Supplementary Material for See, Point, Fly: A Learning-Free VLM Framework for Universal Unmanned Aerial Navigation

Chih Yao Hu[1*] **Yang-Sen Lin**[2*] **Yuna Lee**[2] **Chih-Hai Su**[2] **Jie-Ying Lee**[2]
Shr-Ruei Tsai[2] **Chin-Yang Lin**[2] **Kuan-Wen Chen**[2] **Tsung-Wei Ke**[1] **Yu-Lun Liu**[2]
[1] National Taiwan University, [2] National Yang Ming Chiao Tung University

https://spf-web.pages.dev

1 Overview

In this supplementary material, we present additional details and results to complement the main manuscript, "See, Point, Fly: A Learning-Free VLM Framework for Universal Unmanned Aerial Navigation" (hereafter referred to as the main submission or the main paper).

Section 2 elaborates on our proposed SPF (See, Point, Fly) framework, as detailed in the main submission. Section 3.1 outlines task specifications, evaluation protocols, and provides example prompts (Table 1) for simulated and real-world tasks. Section 3.2 details key implementation parameters, including adaptive scaling, VLM backend, control architecture, and latency. Section 3.3 directs the reader to demonstration videos of qualitative results. Section 3.4 presents an ablation study on our adaptive step-size controller, including setup, quantitative results (Table 2), and visual examples (Figure 1).

This supplementary material aims to provide a deeper understanding of our methodology (presented in the main paper), experimental rigor, and key component benefits. Our video viewer webpage attached along with this supplementary material also provides a comprehensive task demo for each type of real-world and simulator tasks.

2 Method Details

2.1 Reactive Control Loop Execution

Input Variables: Given the structured VLM output $O_t = \{U, V, d_{\text{adj}}\}$, as described in the main paper. We transfer (U, V) into normalized value $(U_{\text{norm}}, V_{\text{norm}})$.

Position Calculations: We use U_{norm} and V_{norm} to calculate the desired 3D displacement vector (S_x, S_y, S_z). Where α and β are the camera's horizontal and vertical half field-of-view angles, respectively. These calculations follow the pinhole camera model detailed in the main submission (see Section 3.3, Eq. 2 of the main paper for the specific formulation used).

$$S_x = U_{\text{norm}} \cdot d_{adj} \cdot \tan(\alpha), \quad S_y = d_{\text{adj}}, \quad S_z = V_{\text{norm}} \cdot d_{adj} \cdot \tan(\beta)$$

Control Parameters: We use the 3D displacement vector (S_x, S_y, S_z) to calculate the UAV control primitive displacement: $(\Delta\theta, \Delta\text{Pitch}, \Delta\text{Throttle})$. The derivation of these control primitives from the 3D displacement vector is detailed in the main submission (see Section 3.4 and Figure 3c of the

[*] The first two authors contributed equally

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

main paper for the specific formulation used).

$$\Delta\theta = \tan^{-1}\left(\frac{S_x}{S_y}\right), \quad \Delta\text{Pitch} = \sqrt{S_x^2 + S_y^2}, \quad \Delta\text{Throttle} = S_z$$

Duration Calculations: Afterwards, we use $(\Delta\theta, \Delta\text{Pitch}, \Delta\text{Throttle})$ and Pre-defined speed $(P_{\text{yaw}}, P_{\text{pitch}}, P_{\text{throttle}})$ to calculate the Duration of each primitive $(D_{\text{yaw}}, D_{\text{pitch}}, D_{\text{throttle}})$.

$$D_{\text{yaw}} = \frac{\Delta\theta}{P_{\text{yaw}}}, \quad D_{\text{pitch}} = \frac{\Delta\text{Pitch}}{P_{\text{pitch}}}, \quad D_{\text{throttle}} = \frac{\Delta\text{Throttle}}{P_{\text{throttle}}}$$

Duration formula:

$$\text{Duration} = \frac{\Delta\text{Distance}}{\text{Predefined\_Speed}}$$

2.2 UAV Command Queue Implementation

Finally, we send enqueued `rc` commands to the UAV using a Python SDK (specifically DJITelloPy [1]).

```python
# send_rc_control(roll, pitch, throttle, yaw_rate)

# Yaw control
def yaw(Pyaw, Dyaw):
    action_queue.append(send_rc_control(0, 0, 0, Pyaw))
    time_sleep(Dyaw)
    action_queue.append(send_rc_control(0, 0, 0, 0)) # Stop yaw rate

# Pitch control
def pitch(Ppitch, Dpitch):
    action_queue.append(send_rc_control(0, Ppitch, 0, 0))
    time_sleep(Dpitch)
    action_queue.append(send_rc_control(0, 0, 0, 0)) # Stop pitch rate
        / reset pitch

# Throttle control
def throttle(Pthrottle, Dthrottle):
    action_queue.append(send_rc_control(0, 0, Pthrottle, 0))
    time_sleep(Dthrottle)
    action_queue.append(send_rc_control(0, 0, 0, 0)) # Reset throttle
```

Listing 1: Python-like pseudocode for UAV command queue

Annotations

1. (U, V): 2D waypoint
2. (d_{adj}): Adaptive step size
3. $(U_{\text{norm}}, V_{\text{norm}})$: Normalized 2D waypoint
4. (S_x, S_y, S_z): 3D displacement vector
5. (α, β): camera's (horizontal FOV, vertical FOV), where FOV is field-of-view angle
6. $(\Delta\theta, \Delta\text{Pitch}, \Delta\text{Throttle})$: UAV control primitive displacements
7. $(P_{\text{yaw}}, P_{\text{pitch}}, P_{\text{throttle}})$: Pre-defined speed of each primitive
8. $(D_{\text{yaw}}, D_{\text{pitch}}, D_{\text{throttle}})$: Duration of each primitive

3 Experimental Details

3.1 Task Specifications and Evaluation Protocols

Task outcomes are classified as Success or Failure. A trial is a Failure if the UAV collides or if, at the task's completion, the target is not visible within the drone's egocentric camera view. A trial is a Success if, without collision, it has completed the specific task as the prompt requested (e.g., fly through the building), or if the target is clearly visible in the final egocentric view and the drone is positioned within 1 meter (real-world) or 1-5 meters (simulator) of the target. These criteria are consistent with common evaluation practices in the AVLN benchmarks (e.g. [2, 3, 4, 5]).

A comprehensive table of examples of prompts (Table 1) is provided to illustrate the detailed instructions for simulated and real-world tasks.

3.2 Implementation Details

Our system uses an adaptive scaling mechanism (detailed in the main paper, Section 3.2) with parameters $s = 10$, $L = 10$, $d_{\text{min}} = 0.1$m and $p = 1.8$. The control architecture operates asynchronously with VLM inference at $\approx 0.3 \sim 1$ Hz and low-level commands at ≈ 10 Hz, resulting in an end-to-end latency of $\approx 1.5 \sim 3$ seconds, primarily due to VLM inference time. Unless otherwise specified, all experiments use Gemini 2.0 Flash [6] as the VLM backend.

3.3 Qualitative Videos

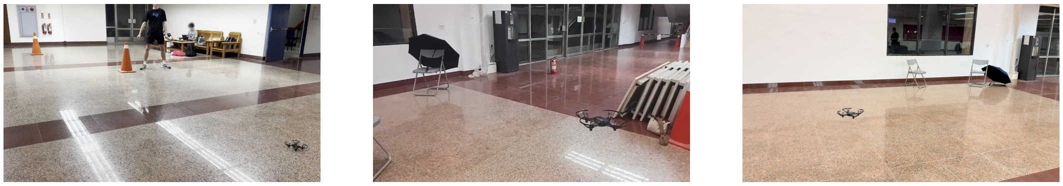
The qualitative results of our experiments are provided as demonstration videos in the `exp_results` directory of the supplementary materials. For convenient viewing, a video viewer webpage is included in the same folder. By opening `index.html` in any modern web browser, the videos corresponding to the various tasks and scenarios discussed in the paper can be easily browsed and viewed. This interface is intended to facilitate the evaluation of our approach and visually support the findings presented.

3.4 Experiment Setup of Adaptive Travel Distance Scaling

To assess our adaptive step-size mechanism (Main Paper, Sec. 3.2), we conducted a real-world ablation study comparing our Adaptive Step-Size Controller against a fixed baseline. The experiments utilized a DJI Tello EDU drone, controlled via DJITelloPy [1] using low-level `rc` velocity commands. Three distinct tasks, designed to test long-horizon planning and reasoning (detailed in Table 2 and Figure 1), were each executed 5 times per controller configuration for robust comparison.

Table 2: Fixed vs. Adaptive Step-Size Controller performance on three real-world tasks. Metrics are Success Rate (SR) and Completion Time (Compl. time: start to finish). The adaptive controller significantly reduces completion times while maintaining or improving SR against the fixed baseline.

Prompt	Controller type	Compl. time	SR (%)
Fly to the cones and the next.	Fixed	61.00s	100
	Adaptive	**28.00s**	**100**
I'm thirsty. Find something that can help me.	Fixed	50.25s	80
	Adaptive	**35.20s**	**100**
It's raining. Head to the comfiest chair that will keep you dry.	Fixed	47.00s	100
	Adaptive	**30.00s**	**100**



Long Horizon | Reasoning I | Reasoning II

Figure 1: Visual examples of the real-world scenarios for the tasks (from left to right): Long Horizon ("Fly to the cones and the next."), Reasoning I ("I'm thirsty. Find something that can help me."), and Reasoning II ("It's raining. Head to the comfiest chair that will keep you dry."). These images depict the types of environments and objectives the UAV encountered during the ablation study evaluating the adaptive step-size controller.

The results in Table 2 show that the adaptive controller significantly reduces task completion times while maintaining or improving success rates (SR). For instance, in the task "Fly to the cones and the next." the completion time was more than halved (61s to 28s) with 100% SR. For "I'm thirsty. Find something that can help me.", the adaptive controller decreased the completion time (50.25s to 35.20s) and improved the SR from 80% to 100%. The "It's raining..." task also saw a substantial time reduction (47s to 30s) with 100% SR. These findings confirm the efficacy of the adaptive mechanism in enhancing operational efficiency and reliability in complex real-world settings.

References

[1] D. Fuentes Escoté, J. Löw, and The DJITelloPy Contributors. DJITelloPy: DJI Tello Python Interface, 2025. URL `https://github.com/damiafuentes/DJITelloPy`.

[2] S. Liu, H. Zhang, Y. Qi, P. Wang, Y. Zhang, and Q. Wu. Aerialvln: Vision-and-language navigation for uavs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15384–15394, 2023.

[3] J. Lee, T. Miyanishi, S. Kurita, K. Sakamoto, D. Azuma, Y. Matsuo, and N. Inoue. Citynav: Language-goal aerial navigation dataset with geographic information. *arXiv preprint arXiv:2406.14240*, 2024.

[4] X. Wang, D. Yang, Z. Wang, H. Kwan, J. Chen, W. Wu, H. Li, Y. Liao, and S. Liu. Towards realistic uav vision-language navigation: Platform, benchmark, and methodology. *arXiv preprint arXiv:2410.07087*, 2024.

[5] Y. Gao, C. Li, Z. You, J. Liu, Z. Li, P. Chen, Q. Chen, Z. Tang, L. Wang, P. Yang, et al. Openfly: A versatile toolchain and large-scale benchmark for aerial vision-language navigation. *arXiv preprint arXiv:2502.18041*, 2025.

[6] Google DeepMind. Gemini, 2025. URL `https://deepmind.google/technologies/gemini/`.

Table 1: Numbered Example Prompts Used Across Task Categories. Each prompt is individually numbered and grouped by environment and category.

Environment	Category	Prompt
Simulation	Navigation	1. Take off and fly to the red crane 2. Take off and fly to the tall white building 3. Take off and fly to the white needle 4. Take off and fly to the black car 5. Fly through the tunnel in front of you
	Obstacle Avoidance	1. Take off and fly to the white needle (with obstacles) 2. Take off and fly to the black car (avoiding obstacles) 3. Fly through the tunnel in front of you 4. Take off and navigate fly through the hollow building without hitting it 5. Navigate through complex bridge structure
	Long Horizon	1. Fly through the first gate and the second 2. Fly to the first tower and then fly to the second tower 3. Go around the tree first and then fly up the hill 4. Look around the plane in front you and then fly to the crane 5. Fly over the building in front of you and search the environment behind it
	Reasoning	1. Take off and scan this city area 2. Fly to an object that can be drive by people 3. Fly to the train cart after the locomotion
	Search / Follow	1. Take off and search for the monorail train 2. Search for the red balloon and fly through each other 3. Take off and search for the train in sight if not look around and find it 4. Take off and search for the tower 5. Take off and search for the lake if you cannot find it in sight look around and search for it
Real-world	Navigation	1. Fly to the chair (long distance)
	Obstacle Avoidance	1. Fly to the person without hitting the cone 2. Fly to the person without hitting the door
	Long Horizon	1. Fly to the chairs and the next 2. Fly to the cone and the next
	Reasoning	1. It's raining, head to the comfiest chair that looks like it'll keep you dry! 2. Fly to the person who needs help 3. I'm thirsty, find something that can help me. 4. Fly to the person in the dark area
	Search / Follow	1. Fly toward the body of the person with red cone 2. Fly toward the person with green shirt