SUDA: A SIMULATION AND DATA GENERATION PLATFORM FOR UAV DYNAMICS-AWARE POLICY LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-guided UAV control is moving from route planning toward continuous action prediction from onboard observations and instructions. Existing datasets typically supervise policies with waypoint or 6-DoF reference trajectories. Yet a reference specifies intended motion, whereas closed-loop flight is governed by the response produced after commands pass through the flight controller and vehicle dynamics. The resulting state also determines the next onboard observation, making execution-aligned command–response–observation trajectories essential for policy learning. To provide this supervision at scale, we introduce SUDA, a simulation and data generation platform, benchmark, and learning framework for controller-executed UAV policies. SUDA-SIM combines OpenUSD/RTX sensing, PhysX dynamics, PX4 SITL execution, lockstep logging, and parallel rollouts under one simulation clock. Using SUDA-SIM, we construct SUDA-UAV with 13,678 controller-executed trajectories and 1.6 million synchronized observations, organized as Dynamics-aware Action Trajectories with 15D motion-response targets. We further build a 200-task PX4 benchmark spanning seven task groups and eight VLA, world-model, and video-action policy families under a common closed-loop protocol. We further use a QwenVLA-style reference policy to study text-to-action pretraining, multimodal Action Chunk fine-tuning, and PPO post-training within the same framework. Evaluations show an advantage for controller-executed supervision: all six chunked families improve both SR and nDTW, and nDTW improves across all eight families. These results provide consistent evidence that controller-executed supervision improves closed-loop UAV policy learning under a shared PX4 SITL evaluation protocol.

1 INTRODUCTION

Language-guided UAV control translates human instructions and onboard observations into autonomous flight actions. Early studies mainly focused on mission planning, route generation, and target navigation, while recent work has moved toward continuous trajectory generation and direct action prediction Sautenkov et al. (2025b;a); Wang et al. (2025b;a). Unlike high-level planning, continuous control is closed-loop: each predicted action passes through the flight controller and vehicle dynamics before the next flight state and observation are formed. Training such policies therefore requires data that capture intended motion and UAV execution of control commands.

However, existing UAV navigation and policy datasets commonly supervise models with waypoint or 6-DoF reference trajectories aligned with egocentric observations and instructions Liu et al. (2023); Lee et al. (2025); Wang et al. (2025b;a). These references specify intended motion, but the trajectory experienced by a flying UAV is produced after commands pass through the flight-control system. As the left comparison of Figure 1 shows, controller execution records the path, body-state evolution, velocity response, and onboard views actually reached after tracking, giving supervision closer to real closed-loop UAV flight than geometric targets alone. Without controller execution, waypoint-level references do not align controller-facing commands with executed states and subsequent observations. Policies trained on such data may predict geometric reference actions without direct supervision for how commands become executed flight states and shape subsequent onboard observations during closed-loop flight. A key bottleneck is the lack of a data-collection platform that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

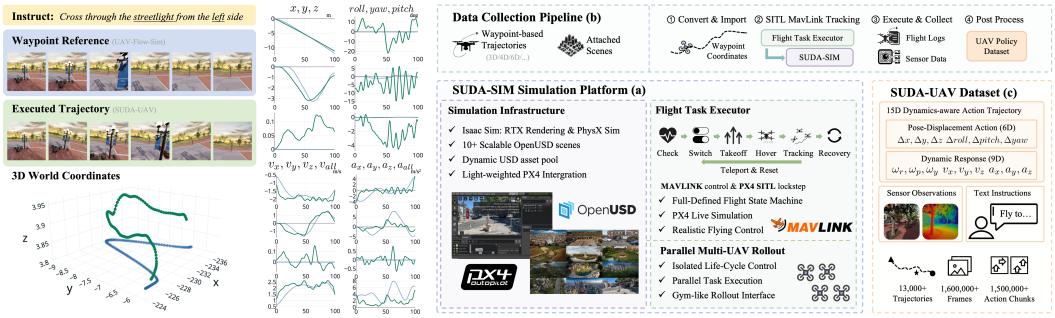


Figure 1: Overview of SUDA for controller-executed UAV policy learning. Left: a waypoint reference, its executed trajectory, and 3D world coordinates; controller execution captures attitude and viewpoint changes that more closely reflect real flight behavior. (a) SUDA-SIM platform. (b) Data collection pipeline. (c) SUDA-UAV with 15D Dynamics-aware Action Trajectories.

synchronizes control commands, executed states, and onboard observations during flight-controller execution, while supporting realistic dynamics and parallel rollouts. Among the representative platforms summarized in Table 1, we find no system that jointly provides flight-controller execution and lockstep-synchronized command–response–observation logging for policy training Shah et al. (2018); Xiao et al. (2020); Song et al. (2021); Xu et al. (2024); Wang et al. (2025b). This limits the construction of controller-executed training trajectories aligned with the observations produced during flight.

To address this limitation, we introduce SUDA-SIM, an integrated platform for flight-controller-mediated UAV simulation, synchronized data generation, policy rollout, and closed-loop evaluation. Figure 1 summarizes the platform: SUDA-SIM integrates OpenUSD scenes, RTX sensing, PhysX flight dynamics, MAVLink communication, and PX4 SITL execution under a shared clock. Its Flight Task Executor manages the complete execution process, while physics simulation, sensing, flight control, and data logging advance in lockstep. To support scalable policy training, SUDA-SIM further provides independent multi-UAV execution, parallel rollouts, and a Gym-compatible interface NVIDIA et al. (2025b); Meier et al. (2015); MAVLink Development Team (2026); Towers et al. (2025). The same figure also summarizes the data-collection pipeline. Waypoint trajectories and simulation scenes are first converted into executable flight tasks. PX4 SITL then tracks the imported waypoints in SUDA-SIM, while the platform records controller-facing commands, executed flight states, onboard observations, and flight logs at 20 Hz. These synchronized records are processed into training data for UAV vision-language-action policies and world models.

Building on SUDA-SIM, we construct SUDA-UAV, a controller-executed dataset for dynamics-aware UAV policy learning. As shown in Figure 1, SUDA-UAV contains 13,678 trajectories, 1,601,000 synchronized observations, and 1,505,254 indexed motion-response windows paired with language instructions. Each 15D motion-response vector combines a 6D pose response (3D egocentric displacement and 3D world-frame relative rotation) with 9D linear-velocity, angular-velocity, and linear-acceleration responses. To capture controller dynamics and action-conditioned observations beyond open-loop metrics, we establish a 200-task PX4 SITL benchmark across seven task groups. Each command is executed through PX4 before the policy receives its next observation, enabling controlled closed-loop evaluation of controller dynamics and action-conditioned observations across eight policy families.

Finally, we evaluate controller-executed supervision under a common PX4 closed-loop benchmark and provide a three-stage QwenVLA-style reference policy for dynamics-aware UAV control. The policy predicts Action Chunks from onboard images, language instructions, and flight states; text-to-action pretraining learns a motion prior, multimodal fine-tuning incorporates controller-executed responses, and PPO supports closed-loop post-training Wang et al. (2026); Schulman et al. (2017). Across eight policy families, SUDA-UAV adaptation improves mean SR from 41.4% to 51.3%, nDTW from 0.533 to 0.570, and OR@1m from 0.626 to 0.677, providing consistent evidence for controller-executed supervision under matched PX4 execution.

Our contributions are threefold. 1) We present SUDA-SIM, an integrated Isaac Sim–PX4 platform for lockstep-synchronized UAV simulation, data collection, and parallel policy rollouts. 2) We construct SUDA-UAV, an execution-aligned dataset that captures controller-facing commands, realized

Table 1: Comparison of UAV simulation platforms in infrastructure, assets, rendering, Flight Controller with Software in the Loop Simulation (FC SITL), RL API, and C–R–O Logging. C–R–O Logging denotes deterministic step-by-step logging of controller-facing commands, executed responses, and resulting observations for policy supervision; it is distinct from FC–sim clock lockstep (Section A.1).

Platform	Infrastructure	Assets	Render	FC SITL	RL API	C–R–O Logging
AirSim Shah et al. (2018)	UE/AirSim	medium	photo	✓	Gym	×
XTDrone Xiao et al. (2020)	Gazebo/PX4	medium	Gazebo	✓	×	×
Flightmare Song et al. (2021)	Unity/custom	rich	photo	×	Gym	×
Gym-PyBullet-Drones Panerati et al. (2021)	PyBullet	basic	basic	×	Gym	×
OmniDrones Xu et al. (2024)	Isaac Sim	medium	RTX/photo	×	TorchRL	×
Pegasus Jacinto et al. (2024)	Isaac Sim	medium	RTX/photo	✓	×	×
OpenUAV Wang et al. (2025b)	UE/AirSim	rich	photo	✓	×	×
AirSimAG Cui et al. (2026)	UE/AirSim	medium	photo	×	×	×
CARLA-Air Zeng et al. (2026)	UE/CARLA+AirSim	rich	photo	×	Gym	×
SUDA-SIM (Ours)	**Isaac Sim**	**rich**	**RTX/photo**	✓	**Gym**	✓

flight responses, and onboard observations. 3) We establish a common PX4 closed-loop benchmark and systematically evaluate controller-executed supervision across eight VLA, world-model, and video-action policy families, with a QwenVLA-style reference policy for multimodal training and closed-loop post-training.

2 Related Work

2.1 UAV Simulation Platforms

UAV simulators cover different layers of the aerial learning stack. Gazebo supports multi-robot simulation Koenig & Howard (2004); AirSim provides visual and physical vehicle simulation; XTDrone builds Gazebo–PX4 workflows; and PX4 SITL with MAVLink Offboard exposes the controller-facing setpoint contract used by PX4-equipped vehicles Shah et al. (2018); Xiao et al. (2020); Meier et al. (2015). Flightmare, Gym-PyBullet-Drones, OmniDrones, Pegasus, and AeroVerse provide complementary support for rendering, dynamics, Gym-style control, batched RL, multi-vehicle flight, or aerial-agent evaluation Song et al. (2021); Panerati et al. (2021); Xu et al. (2024); Jacinto et al. (2024); Yao et al. (2026). Recent platforms also expand realistic UAV navigation, air–ground simulation, heterogeneous collaboration, and synchronized multimodal collection Wang et al. (2025b); Zeng et al. (2026); Cui et al. (2026). SUDA-SIM instead brings rich assets, RTX rendering, PX4 execution, Gym-compatible learning, and lockstep logging into one platform.

2.2 UAV Dynamics and Sim-to-Real Transfer

Learning-based quadrotor control has advanced trajectory tracking under nonlinear dynamics and disturbances. DATT combines feedforward-feedback-adaptive control with $\mathcal{L}_1$ adaptive augmentation for precise tracking of arbitrary trajectories in real-world wind fields Huang et al. (2023). Quach et al. integrate Gaussian splatting with quadrotor dynamics and train Liquid networks via imitation learning, demonstrating zero-shot transfer from a single simulated indoor scene to outdoor real-world flight Quach et al. (2025). These works address tracking accuracy and sim-to-real transfer, respectively. SUDA-SIM occupies a complementary position: it provides a synchronized platform for generating controller-executed supervision across diverse policy families, with PX4 SITL lockstep execution ensuring that recorded responses reflect actual controller behavior.

2.3 Language-Guided UAV Tasks and Policies

Language-guided UAV tasks extend vision-language navigation from discrete instruction following to continuous embodied spaces Anderson et al. (2018); Krantz et al. (2020). Aerial studies span mission and route generation, language-conditioned reasoning, and realistic UAV navigation in UAV-VLA, UAV-VLPA, CognitiveDrone, and OpenUAV Sautenkov et al. (2025b;a); Lykov et al. (2025); Wang et al. (2025b). Recent policies cover navigation through AeroDuo, AutoFly, and AerialVLA Wu et al. (2025); Sun et al. (2026b); Xu et al. (2026), as well as racing, tracking, and aerial manipulation Serpiva et al. (2025); Zhang et al. (2026); Sun et al. (2026a). UAV-Flow pairs language with intended 6-DoF trajectories and adapts OpenVLA for continuous aerial imitation Wang

et al. (2025a); Kim et al. (2025b). ACT introduced Action Chunking for temporally coherent robot actions Zhao et al. (2023), and streaming flow policies update structured action sequences as observations arrive Jiang et al. (2025); Shi et al. (2026). General VLA models span autoregressive, temporal-generative, and structured action prediction Kim et al. (2025b); Black et al. (2025b;a); Kim et al. (2025a); Yang et al. (2026); Wu et al. (2026); Wang et al. (2026); FastWAM and VAMPO provide world-model and video-action formulations Yuan et al. (2026); Ge et al. (2026).

3 SUDA-SIM: Integrated UAV Simulation and Data Generation

3.1 Overview

Controller-executed supervision pairs each policy input with the motion produced by flight control and vehicle dynamics. A waypoint or 6-DoF reference specifies intended motion, while state estimation, cascaded flight control, and vehicle dynamics determine the realized position, attitude, velocity, angular-rate, and acceleration response Lee et al. (2010); Meier et al. (2015). That response also determines the next onboard view and flight state. A Dynamics-aware Action Trajectory aligns these quantities in each closed-loop transition, preserving commanded intent, tracking lag, transient vehicle response, and the resulting viewpoint change.

SUDA-SIM places UAV simulation, synchronized data generation, policy rollout, and closed-loop evaluation on the same lockstep timeline. OpenUSD/RTX sensing, PhysX multirotor dynamics, MAVLink-compatible PX4 SITL, and logging advance under one simulation clock. Task-reference and policy adapters follow the same PX4 setpoint convention, and each completed interval records the setpoint, executed flight state, and resulting observation together.

3.2 Platform Construction

SUDA-SIM integrates three jointly timed components: Isaac Sim simulation and sensing, a hierarchical MAVLink-connected Flight Task Executor, and parallel policy rollout. The shared clock coordinates their timing, state and feedback exchange, episode management, and data recording.

The simulation layer couples PhysX multirotor dynamics with RTX sensor rendering and OpenUSD scene composition in Isaac Sim. SUDA-SIM includes 19 scene configurations converted to USD from Unreal Engine assets associated with OpenUAV Wang et al. (2025b) and CARLA-Air Zeng et al. (2026), as well as from open-source three-dimensional Gaussian splatting (3DGS) scenes. Scenes can be exchanged without changing flight control or data recording, and dynamic assets share the same OpenUSD scene graph. Under Isaac Sim's default settings, the most complex configuration renders at up to 110 frames per second at 1280×720 on one RTX 4090. Rendering sensor frames from live PhysX–PX4 poses preserves the translation, attitude, and viewpoint changes of execution.

The Flight Task Executor turns each task reference into a complete PX4 episode. A hierarchical state machine coordinates flight-controller initialization, mode transitions, autonomous takeoff, hover stabilization, setpoint streaming, and termination. Each episode starts from an aligned pose and yaw, and policy rollout begins after the vehicle reaches a stable hover. This initialization gives the estimator and cascaded controller a consistent state, so the recorded response reflects both the task reference and the ensuing closed-loop dynamics.

A shared clock advances Isaac Sim and PX4 SITL in lockstep, so each 20 Hz record aligns commands, observations, and PX4 state/feedback.

SUDA-SIM supports parallel PX4-controlled vehicles with isolated dynamics, visibility, collision, and control. In the reported experiments, four vehicles share each loaded scene and advance under the same clock. A Gym-compatible batch interface provides synchronized 20 Hz rollouts for policy learning. Reusing a scene reduces the cost of complex visual assets, while instance isolation prevents one task from altering another task's observations or dynamics. The supplementary material reports collection throughput, resource use, reset conditions, and lockstep timing.

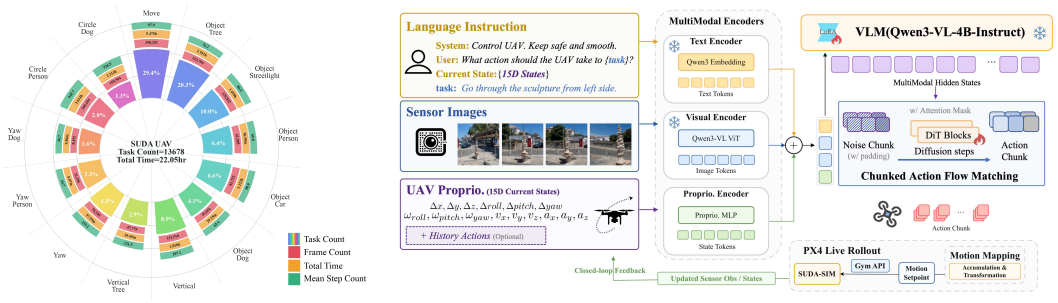
The task-reference and policy adapters follow the same PX4 setpoint convention, initialization sequence, and control timing; each observation is rendered from the pose reached after the assigned physics steps and PX4 update, so training targets and benchmark responses are generated under the same command–response–observation process. Each transition advances the prescribed PhysX and

PX4 updates and is indexed by simulated control time: variations in rendering or policy-inference latency affect wall-clock throughput without changing the update sequence or command–response alignment, and parallel instances follow the same episode definition.

Data recording captures five synchronized streams at 20 Hz: RGB observations, PX4 controller setpoints, PX4 state and feedback estimates, executed motion responses, and 3D world positions, all timestamped by the shared simulation clock. The setpoint covers position, velocity, yaw, and yaw-rate targets; the executed response captures displacement, rotation, velocity, angular velocity, and acceleration after controller tracking. Post-collection processing applies temporal alignment, valid-window cropping, and quality filtering before indexing.

4 SUDA-UAV DATASET

SUDA-UAV is constructed from synchronized controller-executed rollouts in SUDA-SIM. Each trajectory preserves onboard observations, PX4 controller setpoints, PX4 state and feedback, executed responses, and 3D positions over the complete valid task interval. These records support 15D motion-response learning and path-based closed-loop evaluation while retaining task references, controller commands, and realized motion as distinct variables.



(a) **(b)**

Figure 2: SUDA-UAV statistics and QwenVLA-style policy overview. (a) Task distribution with counts, duration, and mean steps. (b) Policy architecture and closed-loop deployment: RGB, instruction, and motion-response inputs condition Action Chunk prediction, which is mapped to PX4 setpoints for SUDA-SIM rollout.

Reference source and coordinate conversion. SUDA-UAV uses the native 6-DoF task references associated with UAV-Flow Wang et al. (2025a), referred to here as UAV-Flow-Sim. Each reference pairs a language instruction with the intended evolution of displacement and orientation. Because the source follows an AirSim-over-Unreal coordinate convention, SUDA-SIM maps each pose to the Isaac/PX4 execution convention before rollout. The converted reference retains the original task semantics and trajectory structure and supplies the task specification from which the task-reference adapter constructs controller-facing setpoints. The synchronized flight log supplies the corresponding 15D motion-response vectors. For the supervision comparison, the reference condition uses the same 15D motion representation, with velocity, angular velocity, and acceleration deterministically derived from the native reference sequence (Section C.3). In this paper, *task reference* denotes the input specification, *reference trajectory* its coordinate-converted path after the predefined corrections in the supplement, *controller setpoint* the command passed to PX4, and *executed response* the recorded motion after flight control and vehicle dynamics.

PX4-executed data generation. Task-reference and policy adapters provide PX4 Offboard-compatible MAVLink setpoint streams PX4 Development Team (2026); MAVLink Development Team (2026), which the Flight Task Executor dispatches during each episode. SUDA-SIM synchronizes the resulting observations, setpoints, PX4 state/feedback, and executed responses; each valid interval becomes one transition in a Dynamics-aware Action Trajectory, and each valid rollout yields one complete trajectory. Policy-generated setpoints and additional task-reference sources follow the same timing and recording procedure. Reset, lockstep timing, coordinate conversion, setpoint mapping, and rollout termination are detailed in the supplementary material.

Dynamics-aware Action Trajectory. For each log with instruction q, we record

$$\mathcal{T}^{\mathrm{DAT}} = \{(I_t, q, \boldsymbol{u}_t^{\mathrm{PX4}}, \boldsymbol{o}_{t+1}^{\mathrm{act}}, I_{t+1})\}_{t=1}^{T-1}, \tag{1}$$

which aligns the onboard observation and controller-facing setpoint at time t with the executed motion-response vector and onboard observation at time $t+1$. Its response sequence is $\mathcal{O}^{\mathrm{act}} = \{\boldsymbol{o}_t^{\mathrm{act}}\}_{t=1}^{T}$, where $\boldsymbol{o}_t^{\mathrm{act}} \in \mathbb{R}^{15}$:

$$\begin{aligned} \boldsymbol{o}_t^{\mathrm{act}} &= [\boldsymbol{d}_t^\top, \boldsymbol{\eta}_t^\top, \boldsymbol{v}_t^\top, \boldsymbol{\omega}_t^\top, \boldsymbol{a}_t^\top]^\top, \\ \boldsymbol{d}_t &= [\Delta x_t, \Delta y_t, \Delta z_t]^\top, \quad \boldsymbol{\eta}_t = [\Delta\phi_t, \Delta\theta_t, \Delta\psi_t]^\top, \\ \boldsymbol{v}_t &= [v_t^x, v_t^y, v_t^z]^\top, \quad \boldsymbol{\omega}_t = [\omega_t^x, \omega_t^y, \omega_t^z]^\top, \\ \boldsymbol{a}_t &= [a_t^x, a_t^y, a_t^z]^\top. \end{aligned} \tag{2}$$

In Eq. (2), $\boldsymbol{d}_t$ is start-of-interval FLU body displacement (m), $\boldsymbol{\eta}_t$ relative Euler rotation (rad), $\boldsymbol{v}_t$ current-frame FLU body velocity (m/s), $\boldsymbol{\omega}_t$ FLU body-frame angular velocity (rad/s), and $\boldsymbol{a}_t$ FLU body-frame linear acceleration (m/s^2). The first two terms encode the transition from $t-1$ to t; the remaining channels are sampled at t, with zero initial displacement and rotation. Thus the tuple in Eq. (1) contains $\boldsymbol{o}_{t+1}^{\mathrm{act}}$, whose pose terms encode the transition from t to $t+1$. All 15 channels are executed-response targets. The split in Eq. (3) retains seven channels for deployment and eight for auxiliary response supervision:

$$\begin{aligned} \boldsymbol{c}_t^{\mathrm{exec}} &= [\boldsymbol{d}_t^\top, \boldsymbol{v}_t^\top, \omega_t^z]^\top \in \mathbb{R}^7, \\ \boldsymbol{r}_t^{\mathrm{resp}} &= [\boldsymbol{\eta}_t^\top, \omega_t^x, \omega_t^y, \boldsymbol{a}_t^\top]^\top \in \mathbb{R}^8. \end{aligned} \tag{3}$$

The recorded PX4 setpoint remains the separate variable $\boldsymbol{u}_t^{\mathrm{PX4}}$ in Eq. (1). Given a predicted motion-response vector, the fixed execution adapter constructs

$$\boldsymbol{u}_{t,h}^{\mathrm{PX4}} = \Pi_{\mathrm{exec}}\left(\widehat{\boldsymbol{c}}_{t,h}^{\mathrm{exec}}; \boldsymbol{u}_{t,h-1}^{\mathrm{PX4}}, \boldsymbol{x}_{t,h}\right). \tag{4}$$

In Eq. (4), $\widehat{\boldsymbol{c}}_{t,h}^{\mathrm{exec}}$ is the policy-predicted 7D execution-facing projection at substep h, $\boldsymbol{u}_{t,h-1}^{\mathrm{PX4}}$ is the preceding accumulated PX4 controller setpoint, and $\boldsymbol{x}_{t,h}$ is the current execution state. Position and yaw targets accumulate; predicted velocity and yaw-rate are applied directly at each step. The policy learns all 15 channels from executed responses. At deployment, Π_{exec} consumes the 7D projection $\boldsymbol{c}_t^{\mathrm{exec}}$; the eight channels in $\boldsymbol{r}_t^{\mathrm{resp}}$ remain executed-response prediction targets, excluded from the PX4 setpoint. The supplement details channel layout, coordinate conversion, and setpoint mapping.

A Dynamics-aware Action Trajectory is the complete observation–setpoint–executed-response–next-observation sequence in Eq. (1). An Action Chunk is a contiguous short-horizon block of future 15D motion-response vectors learned jointly as one supervision target. Of the 15 channels, seven (displacement, velocity, yaw rate) map directly to PX4 setpoints as execution-facing commands; the remaining eight (acceleration, angular velocity, next-observation pose) serve as response-only supervision targets that regularize the shared decoder toward dynamics-aware representations without directly actuating the vehicle. At policy step t, $(I_t, q, \boldsymbol{s}_t)$ conditions $\{\boldsymbol{o}_{t+h}^{\mathrm{act}}\}_{h=1}^{H}$ (chunk length $H{=}16$, with an eight-step execution stride), while $\boldsymbol{o}_t^{\mathrm{act}}$ is the executed response available at t. Chunked policies execute several projected vectors before receiving an updated observation; single-step policies execute one projected vector per policy step.

Dataset statistics. We log synchronized controller-executed transitions at 20 Hz and organize each valid rollout as a Dynamics-aware Action Trajectory. Figure 2(a) summarizes the task distribution. After temporal alignment and valid-window cropping, SUDA-UAV contains 13,678 controller-executed trajectories and 1,601,000 synchronized observation frames. The 13,478 training trajectories span 22.05 hours according to per-trajectory metadata, and the remaining 200 trajectories form the evaluation benchmark. Across the full collection, the frame count corresponds to 22.24 frame-equivalent hours at 20 Hz. Following the temporally grouped action formulations used by Qwen-VLA and GR00T N1 Wang et al. (2026); NVIDIA et al. (2025a), the synchronized records provide 1,505,254 indexed motion-response windows for temporally structured prediction. The trajectories span object approach, object side/target relation, object pass-through/away, synthetic translation, vertical motion, yaw-to-target, and circle tasks. Task-relevant assets include vehicles, pedestrians/animals, and static scene objects. Additional split, scale, path-length, and asset statistics appear in the supplement.

Evaluation. Cross-policy evaluation uses a benchmark of 200 tasks sampled uniformly from the full SUDA-UAV collection and spanning seven groups: object approach, object side/target relation, object pass-through/away, synthetic translation, vertical motion, yaw-to-target, and circle. The

benchmark contains 138 object- or scene-anchored tasks and 62 target-free tasks. Every policy follows the same PX4 SITL protocol. A model-specific projection maps each output to the SUDA motion representation, and the execution adapter produces PX4 Offboard setpoints. Task-aware Success Rate (SR) measures completion under the predicate defined for each task family. Point-count-normalized Dynamic Time Warping (nDTW), Absolute Trajectory Error (ATE), and Final Displacement Error (FDE) compare each controller-executed 3D path with its reference trajectory. Success-weighted DTW (sDTW) combines completion and path alignment, while Oracle Rate at 1 m (OR@1m) records whether the rollout approaches within 1 m of the reference endpoint. The supplement details benchmark construction, category statistics, metrics, and success predicates.

5 Policy Learning Methodology

Our closed-loop chunked policy uses onboard RGB I_t, instruction q, and the current 15D motion-response vector $\boldsymbol{s}_t := \boldsymbol{o}_t^{\mathrm{act}}$ to predict the sequence from $t+1$. Following Qwen-VLA Wang et al. (2026), the *QwenVLA-style policy* couples Qwen3-VL-4B Bai et al. (2025) with a flow-based Action Chunk decoder Lipman et al. (2023) for temporally structured controller-executed motion, as shown in Figure 2(b). We present the architecture and flow-matching objective, text-to-action pretraining and multimodal fine-tuning, and the PPO and DepthAux analyses.

With LoRA Hu et al. (2022), the backbone encodes $Z_t = F_\phi(I_t, \rho(q), g_\gamma(\boldsymbol{s}_t))$, where ρ forms the embodiment prompt and g_γ maps the current motion-response vector to proprioceptive tokens. The decoder predicts a padded action tensor $\boldsymbol{Y}_t$, and a binary mask $\boldsymbol{M}_t$ selects the 15D Action Chunk channels. Dataset-wide 1st/99th percentiles scale each motion channel to $[-1, 1]$ for flow matching.

For flow matching, let $\boldsymbol{Y}_0$ denote a normalized Action Chunk, $\boldsymbol{Y}_1 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\tau \sim \mathcal{U}(0,1)$. We use the masked flow-matching loss in Eq. (5):

$$\begin{aligned} \boldsymbol{Y}_\tau &= (1-\tau)\boldsymbol{Y}_0 + \tau \boldsymbol{Y}_1, \qquad \boldsymbol{u}^\star = \boldsymbol{Y}_1 - \boldsymbol{Y}_0, \\ \boldsymbol{u}_\theta &= f_\theta(\boldsymbol{Y}_\tau, \tau \mid Z_t), \\ \mathcal{L}_{\mathrm{FM}} &= \frac{\|\boldsymbol{M}_t \odot (\boldsymbol{u}_\theta - \boldsymbol{u}^\star)\|_F^2}{\max(\|\boldsymbol{M}_t\|_1, 1)}. \end{aligned} \tag{5}$$

At inference, seeded reverse integration from $\tau = 1$ to $\tau = 0$ transforms Gaussian noise into a normalized Action Chunk and inverts the dataset scaling. For each executed vector, the adapter accumulates predicted displacement into position, applies predicted velocity as feed-forward, and uses the body-z angular rate as the yaw-rate command while integrating it into the yaw target. The PX4 setpoint comprises position, velocity, yaw, and yaw-rate; acceleration remains in the supervised 15D motion-response vector. This design bypasses inverting PX4 controller dynamics: the policy predicts the *desired outcome* of execution (displacement and velocity the controller should realize), and the adapter translates this outcome into setpoint increments that PX4's PID loop tracks. Solver, coordinate, and timing details appear in the supplement.

We train the QwenVLA-style policy through three successive stages Wang et al. (2026). T2A learns a language-conditioned UAV motion prior from instruction text and the current motion-response vector without image input; multimodal SFT grounds this prior in onboard RGB for controller-executed Action Chunk prediction; and PPO post-training optimizes motion, path, and completion rewards after PX4 execution in SUDA-SIM. The DepthAux SFT variant adds quantized depth-feature supervision during training while retaining the same RGB-, instruction-, and motion-response-conditioned inference interface. Training details appear in the supplement.

PPO post-training. The PPO stage starts from SFT (LoRA) w/ SUDA-UAV, retains the 15D Action Chunk representation and the PX4 execution procedure, and optimizes the policy with the clipped policy-ratio objective of PPO Schulman et al. (2017) on closed-loop SUDA-SIM feedback. Seeded flow integration produces a normalized Action Chunk, from which a fixed-variance Gaussian policy samples the first eight steps for each rollout stride. The execution adapter maps seven channels per step to PX4 setpoints; the remaining eight response-only channels bypass PX4, and PPO evaluates the joint log-density of the unclipped sample over all 15 channels. This joint objective is intentional: response-only channels share the decoder with execution-facing channels, and their likelihood terms regularize the shared representation toward physically consistent predictions, even though only seven channels have direct causal effect on the environment. PX4-executed states and

paths determine the motion, path, and task terms; the execution-facing prediction determines regularization. For rollout stride i, define

$$R_i = R_i^{\text{motion}} + R_i^{\text{path}} + R_i^{\text{task}} - R_i^{\text{reg}},$$
$$\hat{S}_i = \frac{R_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \varepsilon_{\text{std}}}. \tag{6}$$

Eq. (6) combines motion, path, and task-completion terms from the PX4-executed trajectory, while R_i^{reg} penalizes the predicted execution-facing commands. Batch statistics standardize the stride return into $\hat{S}_i$, with $\varepsilon_{\text{std}} > 0$ as a numerical stabilizer. Let $\ell_{\theta,i}^{\text{joint}}$ and $\ell_{\theta_{\text{old}},i}^{\text{joint}}$ denote the Gaussian surrogate log-densities over all 15 channels of the eight-step sample. The ratio is $r_i(\theta) = \exp(\ell_{\theta,i}^{\text{joint}} - \ell_{\theta_{\text{old}},i}^{\text{joint}})$, and $\bar{r}_i(\theta) = \text{clip}(r_i(\theta), 1 - \epsilon_{\text{PPO}}, 1 + \epsilon_{\text{PPO}})$. We maximize the objective in Eq. (7):

$$\mathcal{J}_{\text{PPO}} = \mathbb{E}_i \left[\min \left(r_i(\theta)\hat{S}_i, \bar{r}_i(\theta)\hat{S}_i \right) \right]. \tag{7}$$

DepthAux supervision. DepthAux uses simulator-rendered depth as training-time auxiliary supervision. A frozen depth vector-quantized variational autoencoder (VQ-VAE), pretrained on SUDA depth data, maps D_t to a quantized feature target $\boldsymbol{Z}_t^D = \text{sg}(E_D(D_t)) \in \mathbb{R}^{64\times256}$. A DepthAux head predicts $\widehat{\boldsymbol{Z}}_t^D = G_\zeta(Z_t)$ from pooled VLM features. The objective in Eq. (8) is

$$\mathcal{L}_{\text{DepthAux}} = \mathcal{L}_{\text{FM}} + \frac{\lambda_D}{64 \cdot 256} \left\| \widehat{\boldsymbol{Z}}_t^D - \boldsymbol{Z}_t^D \right\|_F^2, \tag{8}$$

where $\lambda_D = 0.05$. DepthAux adds quantized geometric supervision during training and retains the SFT (LoRA) RGB–instruction–motion-response Action Chunk interface at deployment.

6 Experiments

6.1 Experimental Setting

Dataset and benchmark. We uniformly sample 200 complete trajectory IDs from the SUDA-UAV index as the fixed benchmark; the other 13,478 trajectories provide training data. This follows the in-distribution protocol of UAV-Flow-Sim Wang et al. (2025a), drawing test trajectories from the same simulation environment for controlled comparison under identical visual and dynamics conditions. Each checkpoint is trained and evaluated once. *w/ SUDA-UAV* denotes Controller-Executed Supervision from synchronized PX4-executed records; *w/o SUDA-UAV* denotes Reference Supervision in the same 15D representation, with kinematic channels deterministically derived from native 6-DoF references (Section C.3). Each pair shares its architecture family, parameter count, training steps, optimizer, observation stream, and PX4 execution protocol, with the supervision source as the experimental variable. The eight systems preserve each family's action-generation mechanism under a common interface. All methods share the seven-group PX4 SITL benchmark with reset, takeoff, hover, lockstep timing, and task budgets; task IDs, initial states, references, PX4 execution, action budgets, and metrics are held fixed. We report SR, nDTW, sDTW, ATE, FDE, and OR@1m; details appear in the supplement.

Models and parameters. Eight VLA, world-model, or video-action families retain their original single-step or chunked decoders, with outputs projected to the PX4 adapter. The QwenVLA-style policy uses Qwen3-VL-4B with 33.03M LoRA parameters (0.83% of the backbone) and a 1.16B chunked action flow matching decoder; the supplement reports decoder and optimizer settings.

6.2 Main Results

Under the common PX4 protocol, SUDA-UAV adaptation raises mean SR from 41.4% to 51.3%, nDTW from 0.533 to 0.570, and OR@1m from 0.626 to 0.677 across the eight policy families. Paired task-level bootstrap gives a family-mean gain of 9.9 points in SR (95% CI [7.2, 12.6]) and 0.037 in nDTW ([0.030, 0.044]); complete intervals appear in the supplement. ABot-M0 records the largest SR increase at 19.0 points, and VAMPO gains 0.103 nDTW and 0.140 OR@1m. All six chunked families improve both SR and nDTW, while nDTW improves across all eight comparisons (Figure 3).

Table 2: Closed-loop results with and without SUDA-UAV adaptation. Bold marks w/ values that match or improve over the paired baseline.

Policy	SR (%) ↑		nDTW ↑		OR@1m ↑	
	w/o	w/	w/o	w/	w/o	w/
World-model / video-action, chunk						
VAMPO	49.0	**63.0**	0.546	**0.649**	0.635	**0.775**
FastWAM	37.0	**41.0**	0.523	**0.532**	0.640	0.605
VLA, chunk						
ABot-M0	44.0	**63.0**	0.544	**0.644**	0.635	**0.780**
OpenVLA-OFT	42.0	**51.5**	0.536	**0.561**	0.635	**0.680**
OpenPI-0.5	34.5	**49.0**	0.527	**0.563**	0.595	**0.700**
LingBot-VLA	32.0	**38.5**	0.506	**0.516**	0.585	0.575
VLA, single-step						
OpenVLA-7B	47.0	**52.5**	0.541	**0.547**	0.645	**0.645**
UAV-Flow-OpenVLA	45.5	**51.5**	0.538	**0.546**	0.635	**0.655**
Mean	41.4	**51.3**	0.533	**0.570**	0.626	**0.677**

Table 3: QwenVLA-style supervision and method comparisons; both use the same 15D representation.

Family	Condition	SR ↑	nDTW ↑	sDTW ↑	OR@1m ↑
Supervision Source					
SFT (LoRA)	w/ SUDA	**64.5**	**0.628**	**0.506**	0.680
SFT (LoRA)	w/o SUDA	41.5	0.336	0.210	**0.780**
Method: Multimodal Supervision					
SFT (LoRA)		**64.5**	**0.628**	**0.506**	**0.680**
T2A	-Image	27.5	0.437	0.191	0.640
Method: PPO-based RL					
SFT (LoRA)		64.5	0.628	0.506	0.680
PPO-based RL	+RL	**66.5**	**0.630**	**0.520**	**0.695**
Method: DepthAux					
SFT (LoRA)		64.5	**0.628**	**0.506**	**0.680**
DepthAux	+Depth	**65.0**	0.617	0.488	0.650

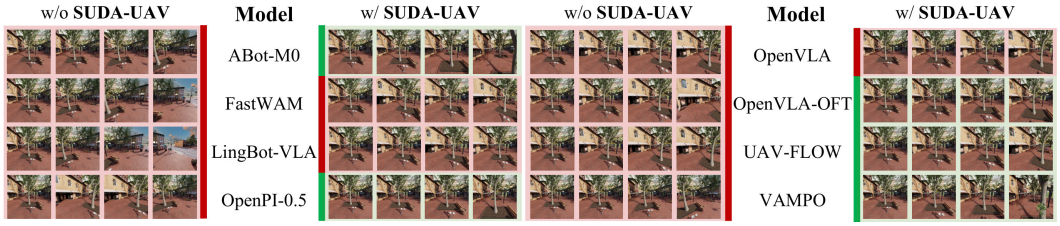


Figure 3: Qualitative result for "fly to the tree ahead": w/o and w/ SUDA-UAV supervision; green: success, red: failure.

6.3 Discussion

Table 3 summarizes four comparisons. Controller-Executed Supervision leads on SR, nDTW, and sDTW, and onboard RGB is decisive: multimodal SFT reaches 64.5% SR where the text-only T2A condition reaches 27.5%. PPO post-training raises SR to 66.5%, with a bootstrap interval overlapping zero (Appendix E). DepthAux slightly raises SR (65.0%) with lower path-level scores. OR@1m diverges: the reference-supervised baseline reaches 0.780 while completing far fewer tasks (41.5% vs. 64.5%). Our evaluation follows the same in-distribution protocol as prior UAV benchmarks Wang et al. (2025a); bootstrap intervals quantifying task variance appear in Appendix E.

7 Conclusion

SUDA provides an integrated path for continuous UAV policy learning: SUDA-SIM scales controller-executed supervision, SUDA-UAV organizes it as Dynamics-aware Action Trajectories, and the PX4 benchmark evaluates policies under a shared closed-loop protocol. QwenVLA-style learning combines motion pretraining, multimodal grounding, rollout calibration, and geometric regularization, a reusable basis for UAV policy learning. Across eight policy families, the results provide consistent evidence that controller-executed supervision improves closed-loop learning under matched PX4 SITL evaluation.

Reproducibility Statement

All datasets, benchmark definitions, training and evaluation code, and model checkpoints are described in the Appendix and will be released for full reproduction.

Ethics Statement

This work involves simulation-based UAV experiments only; no human subjects or personal data are involved.

AI Use Statement

We used generative AI tools for code debugging, LaTeX formatting, and language editing; no AI tools were used for data or experiments.

REFERENCES

Peter Anderson, Qi Wu, Damien Teney, et al. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018. doi: 10.1109/CVPR.2018.00387.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems XXI*, volume 21, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Yangjie Cui, Xin Dong, Boyang Gao, Jinwu Xiang, Daochun Li, and Zhan Tu. AirSimAG: A high-fidelity simulation platform for air-ground collaborative robotics, 2026. URL https://arxiv.org/abs/2603.23079.

Zirui Ge, Pengxiang Ding, Baohua Yin, Qishen Wang, Zhiyong Xie, Yemin Wang, Jinbo Wang, Hengtao Li, Runze Suo, Wenxuan Song, Han Zhao, Shangke Lyu, Zhaoxin Fan, Haoang Li, Ran Cheng, Cheng Chi, Huibin Ge, Yaozhi Luo, and Donglin Wang. VAMPO: Policy optimization for improving visual dynamics in video action models, 2026. URL https://arxiv.org/abs/2603.19370.

Edward J. Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Kevin Huang, Rwik Rana, Alexander Spitzer, Guanya Shi, and Byron Boots. Datt: Deep adaptive trajectory tracking for quadrotor control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 326–340. PMLR, 2023.

Marcelo Jacinto, João Pinto, Jay Patrikar, John Keller, Rita Cunha, Sebastian Scherer, and António Pascoal. Pegasus simulator: An Isaac Sim framework for multiple aerial vehicles simulation. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 917–922, 2024. doi: 10.1109/ICUAS60882.2024.10556959.

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 238–257. PMLR, 2025. URL https://proceedings.mlr.press/v305/jiang25a.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems XXI*, volume 21, 2025a. doi: 10.15607/RSS.2025.XXI.017. URL https://www.roboticsproceedings.org/rss21/p017.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin

Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Nathan Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pp. 2149–2154, 2004. doi: 10.1109/IROS.2004.1389727.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision – ECCV 2020*, pp. 104–120, 2020. doi: 10.1007/978-3-030-58604-1_7.

Jungdae Lee, Taiki Miyanishi, Shuhei Kurita, Koya Sakamoto, Daichi Azuma, Yutaka Matsuo, and Nakamasa Inoue. CityNav: A large-scale dataset for real-world aerial navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5912–5922, 2025. doi: 10.1109/ICCV51701.2025.00559.

Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *Proceedings of the 49th IEEE Conference on Decision and Control*, pp. 5420–5425, 2010. doi: 10.1109/CDC.2010.5717652.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. AerialVLN: Vision-and-language navigation for UAVs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15338–15348, 2023. doi: 10.1109/ICCV51070.2023.01411.

Artem Lykov, Valerii Serpiva, Muhammad Haris Khan, Oleg Sautenkov, Artyom Myshlyaev, Grik Tadevosyan, Yasheerah Yaqoot, and Dzmitry Tsetserukou. Cognitivedrone: A vla model and evaluation benchmark for real-time cognitive task solving and reasoning in uavs, 2025. URL https://arxiv.org/abs/2503.01378.

MAVLink Development Team. Offboard control interface. https://mavlink.io/en/services/offboard_control.html, 2026. Accessed: 2026-07-24.

Lorenz Meier, Dominik Honegger, and Marc Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *International Conference on Robotics and Automation (ICRA)*, pp. 6235–6240, 2015. doi: 10.1109/ICRA.2015.7140074.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots, 2025a. URL https://arxiv.org/abs/2503.14734.

NVIDIA, Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A GPU-accelerated simulation framework for multi-modal robot learning, 2025b. URL https://arxiv.org/abs/2511.04831.

Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P. Schoellig. Learning to fly—a gym environment with PyBullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7512–7519, 2021. doi: 10.1109/IROS51168.2021.9635857.

PX4 Development Team. Offboard mode (generic/all frames). https://docs.px4.io/main/en/flight_modes/offboard, 2026. Accessed: 2026-07-24.

Alex Quach, Makram Chahine, Alexander Amini, Ramin Hasani, and Daniela Rus. Gaussian splatting to real world flight navigation transfer with liquid networks. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1069–1093. PMLR, 2025.

Oleg Sautenkov, Aibek Akhmetkazy, Yasheerah Yaqoot, Muhammad Ahsan Mustafa, Grik Tadevosyan, Artem Lykov, Valerii Serpiva, and Dzmitry Tsetserukou. UAV-VLPA*: Vision-language guided global-local UAV mission planning from satellite imagery. In *2025 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2354–2359, 2025a. doi: 10.1109/ROBIO66223.2025.11377338.

Oleg Sautenkov, Yasheerah Yaqoot, Artem Lykov, Muhammad Ahsan Mustafa, Grik Tadevosyan, Aibek Akhmetkazy, Miguel Altamirano Cabrera, Mikhail Martynov, Sausar Karaf, and Dzmitry Tsetserukou. UAV-VLA: Vision-language-action system for large scale aerial mission generation. In *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 1588–1592, 2025b. doi: 10.1109/HRI61500.2025.10974117.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Valerii Serpiva, Artem Lykov, Artyom Myshlyaev, et al. Racevla: Vla-based racing drone navigation with human-like behaviour, 2025. URL https://arxiv.org/abs/2503.02572.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, volume 5 of *Springer Proceedings in Advanced Robotics*, pp. 621–635, 2018. doi: 10.1007/978-3-319-67361-5_40.

Yiran Shi, Dongqi Guo, Tianchen Zhao, et al. Streamingvla: Streaming vision-language-action model with action flow matching and adaptive early observation, 2026. URL https://arxiv.org/abs/2603.28565.

Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1147–1157, 2021. URL https://proceedings.mlr.press/v155/song21a.html.

Jianli Sun, Bin Tian, Qiyao Zhang, et al. Air-vla: Vision-language-action systems for aerial manipulation, 2026a. URL https://arxiv.org/abs/2601.21602.

Xiaolou Sun, Wufei Si, Wenhui Ni, et al. AutoFly: Vision-language-action model for UAV autonomous navigation in the wild. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://openreview.net/forum?id=88RKxlFUNY.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d7ff1795e8527f6443371c3933bdb52b-Abstract-Datasets_and_Benchmarks_Track.html.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, Xuhong Huang, Pei Lin, Junyang Lin, Dayiheng Liu, Shuai Bai, Jingren Zhou, Jiazhao Zhang, Haoqi Yuan, Gengze Zhou, Hang Yin, Ye Wang, Yiyang Huang, Zixing Lei, Wujian Peng, Delin Chen, Yingming Zheng, Jingyang Fan, Xianwei Zhuang, Xin Zhou, Haoyang Li, Anzhe Chen, Tong Zhang, Xuejing Liu, Yuchong Sun, Ruizhe Chen, Zhaohai Li, Chenxu Lü, Zhibo Yang, Tao Yu, and Xionghui Chen. Qwen-VLA: Unifying vision-language-action modeling across tasks, environments, and robot embodiments, 2026. URL https://arxiv.org/abs/2605.30280.

Xiangyu Wang, Donglin Yang, Yue Liao, Wenhao Zheng, Wenjun Wu, Bin Dai, Hongsheng Li, and Si Liu. UAV-Flow Colosseo: A real-world benchmark for flying-on-a-word UAV imitation learning. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/92cfa104db7fdd053492df3589ddfd49-Abstract-Datasets_and_Benchmarks_Track.html.

Xiangyu Wang, Donglin Yang, Ziqin Wang, Hohin Kwan, Jinyu Chen, Wenjun Wu, Hongsheng Li, Yue Liao, and Si Liu. Towards realistic UAV vision-language navigation: Platform, benchmark, and methodology. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=rUvCIvI4eB.

Ruipu Wu, Yige Zhang, Jinyu Chen, et al. Aeroduo: Aerial duo for uav-based vision and language navigation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 2576–2585, 2025. doi: 10.1145/3746027.3754498.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model, 2026. URL https://arxiv.org/abs/2601.18692.

Kun Xiao, Shaochang Tan, Guohui Wang, Xueyan An, Xiang Wang, and Xiangke Wang. XTDrone: A customizable multi-rotor UAVs simulation platform. In *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*, pp. 55–61, 2020. doi: 10.1109/ICRAS49812.2020.9134922.

Botian Xu, Feng Gao, Chao Yu, Ruize Zhang, Yi Wu, and Yu Wang. Omnidrones: An efficient and flexible platform for reinforcement learning in drone control. *IEEE Robotics and Automation Letters*, 9(3):2838–2844, 2024. doi: 10.1109/LRA.2024.3356168.

Peng Xu, Zhengnan Deng, Jiayan Deng, Zonghua Gu, and Shaohua Wan. Aerialvla: A vision-language-action model for uav navigation via minimalist end-to-end control, 2026. URL https://arxiv.org/abs/2603.14363.

Yandan Yang, Shuang Zeng, Tong Lin, et al. ABot-M0: Vla foundation model for robotic manipulation with action manifold learning, 2026. URL https://arxiv.org/abs/2602.11236.

Fanglong Yao, Yuanchang Yue, Youzhi Liu, et al. AeroVerse: Uav-agent benchmark suite for simulating, pre-training, finetuning, and evaluating aerospace embodied foundation models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2026. doi: 10.1109/TPAMI.2026.3697634.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

Tianle Zeng, Yanci Wen, and Hong Zhang. CARLA-Air: Fly drones inside a CARLA world—a unified infrastructure for air-ground embodied intelligence, 2026. URL https://arxiv.org/abs/2603.28032.

Qiyao Zhang, Shuhua Zheng, Jianli Sun, et al. UAV-Track VLA: Embodied aerial tracking via vision-language-action models, 2026. URL https://arxiv.org/abs/2604.02241.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016.

A Experimental and Reproducibility Details

A.1 Relation to Prior Simulation Platforms

Several prior platforms provide flight-controller-to-simulator (FC–sim) lockstep: AirSim supports PX4 lockstep simulation, XTDrone couples ROS, Gazebo, and PX4 in lockstep, and the Pegasus Isaac Sim backend enables PX4 lockstep by default. The C–R–O Logging column in Table 1 of the main paper refers to a different, joint capability: deterministic step-by-step logging of controller-facing commands, executed responses, and resulting observations for policy supervision. The SUDA-SIM entry marks the combination of FC–sim lockstep controller execution with this policy-oriented logging, together with dataset generation and a common closed-loop learning-and-evaluation interface. Prior platforms can synchronize the flight controller with the simulator clock; SUDA-SIM additionally synchronizes and records the command–response–observation triplet that policy learning consumes.

A.2 Simulation and Collection Throughput

SUDA-SIM contains 19 scenes converted to USD from OpenUAV/CARLA-Air Unreal Engine assets and open-source 3DGS reconstructions. With the default Isaac Sim settings at 1280×720 resolution, the most complex scene renders at up to 110 frames per second. The platform records four UAVs concurrently at 20 Hz, with synchronized RGB observations and optional depth.

An episode takes approximately 70 s of wall-clock time on average, including reset, PX4 restart, takeoff, hover stabilization, task execution, and logging. SUDA-UAV retains the valid task interval, while longer maneuvers such as object circling naturally extend collection time. The largest Unreal Engine-derived scenes occupy approximately 14.7 GB of GPU memory after loading; concurrent collection adds the cost of rendering each vehicle's sensors.

A.3 Computing Environment

Data generation, PX4 SITL rollout, rendering, and benchmark analysis were conducted on Ubuntu 22.04.5 LTS with an Intel Core i9-14900KF CPU, 64 GB of system memory, and one NVIDIA GeForce RTX 4090 GPU with 24 GB of memory. Policy development and evaluation used the software environment in Table A1. Comparison policies were adapted in the same Python/PyTorch environment using either one or eight GPUs, as specified by their training settings.

Table A1: Hardware and software environment used for data generation, policy learning, and closed-loop evaluation.

Component	Version / configuration
Operating system	Ubuntu 22.04.5 LTS; Linux 6.8; glibc 2.35
Simulation	Isaac Sim 5.1.0; PhysX; Python 3.11.13
Flight control	PX4-Autopilot SITL 1.14.3; pymavlink 2.4.49
Isaac-side Python	NumPy 1.26.4; SciPy 1.15.3; Pillow 12.1.0; Flask 3.1.2
Policy environment	CPython 3.10.19; PyTorch 2.7.1+cu128; CUDA 12.8; torchvision 0.22.1+cu128
Learning libraries	Transformers 5.3.0; Accelerate 1.13.0; DeepSpeed 0.18.5; PEFT 0.18.1; W&B 0.23.1

A.4 Code and Data Availability

The code package provided with this submission includes the code required for dataset construction, synchronized Isaac Sim–PX4 execution, policy training and inference, benchmark evaluation, and metric computation, together with the model, optimization, execution, and evaluation settings used in the experiments. Code written by the authors will be distributed under BSD-3-Clause, and SUDA-UAV trajectories, annotations, and metadata under CC BY 4.0. Pegasus Simulator, PX4-Autopilot, external dependencies, and third-party assets retain their original licenses.

B SUDA-UAV DATASET AND BENCHMARK

B.1 DATASET COMPOSITION

SUDA-UAV comprises 13,678 controller-executed trajectories and 1,601,000 synchronized RGB frames recorded at 20 Hz (Figure A1). The training split contains 13,478 trajectories spanning 22.05 hours according to per-trajectory metadata, and 200 trajectories form the benchmark. Frame counting gives a full-collection equivalent of 22.24 hours at 20 Hz. The synchronized records yield 1,505,254 indexed motion-response windows. Following the temporally grouped action formulations of Qwen-VLA and GR00T N1 Wang et al. (2026); NVIDIA et al. (2025a), QwenVLA-style policies learn complete $H = 16$ Action Chunks and execute an eight-step stride at each policy step.

Table A2: SUDA-UAV training and benchmark split.

Scope	Trajectories
Training index	13,478
Held-out benchmark	200
Full dataset	13,678

Table A3 uses the task-family and object-context categories shown in Figure A1 and Figure 2 of the main paper.

Table A3: SUDA-UAV composition under the task-family and object-context classification used in Fig. 2. Shares are rounded to one decimal place. The duration column gives the 22.24 h full-collection frame equivalent at 20 Hz; the 13,478 training trajectories span 22.05 h according to per-trajectory metadata. The synchronized index yields 1,505,254 motion-response windows.

Task family	Object context	Trajectories	Share (%)	Frames	20 Hz duration	Mean steps
Move	—	4,028	29.4	398,252	5.531 h	97.9
Object	Tree	2,770	20.3	213,794	2.969 h	76.2
Object	Streetlight	1,374	10.0	120,842	1.678 h	86.9
Object	Person	880	6.4	61,585	51.32 min	69.0
Object	Car	898	6.6	81,722	1.135 h	90.0
Object	Dog	554	4.1	35,654	29.71 min	63.4
Vertical	—	1,216	8.9	131,510	1.827 h	107.1
Vertical	Tree	390	2.9	47,779	39.82 min	121.5
Yaw	—	584	4.3	70,126	58.44 min	119.1
Yaw	Person	314	2.3	11,196	9.33 min	34.7
Yaw	Dog	224	1.6	8,452	7.04 min	36.7
Circle	Person	274	2.0	260,494	3.618 h	949.7
Circle	Dog	172	1.3	159,594	2.217 h	926.9
Total	—	13,678	100.0	1,601,000	22.24 h	—

Across the full SUDA-UAV index, the root-mean-square (RMS) start-of-interval FLU body displacement is 6.66 cm per 20 Hz step, and the RMS current-frame FLU body velocity is 1.33 m/s. These statistics characterize the scale of the 15D motion-response vectors. Per-axis 1st and 99th percentiles provide the normalization bounds used for policy learning.

B.2 CURATION

Controller-executed trajectories from UAV-Flow-Sim task references and the augmented references used to construct SUDA-UAV were consolidated under a common task index. We standardized instructions involving left/right relations, circle direction, vertical and angular displacement, distance fields, and rotation-side wording. Frame sequences were validated before temporal indexing, and the resulting trajectories were checked for temporal and annotation consistency.

B.3 BENCHMARK COMPOSITION

The benchmark contains 200 tasks sampled uniformly from the full SUDA-UAV collection and shared by all comparison policies. It uses a common PX4 SITL protocol, task-aware SR criteria, and shared geometric metrics. Ten semantic subtypes are grouped into seven evaluation categories: object approach, object side/target relation, object pass-through/away, synthetic translation, vertical motion, yaw-to-target, and circle.

B.4 Benchmark Split and Visual-Domain Control

Trajectory-level isolation. The benchmark is constructed by uniformly sampling 200 complete trajectory IDs before policy training. All observations, controller setpoints, executed responses, and indexed Action Chunk windows associated with these IDs are excluded from optimization. The same fixed split is used by every policy family, while preserving the common scene and task distribution for controlled in-distribution comparison.

Visual-domain control. For the QwenVLA-style paired comparison, both checkpoints receive the same SUDA-SIM RGB observations during online evaluation. The comparison reports end-to-end SUDA-UAV adaptation, while task IDs, initial states, reference trajectories, PX4 execution, action budgets, and metric computation remain fixed.

Per-task records include SR, nDTW, sDTW contributions, ATE, FDE, OR@1m, instruction subtype, and object category, supporting both aggregate and category-level analysis. Complete metrics for every policy family are reported below.

The benchmark includes 102 object-relative, 45 synthetic-move, 32 yaw-to-target, 11 vertical, and 10 circle instructions. Its semantic subtypes comprise 47 object-approach, 21 object-side-relation, 16 object-target, 17 object-pass-through, 1 object-away, 45 synthetic-move, 32 yaw-to-target, 9 vertical, 2 vertical-side-relation, and 10 circle tasks. The corresponding object-category counts are 43 for person, 41 for tree, 35 for dog, 15 for streetlight, and 4 for car. Of the 200 tasks, 138 are anchored to an object or scene element and 62 require no target object.

After coordinate and task-reference correction, reference trajectories range from 0.02 m to 17.47 m, with a median of 1.00 m and a mean of 2.85 m. Controller-executed 3D paths contain 10 to 622 points, with a median of 31 and a mean of 66.62 points per trajectory.

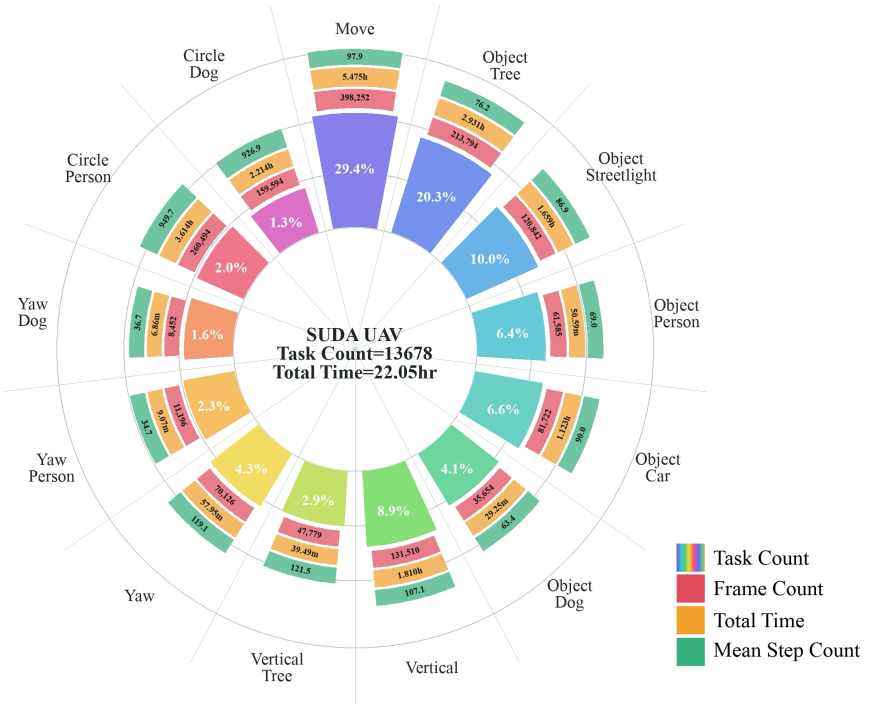


Figure A1: Task distribution and scale of SUDA-UAV. Radial sectors show category proportions. Outer bars report frame count, training-split duration, and mean number of steps for each category.

B.5 Evaluation Metrics and Completion Criteria

Closed-loop SR measures whether a rollout fulfills its natural-language instruction under the criterion defined for that task family. All policies use the same PX4 SITL environment. When a policy's original output differs from the SUDA motion representation, a policy-specific projection precedes

the common PX4 mapping. The mapping combines accumulated position and yaw targets with predicted velocity and yaw-rate feed-forward.

A single-step policy executes one execution-facing projection before receiving the next observation; a chunked policy executes an eight-step stride from its 16-step prediction. Both supervision conditions use the same 15D policy output representation and the same execution adapter during closed-loop evaluation, and both use the same 20 Hz PX4 execution protocol. For chunked policies, a jointly predicted sequence of future motion vectors forms the Action Chunk. Geometric targets are defined by the corrected reference trajectories, whereas policy motion results from controller-executed flight dynamics.

For benchmark task i, let $\mathcal{P}_i = \{\boldsymbol{r}_{i,1}, \ldots, \boldsymbol{r}_{i,T_i}\}$ denote the recorded 3D position sequence of the controller-executed rollout, $\mathcal{R}_i$ the reference trajectory, and $\boldsymbol{p}_i^{\mathrm{goal}}$ the reference endpoint. A category-specific predicate yields the binary completion indicator $S_i \in \{0, 1\}$:

$$\mathrm{SR} = \frac{1}{N} \sum_{i=1}^{N} S_i \times 100\%.$$

OR@1m records whether any point on the complete rollout path lies within 1 m of the reference endpoint:

$$\mathrm{OR@1m} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1} \left[\min_{1 \le t \le T_i} \|\boldsymbol{r}_{i,t} - \boldsymbol{p}_i^{\mathrm{goal}}\|_2 \le 1\,\mathrm{m} \right].$$

For nDTW, let $\mathcal{P}'_i$ and $\mathcal{R}'_i$ denote the controller-executed and reference path sequences used for benchmark evaluation. We compute xyz DTW with Euclidean pointwise costs and normalize it as

$$\begin{aligned} D_i^{\mathrm{DTW}} &= \mathrm{DTW}_{xyz}(\mathcal{P}'_i, \mathcal{R}'_i), \\ L_i^{\mathrm{nDTW}} &= \ell_{\mathrm{nDTW}} \min(|\mathcal{P}'_i|, |\mathcal{R}'_i|), \\ \mathrm{nDTW}_i &= \exp\left(-\frac{D_i^{\mathrm{DTW}}}{L_i^{\mathrm{nDTW}}} \right), \qquad \ell_{\mathrm{nDTW}} = 1\,\mathrm{m}. \end{aligned}$$

The reported nDTW is point-count-normalized by the shorter of the two downsampled path sequences. For ATE, both original paths are linearly resampled by index to $n_i = \max(2, \min(|\mathcal{P}_i|, |\mathcal{R}_i|))$ points. Denoting the resampled points by $\widetilde{\boldsymbol{p}}_{i,j}$ and $\widetilde{\boldsymbol{q}}_{i,j}$, we compute

$$\begin{aligned} \mathrm{ATE}_i &= \frac{1}{n_i} \sum_{j=1}^{n_i} \|\widetilde{\boldsymbol{p}}_{i,j} - \widetilde{\boldsymbol{q}}_{i,j}\|_2, \\ \mathrm{FDE}_i &= \|\boldsymbol{r}_{i,T_i} - \boldsymbol{p}_i^{\mathrm{goal}}\|_2, \\ \mathrm{sDTW} &= \frac{1}{N} \sum_{i=1}^{N} S_i\, \mathrm{nDTW}_i. \end{aligned}$$

FDE is the Euclidean distance between the rollout and reference endpoints. sDTW weights each task's nDTW by its completion indicator. We report nDTW, ATE, and FDE as unweighted arithmetic means over the N benchmark tasks.

All policies share the same PX4 SITL protocol, completion predicates, task targets, and reference trajectories. Each rollout uses the action budget of its task reference, and success is evaluated after the rollout completes, never as an early-termination condition. Every 20 Hz control substep advances exactly ten 5 ms physics ticks, and termination occurs only after all assigned ticks have been consumed. The same coordinate corrections for vertical motion and synthetic translation are applied to every policy.

B.5.1 Task-Specific Completion Predicates

The predicates in Table A4 use the geometric quantities relevant to each instruction. Translation and object-relative tasks combine endpoint, nearest-approach, and reference-prefix distances with progress. Vertical tasks use vertical and horizontal errors; yaw tasks use final yaw error and drift; and circle tasks use swept angle, side consistency, and radial error. Completion is evaluated in the rollout coordinate system, while nDTW, sDTW, ATE, FDE, and OR@1m characterize path alignment and endpoint geometry.

Table A4: Task-aware closed-loop completion predicates for SR. Endpoint, oracle, and prefix denote the final endpoint distance, minimum rollout-to-reference distance, and nearest-prefix distance. Progress terms are normalized along the reference trajectory.

Instruction family / subtype	Completion criterion
move / synthetic move	Endpoint $\leq 0.65\,\mathrm{m}$, or oracle $\leq 0.80\,\mathrm{m}$ with progress ≥ 0.10, or prefix $\leq 0.60\,\mathrm{m}$ with progress ≥ 0.40.
object / approach	Endpoint $\leq 0.60\,\mathrm{m}$, or oracle $\leq 0.50\,\mathrm{m}$, or prefix $\leq 0.32\,\mathrm{m}$ with progress ≥ 0.10.
object / target	Endpoint $\leq 1.05\,\mathrm{m}$, or oracle $\leq 1.00\,\mathrm{m}$ with progress ≥ 0.20, or prefix $\leq 0.75\,\mathrm{m}$ with progress ≥ 0.30.
object / side relation	Endpoint $\leq 0.50\,\mathrm{m}$, or oracle $\leq 0.50\,\mathrm{m}$, or prefix $\leq 0.50\,\mathrm{m}$ with progress ≥ 0.10.
object / away	Endpoint $\leq 0.75\,\mathrm{m}$, or oracle $\leq 1.00\,\mathrm{m}$, or prefix $\leq 0.45\,\mathrm{m}$ with progress ≥ 0.20.
object / pass-through	Endpoint $\leq 0.75\,\mathrm{m}$, or oracle $\leq 1.00\,\mathrm{m}$, or prefix $\leq 0.60\,\mathrm{m}$ with progress ≥ 0.20.
vertical / vertical	Vertical error $\leq 0.25\,\mathrm{m}$ and horizontal error $\leq 1.25\,\mathrm{m}$, or prefix $\leq 0.65\,\mathrm{m}$ with progress ≥ 0.40.
vertical / side relation	Vertical error $\leq 0.70\,\mathrm{m}$, horizontal error $\leq 1.25\,\mathrm{m}$, and progress ≥ 0.20.
yaw / yaw-to-target	Final yaw error $\leq 45^\circ$ with drift $\leq 0.75\,\mathrm{m}$.
circle / circle	Swept angle at least 45°, side consistency ≥ 0.55, and radial error $\leq 0.35\,\mathrm{m}$.

C Execution and Coordinate Protocol

C.1 Reset and Lockstep Execution

Before each rollout, PX4 is stopped and the aircraft and dynamic objects are restored from task metadata, with object height fixed at $z = 0$. PX4 then restarts, takes off, stabilizes in hover, and passes the idle lockstep gate. Rendering follows the live aircraft pose with zero yaw offset. The simulation timeline remains paused during policy inference and advances only by the ticks assigned to the subsequent command.

C.2 Source Frames and Motion-Response Construction

UAV-Flow-Sim task references originate from an AirSim simulation built on Unreal Engine. AirSim reports vehicle kinematics in north-east-down (NED) coordinates and SI units, while raw Unreal coordinates are left-handed, Z-up, and measured in centimeters. SUDA-SIM converts both forms to its east-north-up (ENU) execution frame before PX4 execution. Let $\boldsymbol{p}^{\mathrm{UE}} = [x_{\mathrm{fwd}}, y_{\mathrm{right}}, z_{\mathrm{up}}]^\top$ denote an origin-aligned Unreal position in centimeters. Its SUDA ENU coordinates are

$$\boldsymbol{p}^{\mathrm{ENU}} = \frac{1}{100} \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{C}_{\mathrm{UE}\to\mathrm{ENU}}} \boldsymbol{p}^{\mathrm{UE}},$$

where the scale factor converts centimeters to meters. Rotations are transformed in matrix form:

$$\boldsymbol{R}^{\mathrm{ENU}} = \boldsymbol{C}_{\mathrm{UE}\to\mathrm{ENU}} \boldsymbol{R}^{\mathrm{UE}} \boldsymbol{C}_{\mathrm{UE}\to\mathrm{ENU}}^\top.$$

The relative Euler component is extracted after this transformation. Source poses exported through the AirSim API use the NED-to-ENU conversion

$$\boldsymbol{p}^{\mathrm{ENU}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}}_{\boldsymbol{C}_{\mathrm{NED}\to\mathrm{ENU}}} \boldsymbol{p}^{\mathrm{NED}}.$$

Each 15D motion-response vector is derived from consecutive controller-executed PX4 state/feedback records. Let $\boldsymbol{p}_t^E$ and $\boldsymbol{v}_t^E$ denote ENU position and velocity, $\boldsymbol{q}_t^{\mathrm{att}}$ the unit attitude quaternion, and $R(\boldsymbol{q}_t^{\mathrm{att}})$ the corresponding rotation from the FLU body frame to ENU:

$$\begin{aligned} \boldsymbol{d}_t &= R(\boldsymbol{q}_{t-1}^{\mathrm{att}})^\top (\boldsymbol{p}_t^E - \boldsymbol{p}_{t-1}^E), \\ \boldsymbol{\eta}_t &= \mathrm{wrap}_{[-\pi,\pi)} \left[\mathrm{Euler}_{xyz}(\boldsymbol{q}_t^{\mathrm{att}} \otimes (\boldsymbol{q}_{t-1}^{\mathrm{att}})^{-1}) \right], \\ \boldsymbol{v}_t &= R(\boldsymbol{q}_t^{\mathrm{att}})^\top \boldsymbol{v}_t^E. \end{aligned} \tag{A1}$$

In Equation (A1), displacement uses the start-of-interval FLU frame and velocity the current FLU frame, with x forward, y left, and z up. Angular velocity $\boldsymbol{\omega}_t$ and linear acceleration $\boldsymbol{a}_t$ use synchronized body-frame records after the PX4 forward-right-down (FRD) to FLU sign conversion. The relative rotation is represented in the ENU/world frame with wrapped roll–pitch–yaw components. The first record has zero displacement and relative rotation. This construction aligns each motion-response vector with the executed pose and rendered observation carrying the same timestamp. The controller-facing yaw-aligned mapping used at rollout is defined separately in the Policy-to-PX4 Execution section.

C.3 Reference-Derived Motion Construction

The UAV-Flow-Sim source provides native 6-DoF reference pose trajectories. For the Reference Supervision condition, each trajectory is converted to the same 15D channel layout used for controller-executed supervision:

$$\boldsymbol{o}_t^{\mathrm{ref}} = [(\boldsymbol{d}_t^{\mathrm{ref}})^\top, (\boldsymbol{\eta}_t^{\mathrm{ref}})^\top, (\boldsymbol{v}_t^{\mathrm{ref}})^\top, (\boldsymbol{\omega}_t^{\mathrm{ref}})^\top, (\boldsymbol{a}_t^{\mathrm{ref}})^\top]^\top \in \mathbb{R}^{15}. \tag{A2}$$

Relative displacement and rotation are obtained from consecutive reference poses, while linear velocity, angular velocity, and linear acceleration are deterministically computed by temporal differentiation of the aligned reference sequence, using the same temporal spacing and coordinate convention as Equation (A1). All five components are therefore functions of the native reference trajectory alone, and the resulting supervision contains no controller-executed information. Both supervision conditions share this 15D motion representation; they differ in the provenance of the motion sequence, and only the controller-executed condition is termed motion response.

D QwenVLA-Style Policy Details

D.1 Architecture and Configuration

SUDA-adapted QwenVLA-style variants condition on the current 15D motion state and predict the next-step vector: Controller-Executed Supervision uses $\boldsymbol{s}_t := \boldsymbol{o}_t^{\mathrm{act}}$ and $\boldsymbol{o}_{t+1}^{\mathrm{act}}$, and Reference Supervision uses $\boldsymbol{s}_t := \boldsymbol{o}_t^{\mathrm{ref}}$ and $\boldsymbol{o}_{t+1}^{\mathrm{ref}}$ (Section C.3). Inputs and targets share channel order, UAV-centric frames, units, and percentile normalization. Settings appear in Table A6.

Table A5: Shared architecture settings for the SUDA-adapted QwenVLA-style variants, which predict 16 steps and execute an eight-step stride.

Setting	Value	Setting	Value
Backbone	Qwen3-VL-4B-Instruct	Prompt template	SUDA embodiment/control instruction + task instruction
SUDA-variant inputs	RGB, instruction, current 15D motion vector	Action history	disabled
Prediction output	16-step Action Chunk of 15D motion vectors	Execution adapter	Position–velocity–yaw/yaw-rate PX4 mapping
LoRA r/α/dropout	16 / 32 / 0.05	DiT width/layers/heads	2048 / 16 / 32; head dimension 64
DiT feed-forward dimension	11,136	Maximum sequence length	2,048
Flow timestep buckets	1,000	Prediction horizon H	16
Maximum / active action dim.	32 / 15	Action readout	Seeded 10-step flow integration
Flow latent initialization	Gaussian noise; minimum std 10^{-4}	Training precision	bfloat16
Motion loss	Uniform masked MSE over valid channels	Normalization	Dataset-wide 1st/99th percentiles

D.2 Hyperparameter Selection

We varied the action-decoder learning rate over $\{5\times10^{-5}, 1\times10^{-4}, 1.5\times10^{-4}\}$, the VLM learning rate over $\{2\times10^{-6}, 5\times10^{-6}, 1\times10^{-5}\}$, and action-model dropout over $\{0, 0.05, 0.1\}$. An initial

$2\times2\times2$ grid covered all combinations of the first two learning-rate values and dropout $\{0, 0.1\}$. Full-data experiments also evaluated the final settings in Tables A5 and A6. Checkpoint selection used held-out action MSE; closed-loop benchmark metrics were computed for the selected checkpoints. LoRA rank, action horizon, active action dimension, and the number of flow-solver steps remained fixed.

D.3 Run Counts and Statistical Reporting

Each reported checkpoint was trained once and evaluated once on the fixed 200-task benchmark. The reported metrics are aggregate values from that single evaluation pass. Paired task-level bootstrap intervals over the 200 evaluation tasks are reported in Table A10; these intervals quantify task-level variation for the fixed trained checkpoints and do not capture training-seed variance.

D.4 Ablation Configuration and Alignment

Table A6 aligns every checkpoint in the main-paper ablation table under the same reporting fields. The data-supervision block compares Reference Supervision and Controller-Executed Supervision under a matched QwenVLA-style configuration: both use the same architecture, 15D motion representation, prediction horizon, initialization, optimization schedule, effective batch size, and inference procedure, and differ in whether the supervision represents intended reference motion or realized controller-executed motion. The table reports their inputs, targets, schedules, batch regimes, and learning rates together with the stage-specific settings for T2A, PPO, and DepthAux.

Table A6: Executed training settings for all QwenVLA-style checkpoints in the main-paper ablation table. Each row reports the realized setting for that checkpoint.

Checkpoint / condition	Training input and supervision	Action target	Starting point	Updates / warmup	Effective batch / accumulation	Learning rate
SFT (LoRA), w/o SUDA	SUDA-UAV RGB, instruction, and current 15D reference-derived motion state at 20 Hz; Reference Supervision	16×15 reference-derived motion block	Qwen3-VL-4B; dataset-matched T2A decoder	30,000 / 1,000	64 / 8	decoder 5×10^{-5}; LoRA 2×10^{-6}
T2A, −Image	SUDA-UAV instruction and current 15D response at 20 Hz; no RGB	16×15 motion-response block	frozen Qwen3-VL-4B; train flow decoder	30,000 / 1,000	256 / 1	decoder 5×10^{-5}
SFT (LoRA), w/ SUDA	SUDA-UAV RGB, instruction, and current 15D controller-executed motion state at 20 Hz; Controller-Executed Supervision	16×15 motion-response block	Qwen3-VL-4B; dataset-matched T2A decoder	30,000 / 1,000	64 / 8	decoder 5×10^{-5}; LoRA 2×10^{-6}
PPO-based RL, +RL	Online SUDA-SIM rollouts at 20 Hz with the SFT policy inputs	16×15 block; first 8 steps sampled	SFT (LoRA) w/ SUDA-UAV	511; 4,085 trajectories	8 environments; micro-batch 4	LoRA 2×10^{-8}; decoder 5×10^{-8}
DepthAux, +Depth	SUDA-UAV SFT inputs at 20 Hz plus frozen VQ-VAE depth targets	16×15 motion-response block	SFT architecture plus depth head	15,000 / 1,000	256 / 1	SFT rates; depth head 1×10^{-4}

All five checkpoints are evaluated once on the same fixed 200-task split. Online observations use SUDA-SIM; task IDs, initial states, reference trajectories, 20 Hz PX4 SITL execution, eight-step rollout stride, action budgets, success predicates, and metric computation remain fixed. T2A omits the rendered RGB input by design, while the other four checkpoints consume it.

T2A, both SFT conditions, and DepthAux use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon_{\text{Adam}} = 10^{-8}$, optimizer weight decay 10^{-8}, gradient norm 1.0, a cosine schedule with minimum learning rate 10^{-6}, bfloat16 precision, and DeepSpeed ZeRO-2. RGB inputs are resized to 224×224. Effective

Table A7: Executed optimization settings for the eight cross-policy comparison families evaluated in the main paper.

Policy / variant	Optimizer	Training budget	Batch regime	Learning rate	Adaptation setting
OpenVLA-7B	AdamW	15,000 updates	256 effective	5×10^{-4}	LoRA $r=8, \alpha=8$
UAV-Flow-OpenVLA	AdamW	15,000 updates	256 effective	5×10^{-4}	LoRA $r=8, \alpha=8$
OpenVLA-OFT	AdamW	15,000 updates	256 effective	5×10^{-4}	LoRA $r=8, \alpha=16$ + decoder
ABot-M0	AdamW	15,000 updates	256 effective	5×10^{-4}	LoRA $r=8, \alpha=16$ + action modules
OpenPI-0.5	AdamW	15,000 updates	256 effective	5×10^{-4}	LoRA $r=8, \alpha=16$ + projections
LingBot-VLA	AdamW	15,000 updates	256 effective	5×10^{-4}	LoRA $r=8, \alpha=16$ + projections
FastWAM	AdamW	15,000 updates	256 effective	5×10^{-4}	LoRA $r=8, \alpha=16$ + ActionDiT
VAMPO	AdamW	15,000 updates	256 effective	1×10^{-4}	latent width 384; action window 16

batch sizes, accumulation factors, realized updates, and warmup lengths are checkpoint-specific and reported in Table A6.

For PPO, the implementation uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon_{\text{Adam}} = 10^{-8}$, and optimizer weight decay 10^{-8}. Advantages are standardized within each update batch, the policy micro-batch size is 4, and one optimizer step is taken per update. No separate value-network or GAE parameters are used.

D.5 Flow-Matching Action Decoder

D.5.1 Masked Training Objective

The SUDA-adapted QwenVLA-style policy adopts the flow-matching action decoder of QwenVLA Wang et al. (2026), using a Qwen3-VL-4B backbone and a SUDA action tensor $\boldsymbol{Y}_0 \in \mathbb{R}^{16\times32}$. The first 15 channels at each of 16 steps form the Action Chunk, selected by the binary validity mask $\boldsymbol{M}_t$:

$$M_{t,h,k} = \mathbf{1}[1 \le h \le H]\,\mathbf{1}[1 \le k \le D_{\text{act}}],$$
$$H = 16, \qquad D_{\text{act}} = 15.$$

The valid tensor entries form a 16-step Action Chunk of 15D motion vectors.

During training, we sample

$$\boldsymbol{Y}_1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \qquad \tau \sim \mathcal{U}(0,1),$$

and form

$$\boldsymbol{Y}_\tau = (1-\tau)\boldsymbol{Y}_0 + \tau\boldsymbol{Y}_1.$$

The target velocity is

$$\boldsymbol{u}^\star = \boldsymbol{Y}_1 - \boldsymbol{Y}_0,$$

and the masked loss is

$$\mathcal{L}_{\text{FM}} = \frac{\|\boldsymbol{M}_t \odot (\boldsymbol{u}_\theta - \boldsymbol{u}^\star)\|_F^2}{\max(\|\boldsymbol{M}_t\|_1, 1)}.$$

All valid motion channels receive equal weight in the masked MSE.

D.5.2 Deterministic Flow Integration

At inference, the decoder begins from Gaussian noise and performs 10 flow-solver steps with fixed seed 42, making the resulting Action Chunk reproducible under the update

$$\boldsymbol{Y}^{(0)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),$$
$$\tau_k = 1 - k/10,$$
$$\boldsymbol{Y}^{(k+1)} = \boldsymbol{Y}^{(k)} - \frac{1}{10} f_\theta\Big(\boldsymbol{Y}^{(k)}, \tau_k \mid Z_t\Big), \qquad k = 0, \dots, 9.$$

D.6 Training Stages and Auxiliary Supervision

D.6.1 T2A: Text-to-Action Pretraining

T2A keeps the Qwen3-VL-4B backbone fixed and omits image input. The instruction text enters through the SUDA embodiment/control prompt, and proprioceptive tokens encode the current 15D motion-response vector. Training updates the 16-layer DiT flow-matching action decoder.

SUDA-adapted T2A predicts a 16-step Action Chunk of 15D motion vectors. Eight steps are executed before the next policy step, when multimodal stages receive a new RGB observation and current response. Each SFT (LoRA) condition initializes its action decoder from the corresponding T2A stage and then enables image–instruction–response conditioning.

D.6.2 SFT (LoRA): Multimodal Supervised Fine-Tuning

SFT (LoRA) jointly optimizes the action decoder and Qwen3-VL-4B LoRA parameters under image–instruction–response conditioning. LoRA uses $r = 16$, $\alpha = 32$, and dropout 0.05 on `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. Both SFT conditions follow the same two-stage procedure and use the same 15D Action Chunk interface: Controller-Executed Supervision is constructed from synchronized PX4-executed state/feedback records, and Reference Supervision is constructed from the native 6-DoF UAV-Flow-Sim trajectory as defined in Section C.3. All remaining SFT settings are shared between the two conditions.

D.6.3 PPO Post-Training

PPO post-training starts from the SFT (LoRA) w/ SUDA-UAV policy and uses the clipped policy-ratio objective of PPO Schulman et al. (2017). The reported run resumes its PPO trajectory at update 400 and reaches 511 updates over 4,085 rollout trajectories. Given a deterministically seeded Gaussian latent $\boldsymbol{\xi}_i$, 10-step flow integration produces the normalized prediction $\boldsymbol{\mu}_\theta(I_i, q_i, \boldsymbol{s}_i; \boldsymbol{\xi}_i)$ conditioned on the onboard image, instruction, and current 15D motion-response vector. This prediction parameterizes a fixed-variance Gaussian exploration distribution. We sample $\widetilde{\boldsymbol{Y}}_i \sim \mathcal{N}(\boldsymbol{\mu}_\theta, \sigma^2 \boldsymbol{I})$ with fixed $\sigma = 0.2$ and set $\boldsymbol{Y}_i = \mathrm{clip}(\widetilde{\boldsymbol{Y}}_i, -1, 1)$. The PPO surrogate log-density is evaluated on $\widetilde{\boldsymbol{Y}}_i$, while $\boldsymbol{Y}_i$ is used for execution; the old and updated policy ratios share the same $\boldsymbol{\xi}_i$. Evaluation uses the deterministically seeded flow mean. With horizon $H = 16$, $\boldsymbol{A}_i = (Y_{i,h,j})_{h=1,\ldots,8;\, j=1,\ldots,15}$ denotes the normalized eight-step sample used for a rollout stride. The execution adapter receives seven channels per step, whereas the PPO surrogate log-density covers all 15.

The execution adapter maps each projection to PX4 position, velocity, yaw, and yaw-rate setpoints. Predicted displacement accumulates into the position setpoint. Predicted body-z angular rate is integrated into the yaw setpoint and is also applied as yaw-rate feed-forward; predicted velocity is applied directly as velocity feed-forward. The shaped reward is computed after PX4 has executed the eight-step stride. Single-step policies use the same adapter and receive an updated observation after one step.

For each update batch $\mathcal{B}$, the shaped stride rewards are standardized as

$$\hat{S}_i = \frac{R_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}} + \varepsilon_{\mathrm{std}}}. \tag{A3}$$

Here, $\varepsilon_{\mathrm{std}} > 0$ is a numerical stabilizer. We use $\epsilon_{\mathrm{PPO}} = 0.2$. Batch standardization produces the policy-update weight $\hat{S}_i$, and the Gaussian covariance remains fixed during optimization.

Let $\ell_{\theta,i}^{\mathrm{joint}}$ and $\ell_{\theta_{\mathrm{old}},i}^{\mathrm{joint}}$ denote the Gaussian surrogate log-densities over all 15 channels of the unclipped eight-step sample. The seven execution-facing channels affect the PX4 setpoints directly. The stride weight $\hat{S}_i$ from Equation (A3) reaches the eight response-only channels through the joint density and shared decoder, so both channel groups are updated during post-training. The policy ratio and its clipped form are

$$r_i(\theta) = \exp\left(\ell_{\theta,i}^{\mathrm{joint}} - \ell_{\theta_{\mathrm{old}},i}^{\mathrm{joint}}\right),$$
$$\bar{r}_i(\theta) = \mathrm{clip}(r_i(\theta), 1 - \epsilon_{\mathrm{PPO}}, 1 + \epsilon_{\mathrm{PPO}}).$$

The PPO update maximizes

$$\mathcal{J}_{\mathrm{PPO}} = \mathbb{E}_i\Big[\min\Big(r_i(\theta)\hat{S}_i, \bar{r}_i(\theta)\hat{S}_i\Big)\Big].$$

The reward combines dense goal-directed motion, fidelity to the reference trajectory, task completion, and regularization of execution-facing prediction magnitude. Let D_i^{goal} denote the distance to the reference endpoint and $\Delta D_i^{\mathrm{goal}} = D_{i-1}^{\mathrm{goal}} - D_i^{\mathrm{goal}}$ the progress over one execution stride.

D_i^{track} is the distance from the current controller-executed position to the corresponding reference point. Let E_i^{yaw} denote the absolute yaw error in degrees, Q_i^{nDTW} partial nDTW, and C_i^{path} the controller-executed-to-reference path-length ratio clipped to $[0, 1]$. The regularizers P_i^{xy} and P_i^{z} aggregate horizontal and vertical displacement magnitudes over the execution stride, while P_i^{yaw} aggregates integrated yaw increments from the execution-facing body-z angular-rate channel. $B_i^{\mathrm{task}} \in \{0, 1\}$ indicates whether the task-aware training completion predicate is satisfied. The resulting reward is

$$\begin{aligned} R_i = {} & 2.0\,\Delta D_i^{\mathrm{goal}} - D_i^{\mathrm{track}} - 0.5 D_i^{\mathrm{goal}} - 0.05\frac{E_i^{\mathrm{yaw}}}{180} \\ & + 0.5 Q_i^{\mathrm{nDTW}} + 0.2 C_i^{\mathrm{path}} - 0.02 P_i^{xy} \\ & - 0.05 P_i^{z} - 0.05 P_i^{\mathrm{yaw}} + 7.0 B_i^{\mathrm{task}}. \end{aligned} \tag{A4}$$

In Equation (A4), the $7.0 B_i^{\mathrm{task}}$ term rewards task-aware completion. Distances and displacement penalties are evaluated in meters, integrated-yaw penalties in radians, and E_i^{yaw} in degrees before division by 180. The coefficients set the relative scale of the reward terms before batch standardization.

D.6.4 DepthAux Variant

DepthAux adds simulator-rendered depth supervision to SFT (LoRA) while retaining the same RGB, instruction, and current 15D motion-response inputs and the same 16-step Action Chunk. A frozen depth VQ-VAE encoder–quantizer E_D, pretrained on SUDA depth data, maps the depth image D_t to a 64×256 quantized feature target:

$$\boldsymbol{Z}_t^D = \mathrm{sg}(E_D(D_t)), \qquad \boldsymbol{Z}_t^D \in \mathbb{R}^{64\times 256},$$

where sg stops gradients through the frozen target encoder. The DepthAux head pools the vision-language model (VLM) features Z_t and predicts the corresponding feature tensor:

$$\widehat{\boldsymbol{Z}}_t^D = G_\zeta(Z_t), \qquad \widehat{\boldsymbol{Z}}_t^D \in \mathbb{R}^{64\times 256}.$$

The auxiliary objective is

$$\mathcal{L}_{\mathrm{DepthAux}} = \mathcal{L}_{\mathrm{FM}} + \frac{\lambda_D}{64 \cdot 256}\left\|\widehat{\boldsymbol{Z}}_t^D - \boldsymbol{Z}_t^D\right\|_F^2,$$

where $\lambda_D = 0.05$. Depth supplies quantized feature targets during training; at inference, the DepthAux-trained policy uses RGB, instruction, and the current 15D motion-response vector to predict actions.

E Complete Results and Policy Adaptation

E.1 Complete Quantitative Results

Table A8 reports the complete trajectory and completion metrics for the within-family comparisons. Within each family, *w/ SUDA-UAV* denotes adaptation with controller-executed SUDA-UAV data, whereas *w/o SUDA-UAV* denotes adaptation with reference-trajectory supervision from the same tasks. Both variants use the same PX4 execution procedure.

Under the same benchmark protocol, the GPT-5.5 Policy generalist reference obtains 51.5% SR, 0.528 nDTW, 0.397 sDTW, 1.500 m ATE, 2.928 m FDE, and 0.640 OR@1m.

Table A8: Path-quality metrics (sDTW, ATE, FDE) with and without SUDA-UAV adaptation. SR and nDTW appear in Table 2.

Policy	ΔSR (pp)	sDTW ↑		ATE (m) ↓		FDE (m) ↓	
		w/o	w/	w/o	w/	w/o	w/
VAMPO	+14.0	0.347	0.514	1.002	0.853	1.545	1.302
FastWAM	+4.0	0.279	0.314	1.079	1.042	1.845	1.790
ABot-M0	+19.0	0.326	0.501	1.008	0.816	1.538	1.216
OpenVLA-OFT	+9.5	0.304	0.378	1.023	0.932	1.672	1.580
OpenPI-0.5	+14.5	0.262	0.388	1.085	1.122	1.810	1.785
OpenVLA-7B	+5.5	0.353	0.367	1.015	1.059	1.638	1.700
UAV-Flow-OpenVLA	+6.0	0.335	0.358	1.020	1.030	1.590	1.555
LingBot-VLA	+6.5	0.241	0.299	1.140	1.169	1.935	1.890
Mean	+9.9	0.306	0.390	1.047	1.003	1.697	1.602

E.2 Paired Task-Level Bootstrap Analysis

We resample the fixed 200 task IDs with replacement for 10,000 paired replicates, using seed 20260728 and the same sampled task counts for every checkpoint. Tables A9 and A10 report 95% percentile intervals for task-level metrics and paired effects. The analysis covers the checkpoints reported in the main paper and the GPT-5.5 Policy reference.

Table A9: Task-level point estimates with 95% percentile bootstrap confidence intervals on the fixed 200-task benchmark. SUDA denotes SUDA-UAV.

Checkpoint	SR (%)	nDTW	sDTW	ATE (m)	FDE (m)	OR@1m
PPO w/ SUDA	66.5 [60.0,73.0]	.630 [.591,.668]	.520 [.466,.575]	.768 [.640,.906]	1.401 [1.157,1.661]	.695 [.630,.755]
SFT (LoRA) w/ SUDA	64.5 [58.0,71.0]	.628 [.588,.666]	.506 [.451,.561]	.775 [.646,.914]	1.403 [1.161,1.665]	.680 [.615,.745]
DepthAux w/ SUDA	65.0 [58.5,71.5]	.617 [.579,.655]	.488 [.434,.543]	.812 [.679,.954]	1.465 [1.218,1.731]	.650 [.585,.715]
T2A w/ SUDA	27.5 [21.5,34.0]	.437 [.398,.475]	.191 [.148,.236]	1.567 [1.291,1.874]	2.727 [2.294,3.210]	.640 [.575,.705]
SFT (LoRA) w/o SUDA	41.5 [35.0,48.5]	.336 [.303,.369]	.210 [.171,.250]	1.680 [1.493,1.878]	3.200 [2.836,3.579]	.780 [.720,.835]
VAMPO w/ SUDA	63.0 [56.5,69.5]	.649 [.604,.693]	.514 [.455,.572]	.853 [.688,1.034]	1.715 [1.360,2.100]	.775 [.715,.830]
VAMPO w/o SUDA	49.0 [42.0,56.0]	.546 [.503,.587]	.347 [.292,.400]	1.002 [.852,1.162]	1.777 [1.522,2.052]	.635 [.565,.700]
FastWAM w/ SUDA	41.0 [34.0,47.5]	.532 [.490,.574]	.314 [.259,.368]	1.042 [.885,1.215]	1.846 [1.577,2.135]	.605 [.535,.670]
FastWAM w/o SUDA	37.0 [30.5,44.0]	.523 [.480,.564]	.279 [.228,.332]	1.079 [.915,1.254]	1.912 [1.622,2.228]	.640 [.570,.705]
ABot-M0 w/ SUDA	63.0 [56.0,69.5]	.644 [.599,.688]	.501 [.443,.559]	.816 [.670,.974]	1.629 [1.313,1.969]	.780 [.720,.835]
ABot-M0 w/o SUDA	44.0 [37.0,51.0]	.544 [.501,.586]	.326 [.271,.379]	1.008 [.855,1.171]	1.837 [1.561,2.132]	.635 [.570,.700]
OpenVLA-OFT w/ SUDA	51.5 [44.5,58.5]	.561 [.518,.602]	.378 [.320,.433]	.932 [.797,1.079]	1.613 [1.382,1.865]	.680 [.615,.745]
OpenVLA-OFT w/o SUDA	42.0 [35.0,49.0]	.536 [.493,.577]	.304 [.251,.359]	1.023 [.869,1.190]	1.788 [1.528,2.070]	.635 [.565,.700]
OpenPI-0.5 w/ SUDA	49.0 [42.0,56.0]	.563 [.517,.606]	.388 [.328,.446]	1.122 [.922,1.343]	2.269 [1.794,2.783]	.700 [.635,.765]
OpenPI-0.5 w/o SUDA	34.5 [28.0,41.0]	.527 [.483,.570]	.262 [.211,.313]	1.085 [.915,1.267]	1.811 [1.527,2.124]	.595 [.525,.665]
LingBot-VLA w/ SUDA	38.5 [31.5,45.0]	.516 [.473,.557]	.299 [.245,.352]	1.169 [.982,1.375]	2.128 [1.805,2.473]	.575 [.505,.645]
LingBot-VLA w/o SUDA	32.0 [25.5,38.5]	.506 [.464,.547]	.241 [.191,.291]	1.140 [.969,1.327]	1.953 [1.662,2.275]	.585 [.515,.650]
OpenVLA-7B w/ SUDA	52.5 [45.5,59.5]	.547 [.503,.588]	.367 [.311,.420]	1.059 [.892,1.240]	1.959 [1.652,2.288]	.645 [.580,.710]
OpenVLA-7B w/o SUDA	47.0 [40.0,54.0]	.541 [.498,.583]	.353 [.297,.407]	1.015 [.862,1.180]	1.806 [1.546,2.087]	.645 [.580,.710]
UAV-Flow-OpenVLA w/ SUDA	51.5 [44.5,58.5]	.546 [.502,.588]	.358 [.302,.412]	1.062 [.894,1.243]	1.963 [1.655,2.291]	.655 [.590,.720]
UAV-Flow-OpenVLA w/o SUDA	45.5 [38.5,52.5]	.538 [.494,.578]	.335 [.281,.388]	1.020 [.867,1.185]	1.811 [1.551,2.091]	.635 [.570,.700]
GPT-5.5 Policy	51.5 [44.5,58.0]	.528 [.480,.573]	.397 [.339,.452]	1.500 [1.219,1.815]	2.928 [2.359,3.570]	.640 [.570,.705]

Table A10: Paired task-level bootstrap effects for the aggregate and QwenVLA-style comparisons reported in the main paper. Brackets give 95% percentile confidence intervals.

Comparison (first minus second)	ΔSR (pp)	ΔnDTW	ΔsDTW	ΔOR@1m
Family mean, w/ vs. w/o SUDA-UAV	9.9 [7.2, 12.6]	.037 [.030, .044]	.084 [.063, .105]	.051 [.033, .071]
SFT (LoRA) vs. T2A, w/ SUDA-UAV	37.0 [30.0, 44.0]	.190 [.163, .220]	.316 [.262, .369]	.040 [.000, .080]
PPO vs. SFT (LoRA), w/ SUDA-UAV	2.0 [−1.5, 5.5]	.002 [−.002, .007]	.014 [−.004, .034]	.015 [−.015, .045]
DepthAux vs. SFT (LoRA), w/ SUDA-UAV	0.5 [−4.5, 5.5]	−.010 [−.017, −.003]	−.018 [−.048, .012]	−.030 [−.060, −.005]

Table A11 reports the end-to-end data-adaptation comparison and the method-level refinements. It includes SFT (LoRA) w/o and w/ SUDA-UAV, together with the SUDA-supervised T2A, PPO post-training, and DepthAux variants. The multimodal-supervision comparison uses SFT (LoRA) and T2A with SUDA-UAV supervision; T2A receives instruction text and the current 15D motion-response vector without RGB. The variants share the QwenVLA-style backbone, Action Chunk decoder, and inference procedure; DepthAux additionally uses depth-feature supervision during training. Both SFT checkpoints receive the same SUDA-SIM RGB observations during online evaluation; flight execution, task identities, initial states, action budgets, and metric computation remain fixed.

Table A11: Trajectory-error metrics (ATE, FDE) for QwenVLA-style checkpoints. SR, nDTW, sDTW, and OR@1m appear in Table 3.

Variant	N	ATE (m) ↓	FDE (m) ↓
SFT (LoRA) w/o SUDA-UAV	200	1.680	3.200
T2A w/ SUDA-UAV	200	1.567	2.727
SFT (LoRA) w/ SUDA-UAV	200	0.775	1.403
PPO post-training w/ SUDA-UAV	200	**0.768**	**1.401**
DepthAux w/ SUDA-UAV	200	0.812	1.465

Table A12: Training components, adapted parameter counts, objectives, and action representations for the evaluated policy families.

Policy	Adaptation / optimized components	Adapted params	Training objective	Original action output / horizon
QwenVLA-style policy variants				
T2A	16-layer DiT action decoder	1160.646M	Text-to-action flow matching; fixed VLM and no RGB	16×15 block in a 16×32 tensor; stride 8
SFT (LoRA) w/ SUDA-UAV	Qwen3-VL-4B LoRA + flow decoder	33.030M + 1160.646M	Multimodal SFT on 15D controller-executed targets	16×15 block in a 16×32 tensor; stride 8
PPO post-training	Qwen3-VL-4B LoRA + action decoder	33.030M + 1160.646M	PPO-based task-level post-training from SFT (LoRA) w/ SUDA-UAV	16×15 block in a 16×32 tensor; stride 8
DepthAux	Qwen3-VL-4B LoRA + decoder + depth-feature path	33.030M + 1160.646M + 19.421M	Multimodal SFT with depth-feature supervision	16×15 block in a 16×32 tensor; stride 8
SFT (LoRA) w/o SUDA-UAV	Qwen3-VL-4B LoRA + flow decoder	33.030M + 1160.646M	Multimodal SFT on 15D reference-derived motion targets	16×15 motion block in a 16×32 tensor; stride 8
Comparison policies				
OpenVLA-7B	OpenVLA-7B LoRA; discretized action tokens	27.707M LoRA	Token-level LoRA SFT on 15D motion-response vectors	One 15D motion-response vector, single-step
UAV-Flow-OpenVLA	OpenVLA-style LoRA with UAV-Flow action representation	27.707M LoRA	Action-token LoRA SFT on 15D motion-response vectors	One 15D motion-response vector, single-step
OpenVLA-OFT	OpenVLA-7B LoRA + continuous decoder	27.707M + 4.441M	LoRA SFT with 16-step regression	16-step Action Chunk with 15D motion-response vectors
ABot-M0	Qwen3-VL-4B LoRA + 14D modules + projection	20.136M + 5.605M	Regression from original actions to 15D motion-response vectors	Original 14D Action Chunk projected to 15D
OpenPI-0.5	PaliGemma LoRA + 32D action projections	14.179M + 2.177M	Regression from original actions to 15D motion-response vectors	Original 32D Action Chunk projected to 15D
LingBot-VLA	Qwen2.5-VL-3B LoRA + 15D/75D projections	20.682M + 0.139M	LoRA SFT with 15D projection training	Original 75D sequence; 16-step chunk, stride 8
FastWAM	PaliGemma LoRA + visual projection + ActionDiT	14.179M + 8.393M + 1020.902M	Rectified-flow prediction on 16-step Action Chunks	16-step Action Chunk with 15D motion-response vectors
VAMPO	Diffusion policy with observation/text projections and 15D action modules	14.669M	Flow matching on 16-step SUDA Action Chunks	16-step Action Chunk with 15D motion-response vectors

E.3 Policy Outputs and Adapted Parameters

Table A12 reports trainable parameters and optimized components. SUDA-adapted QwenVLA-style variants use the first 15 channels of the 16×32 tensor for both supervision conditions. Counts

include adapted LoRA weights, decoders, projections, or policy modules and exclude the frozen DepthAux VQ-VAE. All policies use the same 200-task PX4 SITL benchmark and PX4 Offboard mode. Single-step policies update observations after each substep; chunked policies execute an eight-step stride.

The evaluated families use several action-generation mechanisms. OpenPI-0.5 Black et al. (2025a), built on the π_0 flow model Black et al. (2025b), predicts low-level Action Chunks. LingBot-VLA Wu et al. (2026) maps discrete representations to continuous actions, OpenVLA-OFT Kim et al. (2025a) adds continuous action-sequence regression to OpenVLA Kim et al. (2025b), and ABot-M0 Yang et al. (2026) learns a chunked action manifold. The remaining families are OpenVLA-7B, UAV-Flow-OpenVLA, FastWAM, and VAMPO. When a policy's original action space differs from the SUDA representation, a policy-specific projection is applied before the common PX4 mapping.

F Policy-to-PX4 Execution

F.1 Receding-Horizon Execution

Policies that predict 15D Action Chunks are trained on segments sampled from Dynamics-aware Action Trajectories and retain this motion-response structure at inference. Single-step policies use the same vector-level mapping. At policy step t, let $\widehat{\boldsymbol{o}}^{\mathrm{act}}_{t,k} := \widehat{\boldsymbol{o}}^{\mathrm{act}}_{t+k+1}$, $k = 0, \ldots, 7$, denote the kth future motion-response vector selected for execution. Thus $h = k + 1$ matches the main-paper execution index. Its execution-facing projection is

$$\widehat{\boldsymbol{c}}^{\mathrm{exec}}_{t,k} = \left[\widehat{\boldsymbol{d}}^{\top}_{t,k}, \widehat{\boldsymbol{v}}^{\top}_{t,k}, \widehat{\omega}^{z}_{t,k}\right]^{\top} \in \mathbb{R}^7,$$

while relative rotation, body-x/y angular rates, and linear acceleration form eight additional executed-response targets that remain supervised, excluded from setpoint construction.

The execution adapter maps the seven selected channels to a PX4 Offboard setpoint whose active fields are position, velocity, yaw, and yaw-rate. Acceleration remains part of the supervised response. The mapping to the next control interval is

$$\boldsymbol{u}^{\mathrm{PX4}}_{t,k+1} = \Pi_{\mathrm{exec}}\left(\widehat{\boldsymbol{c}}^{\mathrm{exec}}_{t,k}; \boldsymbol{r}^{\mathrm{sp},E}_{t,k}, \psi^{\mathrm{sp},E}_{t,k}, \psi^{\mathrm{meas}}_{t,k}\right).$$

Here, $\boldsymbol{r}^{\mathrm{sp},E}_{t,k}$ and $\psi^{\mathrm{sp},E}_{t,k}$ are the preceding ENU position and yaw setpoints, and $\psi^{\mathrm{meas}}_{t,k}$ is the measured yaw. The output $\boldsymbol{u}^{\mathrm{PX4}}_{t,k+1}$ specifies local position, velocity, yaw, and yaw-rate for the next interval. The mapping Π_{exec} applies command clipping, body-to-ENU rotation, setpoint accumulation, and the final ENU-to-NED conversion. The recorded controller setpoint $\boldsymbol{u}^{\mathrm{PX4}}$ remains distinct from both the task reference and the predicted motion-response vector.

QwenVLA-style and other chunked policies use receding-horizon execution: they predict 16 motion-response vectors and execute the seven-channel execution-facing projections of the first eight before receiving an updated observation. At 20 Hz, the prediction horizon spans 0.8 s and the execution stride spans 0.4 s. Each substep advances ten lockstep physics ticks of 5 ms.

F.2 Command Construction

At substep k, the 15D motion-response vector is ordered as displacement $\boldsymbol{d}$, relative rotation $\boldsymbol{\eta}$, linear velocity $\boldsymbol{v}$, angular velocity $\boldsymbol{\omega}$, and linear acceleration $\boldsymbol{a}$. These groups correspond to `action[0:3]`, `action[3:6]`, `action[6:9]`, `action[9:12]`, and `action[12:15]`, respectively. The state labels follow Equation (A1); linear quantities use SI units, and angular channels use radians and radians per second.

For setpoint construction, the adapter interprets predicted displacement and linear velocity as yaw-aligned forward-left-up (FLU) command inputs (x forward, y left, z up). It interprets the predicted body-z angular-rate channel as the yaw-rate command; relative rotation, body-x/y angular rates, and linear acceleration remain supervised response channels, excluded from the setpoint mapping. The seven adapter-input channels are

$$\boldsymbol{d}^{\mathrm{cmd}}_{t,k} = \widehat{\boldsymbol{o}}^{\mathrm{act}}_{t,k}[0{:}3], \quad \boldsymbol{v}^{\mathrm{cmd}}_{t,k} = \widehat{\boldsymbol{o}}^{\mathrm{act}}_{t,k}[6{:}9],$$
$$\dot{\psi}^{\mathrm{cmd}}_{t,k} := \widehat{\boldsymbol{o}}^{\mathrm{act}}_{t,k}[11],$$

where $\boldsymbol{d}_{t,k}^{\mathrm{cmd}}$ and $\boldsymbol{v}_{t,k}^{\mathrm{cmd}}$ are the yaw-aligned displacement and velocity commands, and $\dot{\psi}_{t,k}^{\mathrm{cmd}} := \widehat{\omega}_{t,k}^{z}$ is the yaw-rate command. Velocity is taken directly from the predicted velocity channels. The element `action[5]` remains the supervised relative-Euler yaw, while `action[11]` supplies the controller-facing yaw-rate command. For control frequency f_c and $\Delta t = 1/f_c$, the executed yaw increment is

$$\delta\psi_{t,k}^{\mathrm{cmd}} = \dot{\psi}_{t,k}^{\mathrm{cmd}} \Delta t = \frac{\widehat{\boldsymbol{o}}_{t,k}^{\mathrm{act}}[11]}{f_c}.$$

The measured yaw rotates the command-frame displacement and velocity into the ENU world frame:

$$\Delta \boldsymbol{r}_{t,k}^{E} = R_z(\psi_{t,k}^{\mathrm{meas}}) \boldsymbol{d}_{t,k}^{\mathrm{cmd}},$$
$$\boldsymbol{v}_{t,k}^{E} = R_z(\psi_{t,k}^{\mathrm{meas}}) \boldsymbol{v}_{t,k}^{\mathrm{cmd}}.$$

The position and yaw targets accumulate from the previous setpoints:

$$\boldsymbol{r}_{t,k+1}^{\mathrm{sp},E} = \boldsymbol{r}_{t,k}^{\mathrm{sp},E} + \Delta \boldsymbol{r}_{t,k}^{E},$$
$$\psi_{t,k+1}^{\mathrm{sp},E} = \psi_{t,k}^{\mathrm{sp},E} + \delta\psi_{t,k}^{\mathrm{cmd}}.$$

At the first policy step, $\boldsymbol{r}_{1,0}^{\mathrm{sp},E}$ and $\psi_{1,0}^{\mathrm{sp},E}$ are the position and yaw targets established during hover and idle-gate initialization. The terminal setpoint of each eight-step stride initializes the next stride, preserving accumulation across policy steps. Velocity and yaw-rate feed-forward terms are assigned at each substep:

$$\boldsymbol{v}_{t,k+1}^{\mathrm{sp},E} = \boldsymbol{v}_{t,k}^{E}, \qquad \dot{\psi}_{t,k+1}^{\mathrm{sp},E} = \dot{\psi}_{t,k}^{\mathrm{cmd}}.$$

F.3 MAVLink Conversion and Limits

Setpoints are transmitted through `SET_POSITION_TARGET_LOCAL_NED` in `MAV_FRAME_LOCAL_NED`. ENU quantities are converted by

$$x^N = y^E, \qquad y^N = x^E, \qquad z^N = -z^E,$$
$$v_x^N = v_y^E, \qquad v_y^N = v_x^E, \qquad v_z^N = -v_z^E,$$
$$\psi^N = \frac{\pi}{2} - \psi^E, \quad \dot{\psi}^N = -\dot{\psi}^E.$$

Acceleration remains a supervised component of each 15D motion-response vector but is excluded from the MAVLink feed-forward fields. PX4 receives accumulated position and yaw targets together with model-predicted velocity and yaw-rate feed-forward:

$$\boldsymbol{v}_{t,k+1}^{\mathrm{sp},E} = \boldsymbol{v}_{t,k}^{E} = R_z(\psi_{t,k}^{\mathrm{meas}}) \widehat{\boldsymbol{o}}_{t,k}^{\mathrm{act}}[6{:}9],$$
$$\dot{\psi}_{t,k+1}^{\mathrm{sp},E} = \dot{\psi}_{t,k}^{\mathrm{cmd}} = \widehat{\boldsymbol{o}}_{t,k}^{\mathrm{act}}[11],$$
$$\delta\psi_{t,k}^{\mathrm{cmd}} = \frac{\widehat{\boldsymbol{o}}_{t,k}^{\mathrm{act}}[11]}{f_c}.$$

Before setpoint construction, the displacement norm is clipped to 0.5 m. The yaw-rate command $\dot{\psi}_{t,k}^{\mathrm{cmd}} = \widehat{\boldsymbol{o}}_{t,k}^{\mathrm{act}}[11]$ is clipped to $\pm 180^\circ/\mathrm{s}$, and its integrated increment $\delta\psi_{t,k}^{\mathrm{cmd}} = \dot{\psi}_{t,k}^{\mathrm{cmd}}/f_c$ to $\pm 45^\circ$. These limits are applied independently at every executed substep.

Rendering uses the live vehicle pose with zero yaw offset; this rendering convention leaves PX4 setpoints unchanged.

Vehicle placement, takeoff, hover, and idle-gate initialization use position and yaw setpoints. Policy rollout begins after this initialization is complete. Measured yaw enters the execution-side body-to-world transformation, while the policy receives the current 15D motion-response vector as input.