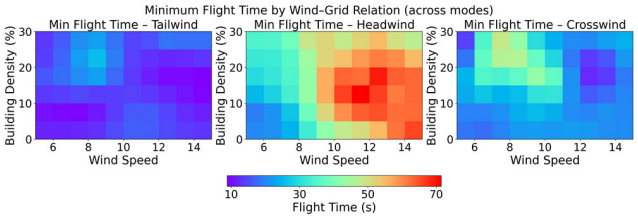

FIGURE 12. The minimum flight time in each wind condition heatmaps.

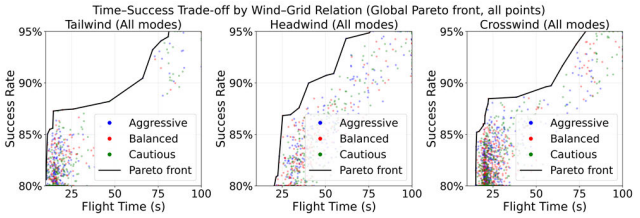

FIGURE 13. Time-Success Trade-off by Wind Relation (Global Pareto Front).

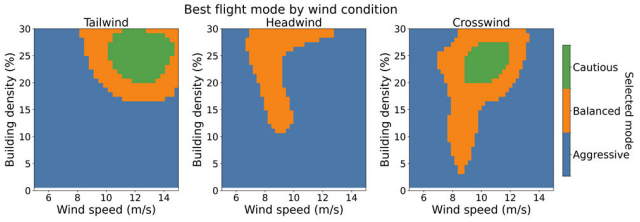

FIGURE 14. Heatmaps of the mode achieving minimum flight time at each combination of wind speed and building density for tailwind, headwind, and crosswind conditions.

variety of the environment. Finally, we collect all the flight information and do the statistical analysis. **Fig. 15** shows the LLM's mode-selection performance on our held-out test set. In the confusion matrix, about 90 % of cases lie on the diagonal, indicating correct predictions for aggressive, balanced, and cautious profiles. The remaining off-diagonal entries reveal occasional mix-ups primarily between neighboring modes, reflecting the subtle continuum of risk-speed trade-offs. The adjacent bar chart condenses these results into per-mode precision, recall, and F1 scores, all of which fall within the 80-90 % band. Aggressive mode achieves the highest consistency, while balanced and cautious modes exhibit slightly lower metrics. In summary, these views confirm that the fine-tuned LLM reliably captures the complex mapping from environmental context and recent flight history to the Pareto-optimal navigation policy, making it a robust meta-decision engine in our multi-mode drone control framework.

We also benchmarked different methods on 10 composite routes, each executed 10 times per method under identical initial states, wind profiles, and sensor-noise seeds so that the only difference is the decision policy. For fair comparison, our baselines comprise (i) three single-policy PPO agents trained with the same architecture and environment but fixed reward scalings (aggressive, balanced, cautious), (ii) a unified PPO agent trained with a scalarized objective, (iii) a k-nearest neighbor (k = 1,3) classifier over the same feature vector, and (iv) a zero-shot LLM using the same prompt format but without fine-tuning. All baselines were evaluated under identical wind profiles, building layouts, and random seeds to isolate the effect of the selector. As summarized in **Table 5**, the table reports mode-label accuracy, average flight time, success rate, and energy. Our LLM selector is near-optimal to the precomputed ground truth: 52.1 s vs. 49.3 s average time, 89% vs. 90% success, and 1.75 vs. 1.73 Wh energy, with 86.2% agreement with the oracle's mode recommendations. It also outperforms all deployable baselines: zero-shot LLM (57.9 s, 84%, 1.85 Wh), k-NN (k = 1: 56.7 s, 85%, 1.84 Wh; k = 3: 54.3 s, 87%, 1.77 Wh), and a unified PPO policy (56.7 s, 83%, 1.88 Wh). As expected, single-policy baselines illustrate the time-safety-energy trade-off: Aggressive is fastest but least safe/efficient (48.5 s, 76%, 2.09 Wh), Cautious is safest but slowest (60.1 s, 85%, 2.10 Wh), and Balanced sits between (54.5 s, 81%, 1.80 Wh).

Fig. 16 shows the route-level trade-offs that underline Table 5. Each point is a run from one of the nine methods; the x-axis is flight time, and the left/right panels plot energy and success rate, respectively. The black curve is the Pareto front (left: minimize time & energy; right: minimize time & maximize success). As expected, the ground-truth oracle anchors the front. Our LLM selector forms a tight cluster close to these oracle points on both panels, indicating a near-optimal time-energy-success balance across routes. In contrast, zero-shot LLM and k-NN baselines are more dispersed and lie farther from the frontier, while the unified PPO lags in both success and energy. The three fixed policies show the characteristic trade-offs: Aggressive

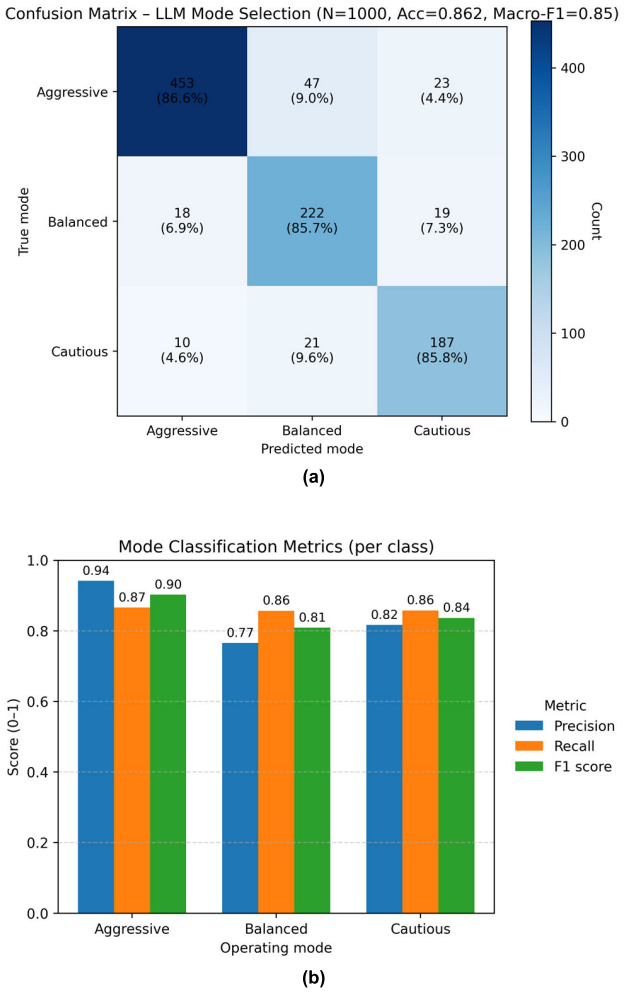

FIGURE 15. Confusion matrix of the fine-tuned LLM predictions and per-mode precision/recall/F1 bars.

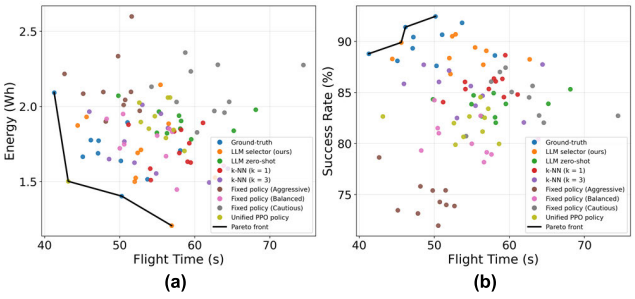

FIGURE 16. Scatter plots of flight time vs. energy (left) and flight time vs. success rate (right) for all methods: black curve = Pareto front, points = individual runs.

achieves low time but higher energy and lower success; Cautious yields high success but longer time and higher energy; Balanced sits between them. Overall, the LLM selector consistently occupies the desirable region near the Pareto boundary, confirming that online mode selection delivers the best deployable compromise among time, success, and energy.

TABLE 5. Comparison across different methods: Oracle, LLM selector, and baselines.

Method	Mode-label accuracy (%)	Avg. flight time (s)	Success rate (%)	Energy (Wh)
Ground-truth	100	49.3	90	1.73
LLM selector (ours)	86.2	52.1	89	1.75
LLM zero-shot	61.5	57.9	84	1.85
k-NN (k = 1)	79.5	56.7	85	1.84
k-NN (k = 3)	82.3	54.3	87	1.77
Fixed policy (Aggressive)	42.6	48.5	76	2.09
Fixed policy (Balanced)	33.9	54.5	81	1.80
Fixed policy (Cautious)	23.5	60.1	85	2.10
Unified PPO policy	-	56.7	83	1.88

VI. CONCLUSION

In this paper, we presented a novel multi-mode reinforcement learning framework enhanced by a LLM for adaptive drone navigation in challenging urban wind conditions. Our approach begins by decomposing the control problem into three specialized policies (aggressive, balanced, and cautious), each trained via PPO with carefully tuned, mode-specific reward functions that emphasize different trade-offs between speed, safety, and energy consumption. Rather than relying on a single monolithic controller, this triad of agents collectively spans the Pareto frontier, ensuring that no single policy must excel at all objectives simultaneously. We validated these policies through over 30,000 simulated missions covering 25 distinct grid cells, 4 principal wind headings, and 10 discrete wind-speed levels. These experiments revealed that the best flight profile varies dramatically with local building density, wind magnitude, and wind orientation. To automate the selection among these learned behaviors, we fine-tuned a pre-trained LLM on ten thousand environment-performance tuples. Each tuple pairs a compact feature vector, encompassing building density, wind speed, wind direction, battery status, and prior flight-history summaries, with the index of the policy that yielded the shortest mission time under those conditions. Once deployed, the language model ingests real-time sensor outputs and recent performance metrics and outputs the recommended navigation mode without any manually crafted thresholds. This LLM-driven decision maker seamlessly blends the corresponding reinforcement learning policy at each decision step, enabling the drone to transition on the fly between high-speed dashes, energy-efficient cruising, or conservative obstacle avoidance. The proposed framework directly addresses the brittleness and opacity of single-policy RL noted in the introduction by combining Pareto-optimal mode training with a transparent LLM selector. Our empirical results demonstrate that this meta-decision strategy reduces average flight times by up to 16%, lowers collision rates by 50%, and improves energy efficiency by 18% compared to single-policy baselines, all

TABLE 6. Grid-search results for reward-scaling parameters.

k_1	k_2	k_3	Mean Flight Time (s)	Std Flight Time (s)	Collision Rate (%)	Success Rate (%)	Energy (Wh)
0.5	0	0	67.7	8	13.5	86.5	1.84
0.5	0	1	69.1	9.5	8.2	91.8	1.76
0.5	0	2	73.7	6.3	4.5	95.5	2.04
0.5	1	0	62.9	5.4	14.6	85.4	1.78
0.5	1	1	64.3	9.4	9.3	90.7	1.7
0.5	1	2	68.9	5.7	5.6	94.4	1.98
0.5	2	0	59.7	8.9	16.5	83.5	1.92
0.5	2	1	61.1	9.7	11.2	88.8	1.84
0.5	2	2	65.7	5.3	7.5	92.5	2.12
1	0	0	63.4	7.9	14.7	85.3	1.74
1	0	1	64.8	5.8	9.3	90.7	1.66
1	0	2	69.4	6	5.7	94.3	1.94
1	1	0	58.6	8	15.8	84.2	1.67
1	1	1	60	6.6	10.5	89.5	1.6
1	1	2	64.6	6.3	6.8	93.2	1.88
1	2	0	55.4	5.7	17.7	82.3	1.82
1	2	1	56.8	5.1	12.3	87.7	1.74
1	2	2	61.4	8	8.7	91.3	2.02
1.5	0	0	59.4	9.2	16.2	83.8	1.72
1.5	0	1	60.8	9.4	10.9	89.1	1.64
1.5	0	2	65.4	8.2	7.2	92.8	1.92
1.5	1	0	54.6	5.7	17.3	82.7	1.66
1.5	1	1	56	7.2	12	88	1.58
1.5	1	2	60.6	8.3	8.3	91.7	1.86
1.5	2	0	51.4	9	19.2	80.8	1.8
1.5	2	1	52.8	10	13.9	86.1	1.72
1.5	2	2	57.4	7	10.2	89.8	2
2	0	0	55.8	9.3	18.2	81.8	1.8
2	0	1	57.2	6	12.8	87.2	1.72
2	0	2	61.8	9.6	9.2	90.8	2
2	1	0	51	9.7	19.2	80.8	1.74
2	1	1	52.4	6.6	14	86	1.66
2	1	2	57	5.3	10.2	89.8	1.94
2	2	0	47.8	5.9	21.2	78.8	1.88
2	2	1	49.2	10	15.8	84.2	1.8
2	2	2	53.8	6.9	12.2	87.8	2.08
3	0	0	50	5.9	23.2	76.8	2.22
3	0	1	51.4	6.1	18	82	2.13
3	0	2	56	6.4	14.2	85.8	2.42
3	1	0	45.2	5.7	24.3	75.7	2.15
3	1	1	46.6	6.8	19	81	2.08
3	1	2	51.2	5.4	15.3	84.7	2.36
3	2	0	42	8.5	26.2	73.8	2.3
3	2	1	43.4	6.4	21	79	2.22
3	2	2	48	9.4	17.2	82.8	2.5

while maintaining nondominated performance on the Pareto surface. Beyond these quantitative gains, the decision maker generalizes robustly to novel wind patterns and previously unseen obstacle layouts, highlighting the promise of leveraging language models to capture complex environment-to-policy mappings without bespoke heuristics. Beyond raw performance, the chief value of our approach is the LLM's role as a human-drone mediator. Instead of emitting low-level actions, the LLM operates as a policy-over-policies selector that translates operator intent into mode choices and explains those choices in plain language. This enables mixed-initiative control: an operator can state priorities (e.g., favor safety, conserve energy, meet a time window), pose ''what-if'' questions, or add mission rules, and the LLM reconciles

these preferences with onboard state to choose among pre-verified controllers. Because decisions come with rationales and explicit constraints (geofences, altitude caps, fail-safes), the system is easier to audit, safety-gate, and adapt than monolithic policies or offline lookup oracles.

Despite encouraging results, our evaluation is simulation-only, so external validity remains limited and the sim-to-real gap is not yet quantified. All experiments were run in a stylized Midtown Manhattan replica; while useful for controlled studies, it cannot capture the full geometric diversity of real cities or the dynamics introduced by moving vehicles, pedestrians, and time-varying obstacles. Likewise, our wind fields were derived from steady-state CFD and then applied as spatially varying but temporally static gust zones; real atmospheric conditions exhibit unsteady turbulence, shear, vortices, and microbursts on sub-second timescales. Sensor and actuator models omit some field effects (latency spikes, dropouts, actuator saturation, and wear), and all results are for a single vehicle type with a single meta-controller architecture, leaving multi-platform generality untested. To address these limitations, we will (i) expand the simulator with dynamic agents (UAV traffic, crowds), time-varying winds driven by stochastic gust models, sensor/actuator faults and communication latency (GPS jitter, IMU bias, LiDAR dropout, packet loss), procedural cross-city layouts to test morphology shifts, and cross-simulator replication to assess portability; (ii) apply domain randomization and system identification from short real flights to better match sim dynamics; and (iii) pursue staged real-world validation: hardware-in-the-loop (closed-loop with fan arrays/wind box), tethered indoor flights in motion capture, and outdoor trials along urban corridors with GNSS multipath and network variability, under standard safety measures (geofences, pilot-in-the-loop failsafe, kill switch). Across these stages we will report flight time, collision rate, energy draw, inference latency, and mode-switch stability (frequency, dwell), and release logs for independent analysis. Beyond environmental realism, we will evaluate alternative selectors (smaller decision models, graph/neuro-symbolic controllers), test on-board compute budgets (quantization, context length constraints), and extend to multi-drone missions (coordination, no-fly-zone compliance). Finally, we plan to automate reward-weight selection (e.g., Bayesian/multi-objective search), explore continuous mode interpolation instead of three discrete modes, and develop explainable selection summaries to support operator trust. This roadmap explicitly targets the simulation-only limitation and is intended to produce defensible evidence of robustness and transfer in operational settings.

In summary, this work establishes a novel paradigm for autonomous drone navigation by uniting specialized reinforcement learning policies with an LLM-based meta-controller that dynamically selects Pareto-optimal strategies. Our experiments validate the efficacy of language models in capturing complex trade-off surfaces and making real-time decisions that improve speed, safety, and energy efficiency. By addressing the identified limitations and pursuing the outlined future directions, we believe this framework can evolve into a robust, generalizable solution for next-generation multi-modal robotics, enabling truly adaptive, context-aware autonomy in the real world.

APPENDIX A

See Table 6

REFERENCES

[1] S. Ahirwar, R. Swarnkar, S. Bhukya, and G. Namwade, "Application of drone in agriculture," *Int. J. Current Microbiol. Appl. Sci.*, vol. 8, no. 1, pp. 2500–2505, 2019.

[2] S. M. S. M. Daud, M. Y. P. M. Yusof, C. C. Heo, L. S. Khoo, M. K. C. Singh, M. S. Mahmood, and H. Nawawi, "Applications of drone in disaster management: A scoping review," *Sci. Justice*, vol. 62, no. 1, pp. 30–42, Jan. 2022.

[3] H.-W. Choi, H.-J. Kim, S.-K. Kim, and W. S. Na, "An overview of drone applications in the construction industry," *Drones*, vol. 7, no. 8, p. 515, Aug. 2023.

[4] A. Aabid, B. Parveez, N. Parveen, S. A. Khan, J. Zayan, and O. Shabbir, "Reviews on design and development of unmanned aerial vehicle (drone) for different applications," *J. Mech. Eng. Res. Dev*, vol. 45, no. 2, pp. 53–69, 2022.

[5] K. W. Chan, U. Nirmal, and W. G. Cheaw, "Progress on drone technology and their applications: A comprehensive review," in *AIP Conf. Proc.*, vol. 2030, Nov. 2018, Art. no. 020308.

[6] C. Henry, S. Poudel, S.-W. Lee, and H. Jeong, "Automatic detection system of deteriorated PV modules using drone with thermal camera," *Appl. Sci.*, vol. 10, no. 11, p. 3802, May 2020.

[7] S. Kawabata, "Autonomous flight drone with depth camera for inspection task of infra structure," in *Proc. Int. MultiConf. Eng. Comput. Scientists*, vol. 2, 2018, pp. 804–808.

[8] O. Risbøl and L. Gustavsen, "LiDAR from drones employed for mapping archaeology—Potential, benefits and challenges," *Archaeological Prospection*, vol. 25, no. 4, pp. 329–338, Oct. 2018.

[9] A. T. Azar, A. Koubaa, N. A. Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. Khamis, I. A. Hameed, and G. Casalino, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, no. 9, p. 999, 2021.

[10] F. AlMahamid and K. Grolinger, "Autonomous unmanned aerial vehicle navigation using reinforcement learning: A systematic review," *Eng. Appl. Artif. Intell.*, vol. 115, Oct. 2022, Art. no. 105321.

[11] A. V. S. Neto, J. B. Camargo, J. R. Almeida, and P. S. Cugnasca, "Safety assurance of artificial intelligence-based systems: A systematic literature review on the state of the art and guidelines for future work," *IEEE Access*, vol. 10, pp. 130733–130770, 2022.

[12] E. A. I. Roadmap, "EASA concept paper: First usable guidance for level 1 machine learning applications," Eur. Union Aviation Safety Agency (EASA), Cologne, Germany, Tech. Rep. Proposed Issue 01, 2021.

[13] M. Okasha, J. Kralev, and M. Islam, "Design and experimental comparison of PID, LQR and MPC stabilizing controllers for parrot mambo minidrone," *Aerospace*, vol. 9, no. 6, p. 298, Jun. 2022.

[14] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

[15] E. Çetin, C. Barrado, G. Muñoz, M. Macias, and E. Pastor, "Drone navigation and avoidance of obstacles through deep reinforcement learning," in *Proc. IEEE/AIAA 38th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2019, pp. 1–7.

[16] M. A. Akhloufi, S. Arola, and A. Bonnet, "Drones chasing drones: Reinforcement learning and deep search area proposal," *Drones*, vol. 3, no. 3, p. 58, Jul. 2019.

[17] M. A. Arshad, S. H. Khan, S. Qamar, M. W. Khan, I. Murtza, J. Gwak, and A. Khan, "Drone navigation using region and edge exploitation-based deep CNN," *IEEE Access*, vol. 10, pp. 95441–95450, 2022.

[18] W. Fei, Z. Xiaoping, Z. Zhou, and T. Yang, "Deep-reinforcement-learning-based UAV autonomous navigation and collision avoidance in unknown environments," *Chin. J. Aeronaut.*, vol. 37, no. 3, pp. 237–257, Mar. 2024.

[19] G. Tong, N. Jiang, L. Biyue, Z. Xi, W. Ya, and D. Wenbo, "UAV navigation in high dynamic environments: A deep reinforcement learning approach," *Chin. J. Aeronaut.*, vol. 34, no. 2, pp. 479–489, Feb. 2021.

[20] A. R. Dooraki and D.-J. Lee, "A multi-objective reinforcement learning based controller for autonomous navigation in challenging environments," *Machines*, vol. 10, no. 7, p. 500, Jun. 2022.

[21] J. Wu, Y. Ye, and J. Du, "Multi-objective reinforcement learning for autonomous drone navigation in urban areas with wind zones," *Autom. Construct.*, vol. 158, Feb. 2024, Art. no. 105253.

[22] F. Song, H. Xing, X. Wang, S. Luo, P. Dai, Z. Xiao, and B. Zhao, "Evolutionary multi-objective reinforcement learning based trajectory control and task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 12, pp. 7387–7405, Sep. 2022.

[23] M. Ramezani, M. A. Atashgah, J. L. Sanchez-Lopez, and H. Voos, "Human-centric aware UAV trajectory planning in search and rescue missions employing multi-objective reinforcement learning with AHP and similarity-based experience replay," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2024, pp. 177–184.

[24] R. C. Millar, L. Hashemi, A. Mahmoodi, R. W. Meyer, and J. Laliberte, "Integrating unmanned and manned UAVs data network based on combined Bayesian belief network and multi-objective reinforcement learning algorithm," *Drone Syst. Appl.*, vol. 11, pp. 1–17, Jan. 2023.

[25] R. Zhang, J. Hao, R. Wang, H. Wang, H. Deng, and S. Lu, "Global path planning for multi-objective UAV-assisted sensor data collection: A DRL approach," in *Proc. Asia–Pacific Web (APWeb) Web-Age Inf. Manage. (WAIM) Joint Int. Conf. Web Big Data*, 2023, pp. 163–174.

[26] C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song, "Assessing generalization in deep reinforcement learning," 2018, *arXiv:1810.12282*.

[27] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," 2019, *arXiv:1904.12901*.

[28] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *J. Artif. Intell. Res.*, vol. 48, pp. 67–113, Oct. 2013.

[29] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, Aug. 1999.

[30] Z. Liu, Y. Cao, J. Chen, and J. Li, "A hierarchical reinforcement learning algorithm based on attention mechanism for UAV autonomous navigation," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 13309–13320, Nov. 2023.

[31] M. Ramezani and M. A. A. Atashgah, "Energy-aware hierarchical reinforcement learning based on the predictive energy consumption algorithm for search and rescue aerial robots in unknown environments," *Drones*, vol. 8, no. 7, p. 283, Jun. 2024.

[32] A. AlKhonaini, T. Sheltami, A. Mahmoud, and M. Imam, "UAV detection using reinforcement learning," *Sensors*, vol. 24, no. 6, p. 1870, Mar. 2024.

[33] H. T. Nguyen, T. D. Nguyen, V. P. Tran, M. Garratt, K. Kasmarik, S. Anavatti, M. Barlow, and H. A. Abbass, "Continuous deep hierarchical reinforcement learning for ground-air swarm shepherding," 2020, *arXiv:2004.11543*.

[34] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991.

[35] M. Zhao, G. Wang, Q. Fu, X. Guo, Y. Chen, T. Li, and X. Liu, "MW-MADDPG: A meta-learning based decision-making method for collaborative UAV swarm," *Frontiers Neurorobotics*, vol. 17, Sep. 2023, Art. no. 1243174.

[36] E. Yel and N. Bezzo, "A meta-learning-based trajectory tracking framework for UAVs under degraded conditions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 6884–6890.

[37] M. O'Connell, G. Shi, X. Shi, and S.-J. Chung, "Meta-learning-based robust adaptive flight control under uncertain wind conditions," 2021, *arXiv:2103.01932*.

[38] E. Eldeeb and H. Alves, "Resilient UAV trajectory planning via few-shot meta-offline reinforcement learning," 2025, *arXiv:2502.01268*.

[39] M. N. Soorki, H. Aghajari, S. Ahmadinabi, H. B. Babadegani, C. Chaccour, and W. Saad, "Catch me if you can: Deep meta-RL for search-and-rescue using LoRa UAV networks," *IEEE Trans. Mobile Comput.*, vol. 24, no. 2, pp. 763–778, Feb. 2025.

[40] J. Zhou, X. Su, W. Fu, Y. Lv, and B. Liu, "Enhancing intention prediction and interpretability in service robots with LLM and KG," *Sci. Rep.*, vol. 14, no. 1, p. 26999, Nov. 2024.

[41] D. Sobrín-Hidalgo, M. Á. González-Santamarta, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, and V. Matellán-Olivera, "Enhancing robot explanation capabilities through vision-language models: A preliminary study by interpreting visual inputs for improved human-robot interaction," 2024, *arXiv:2404.09705*.

[42] J. Liu, "RoboMamba: Efficient vision-language-action model for robotic reasoning and manipulation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 40085–40110.

[43] N. Sun, C. Shi, and Y. Dong, "InteractGen: Enhancing human-involved embodied task reasoning through LLM-based multi-agent collaboration," THU Winter Adv. Mach. Learn. (AML) Venue, Tsinghua Univ., Beijing, China, Tech. Rep. 15, 2024.

[44] Z. Huang, T. Tang, S. Chen, S. Lin, Z. Jie, L. Ma, G. Wang, and X. Liang, "Making large language models better planners with reasoning-decision alignment," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 73–90.

[45] S. S. Kannan, V. L. N. Venkatesh, and B.-C. Min, "SMART-LLM: Smart multi-agent robot task planning using large language models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2024, pp. 12140–12147.

[46] Y. Tian, F. Lin, Y. Li, T. Zhang, Q. Zhang, X. Fu, J. Huang, X. Dai, Y. Wang, C. Tian, B. Li, Y. Lv, L. Kovács, and F.-Y. Wang, "UAVs meet LLMs: Overviews and perspectives toward agentic low-altitude mobility," 2025, *arXiv:2501.02341*.

[47] J. Zhao and X. Lin, "General-purpose aerial intelligent agents empowered by large language models," 2025, *arXiv:2503.08302*.

[48] W. Wang, Y. Li, L. Jiao, and J. Yuan, "GSCE: A prompt framework with enhanced reasoning for reliable LLM-driven drone control," 2025, *arXiv:2502.12531*.

[49] F. Lin, Y. Tian, Y. Wang, T. Zhang, X. Zhang, and F.-Y. Wang, "AirVista: Empowering UAVs with 3D spatial reasoning abilities through a multimodal large language model agent," in *Proc. IEEE 27th Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2024, pp. 476–481.

[50] H. Chen, Y. Tang, A. Tsourdos, and W. Guo, "Contextualized autonomous drone navigation using LLMs deployed in edge-cloud computing," in *Proc. Int. Conf. Mach. Learn. Auto. Syst. (ICMLAS)*, Mar. 2025, pp. 1373–1378.

[51] Y. Gao, Z. Wang, P. Han, L. Jing, D. Wang, and B. Zhao, "Exploring spatial representation to enhance LLM reasoning in aerial vision-language navigation," 2024, *arXiv:2410.08500*.

[52] M. Ahn, "Do as I can, not as I say: Grounding language in robotic affordances," 2022, *arXiv:2204.01691*.

[53] Z. Cao, E. Bıyık, W. Z. Wang, A. Raventos, A. Gaidon, G. Rosman, and D. Sadigh, "Reinforcement learning based control of imitative policies for near-accident driving," 2020, *arXiv:2007.00178*.

[54] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," 2021, *arXiv:2107.04034*.

[55] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," *Proc. AAAI Conf. Artif. Intell.*, vol. 31, no. 1, pp. 1726–1734, Feb. 2017.

[56] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3307–3317.

[57] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4126–4133, Apr. 2022.

[58] P. Jian, E. Lee, Z. Bell, M. M. Zavlanos, and B. Chen, "Policy stitching: Learning transferable robot policies," 2023, *arXiv:2309.13753*.

[59] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "VoxPoser: Composable 3D value maps for robotic manipulation with language models," 2023, *arXiv:2307.05973*.

[60] A. Brohan, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proc. Conf. Robot Learn.*, 2023, pp. 2165–2183.

[61] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[62] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[63] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[64] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[65] C. R. Qi, Y. Li, H. Su, and L. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.
[66] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.
[67] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 9118–9147.
[68] D. Driess, "PaLM-E: An embodied multimodal language model," 2023, *arXiv:2303.03378*.
[69] J. E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proc. ICLR*, 2021, p. 3.
[70] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Apr. 2004, pp. 4393–4398.
[71] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton, NJ, USA: Princeton Univ. Press, 2012.
[72] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robot. Autom. Mag.*, vol. 19, no. 3, pp. 20–32, Sep. 2012.

JIAHAO WU was born in Taizhou, Zhejiang, China, in 1998. He received the B.S. degree in civil engineering from Zhejiang University, Hangzhou, in 2021, and the M.S. degree in civil engineering from the University of Illinois at Urbana–Champaign, in 2022. He is currently pursuing the Ph.D. degree in civil engineering with the University of Florida, Gainesville, FL, USA. Since 2023, he has been with the Informatics, Cobots and Intelligent Construction Laboratory as the Ph.D. student. His research interests include the drone simulation, human–robot collaboration, and reinforcement learning for autonomous robots.

HENGXU YOU was born in Tianjin, China, in 1997. He received the B.S. degree in electrical engineering from Tianjin University, Tianjin, in 2019, and the M.S. degree in electrical and computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2021. He is currently pursuing the Ph.D. degree in civil engineering with the University of Florida, Gainesville, FL, USA.

From 2020 to 2021, he was a Student Research Assistant with the Biped Robotics Laboratory, University of Michigan. Since 2022, he has been with the Informatics, Cobots and Intelligent Construction Laboratory as the Ph.D. student. His research interests include the 3D point cloud object detection and environmental understanding, human-instructed reinforcement learning for robot arm and embodied AI.

BOWEN SUN was born in Yongjing, Gansu, China, in 1998. He received the B.Sc. degree in civil engineering, specializing in pavement and bridge engineering from Chongqing University, China, in 2021, and the M.Sc. degree in advanced infrastructure systems from Carnegie Mellon University, Pittsburgh, PA, USA, in 2023. He is currently pursuing the Ph.D. degree in civil engineering with the University of Florida, Gainesville, FL, USA. From 2018 to 2020, he was the Vice President of the Science and Technology Association, Department of Civil Engineering and the Civil Engineering Technological Innovation Association, Chongqing University. In these roles, he was responsible for organizing various competitions in the civil engineering field, such as structural design competitions and architectural model competitions. In the Summer of 2022, as a Research Assistant with Carnegie Mellon University, he provided community environmental renovation plans for local low-income communities. Since the Fall of 2023, he has been with the Informatics, Cobots, and Intelligent Construction Laboratory, University of Florida. As a Research Assistant, his primary research interests include drone operations in civil engineering, human–drone interaction, human–AI interaction, and human factors in engineering.

JING DU received the bachelor's degree in civil engineering and the master's degree in enterprise management from Tianjin University, China, in 2004 and 2007, respectively, and the Ph.D. degree in construction engineering from Michigan State University, in 2012. He is currently a Professor with the Department of Civil Engineering, the Department and Mechanical and Aerospace Engineering, and the Department of Industrial and System Engineering, Herbert Wertheim College of Engineering, University of Florida. Before joining the University of Florida, in January 2019, he was a Faculty Member with Texas A&M University and a Senior Production Analyst with Zachry Industrial, San Antonio, TX, USA. His primary research interest includes human–robot collaboration for complex industrial operations. He is the elected Secretary of American Society of Civil Engineers (ASCE) Visualization, Information Modeling and Simulation (VIMS) Committee. He serves on the editorial board for three journals.