UAV Flight Log Generation and High-Level Mission Planning Using Vision-Language Models

TsungYen Yu
National Chengchi University

Abstract

This study aims to establish a staged pipeline that integrates vision-language models (VLMs) with UAV control, allowing drones to simultaneously perform image interpretation while executing missions. Specifically, the drone continuously analyzes real-time images captured by its camera, evaluates how to execute high-level abstract commands issued by the user (e.g., “find an old temple”), and determines whether the objective is successfully achieved. At the same time, the UAV system records flight parameters, key onsite events, and important landmarks in a detailed log format similar to a nautical logbook, then returns structured information to the operator to provide comprehensive feedback and decision support.

In the experiments, AirSim (Unreal Engine + ROS2) is used as a UAV simulation environment to reduce experimental costs and shorten the development cycle. Because AirSim uses the same control signals as real-world drones and Unreal Engine can simulate realistic environments, if the latency of VLM data transmission can be reduced, this technology has direct potential for deployment on real drones, pushing forward the future of UAV automation and intelligent applications.

1. Introduction

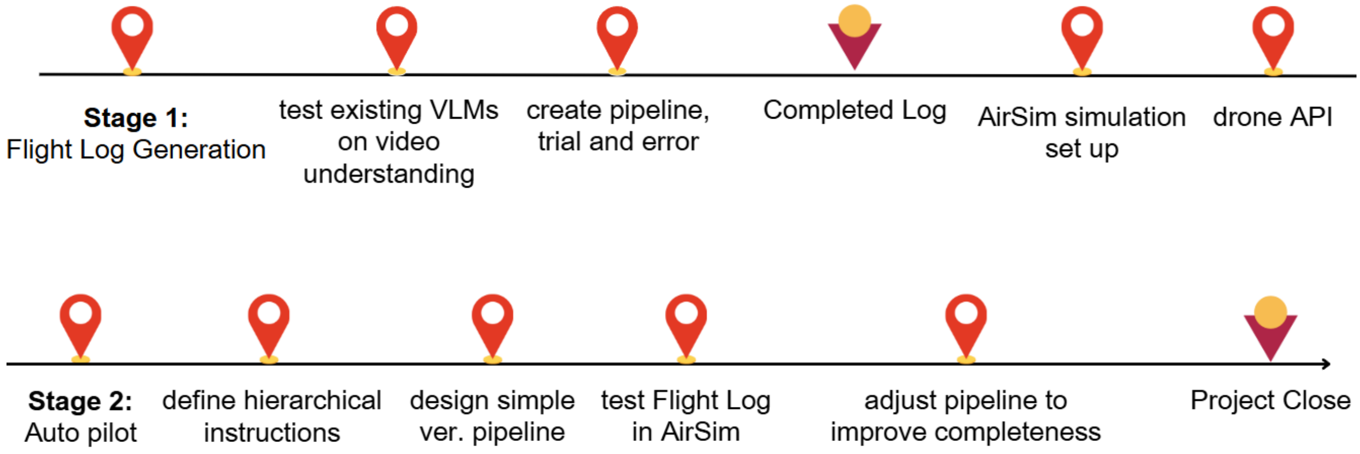
With the rapid rise of artificial intelligence in computer vision, powerful architectures such as Convolutional Neural Networks and the more recent Vision Transformers are reshaping the landscape of autonomous systems. For UAVs, this means evolving from rigid GPS-dependent routes to true situational aware autonomous flight. Meanwhile, the field of natural language processing (NLP) has also witnessed its own advancement with the emergence of Large Language Models (LLMs), unlocking machine capabilities for complex human-language reasoning.

These two once-independent areas were eventually bridged by the birth of the CLIP model, opening the era of vision-language models (VLMs). This has provided a novel interaction interface, allowing users without prior mechanical control experience to easily communicate with drones using natural language and enabling the drones to execute tasks autonomously.

Currently, many emerging technologies are targeting applications in VLM + Robotics, such as Vision-Language-Action (VLA) models. These models fine-tune pre-trained VLMs to enable closed-loop control of machinery, directly outputting control commands. This study focuses on leveraging VLMs’ high-level reasoning and versatile capabilities to propose a system framework that integrates these domains. Within AirSim, it achieves natural human-drone collaboration and generates a structured flight log using the language capabilities of VLMs for efficient post-flight analysis.

2. Methodology

The figure below provides an overview of the development process for the overall study:

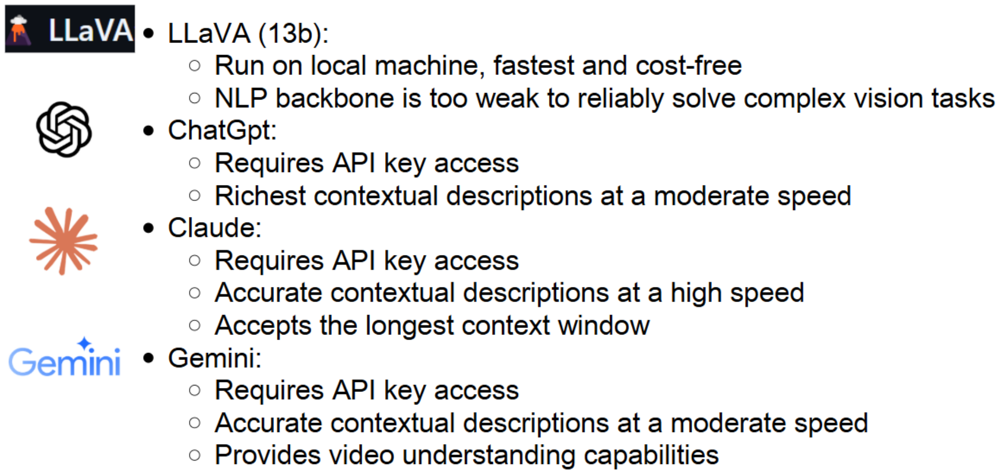

Stage 1:
Flight Log Generation
test existing VLMs on video understanding
create pipeline, trial and error
Completed Log
AirSim simulation set up
drone API
Stage 2:
Auto pilot
define hierarchical instructions
design simple ver. pipeline
test Flight Log in AirSim
adjust pipeline to improve completeness
Project Close

2-1. Flight Log Generation

The first part of this study aims to build an effective pipeline to process any given UAV aerial footage and convert it into a concrete, structured flight log for post-flight evaluation. The steps are:

A. Selecting and Testing VLMs

To evaluate how well major VLMs adapt to top-down aerial images, their spatial understanding, temporal reasoning, and semantic feedback abilities, I input 10 consecutive aerial frames captured every 3 seconds and requested detailed summaries. I tested four models: LLaVA (13b), ChatGPT 5, Claude-sonnet 4.5, and Gemini 2.5:

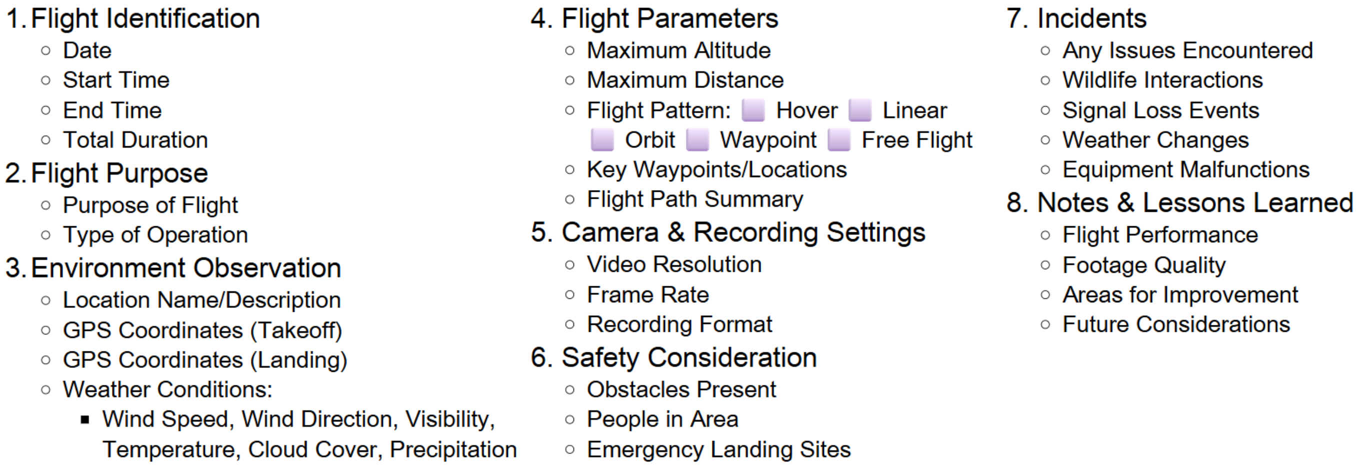


- LLaVA (13b):
  - Run on local machine, fastest and cost-free
  - NLP backbone is too weak to reliably solve complex vision tasks
- ChatGpt:
  - Requires API key access
  - Richest contextual descriptions at a moderate speed
- Claude:
  - Requires API key access
  - Accurate contextual descriptions at a high speed
  - Accepts the longest context window
- Gemini:
  - Requires API key access
  - Accurate contextual descriptions at a moderate speed
  - Provides video understanding capabilities

Therefore, ChatGPT 5 was chosen as the "eyes," Claude-sonnet 4.5 as the "brain," and Gemini 2.5 as an independent agent using its native video-understanding capabilities.

B. Designing the Flight Log Format Based on VLM Capabilities

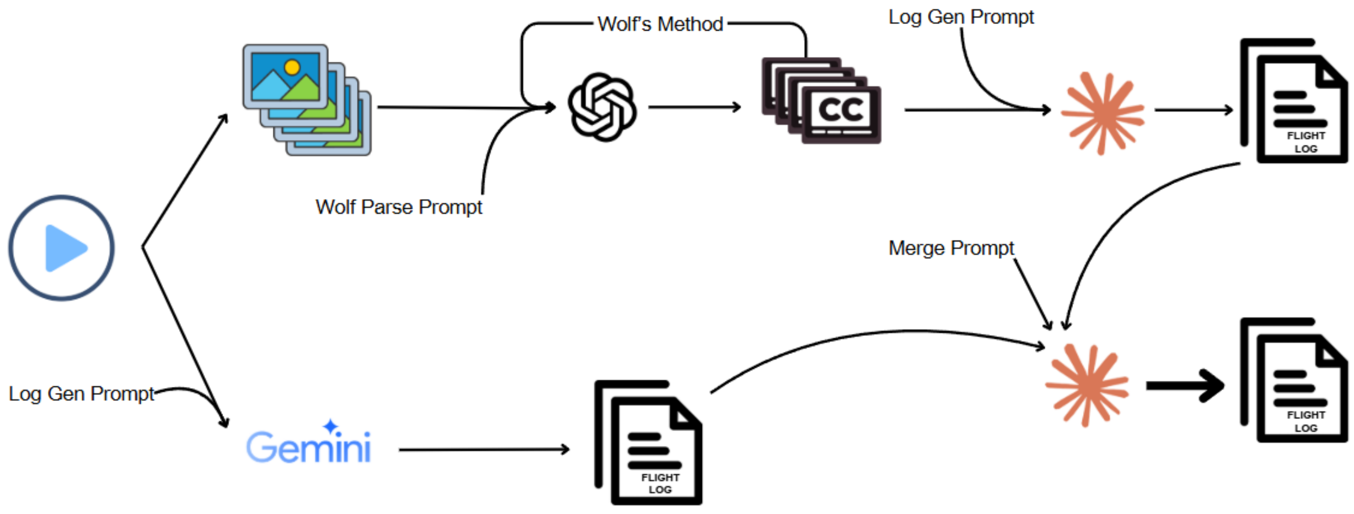
The figure below shows the final selected flight-log template:



1. Flight Identification
   - Date
   - Start Time
   - End Time
   - Total Duration
2. Flight Purpose
   - Purpose of Flight
   - Type of Operation
3. Environment Observation
   - Location Name/Description
   - GPS Coordinates (Takeoff)
   - GPS Coordinates (Landing)
   - Weather Conditions:
     - Wind Speed, Wind Direction, Visibility, Temperature, Cloud Cover, Precipitation
4. Flight Parameters
   - Maximum Altitude
   - Maximum Distance
   - Flight Pattern: □ Hover □ Linear □ Orbit □ Waypoint □ Free Flight
   - Key Waypoints/Locations
   - Flight Path Summary
5. Camera & Recording Settings
   - Video Resolution
   - Frame Rate
   - Recording Format
6. Safety Consideration
   - Obstacles Present
   - People in Area
   - Emergency Landing Sites
7. Incidents
   - Any Issues Encountered
   - Wildlife Interactions
   - Signal Loss Events
   - Weather Changes
   - Equipment Malfunctions
8. Notes & Lessons Learned
   - Flight Performance
   - Footage Quality
   - Areas for Improvement
   - Future Considerations

C. Designing the Pipeline to Convert Aerial Footage into a Flight Log

The pipeline consists of two parts. The first utilizes a frame-by-frame approach with a sampling rate of 0.33 FPS. In this stage, key frames are converted into textual captions, which are then combined into a final flight log. This uses the approach proposed by Wolf [1] wherein the caption derived from the previous key frame is fed into the VLM alongside the current input key frame to maintain temporally consistent reasoning. The second part is more direct, leveraging the video understanding capabilities of Gemini 2.5 to directly convert the aerial footage into a log. By employing these two completely different approaches, we aim to have them complement each other and mitigate their respective limitations. The actual pipeline is illustrated below:



2-2. Autonomous Flight

The second part of this study aims to achieve autonomous UAV flight in which users issue high-level abstract commands, and the drone can automatically fly in AirSim. The steps are:

A. Building the AirSim Environment and Designing UAV Flight APIs
To simplify VLM control, UAV actions are limited to three types:

1. Fly forward x
2. Rotate in place θ
3. Vertical movement $\pm x$

B. Classifying User Commands into High-, Mid-, and Low-Level Instructions

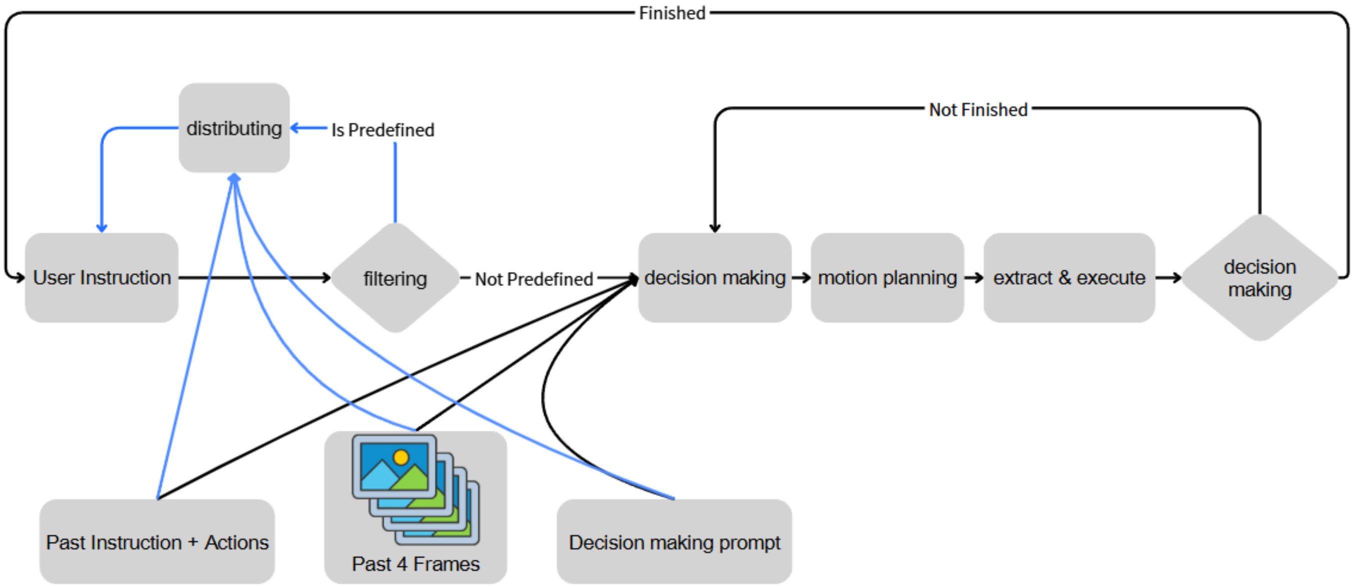
	Name	Description	Example
High Level	(Vision) Task	Ambiguous, high-level goals without specific "how-to" instructions.	"Find me an old temple", "Fly until you see a roundabout"
High Level	Predefined Mission	Complicated but commonly used. Hard for VLMs to handle	"Return Flight", "Areal Scan"
Middle Level	Navigation	Gives a specific idea of "how" to fly	"Fly ahead for 100m and turn left", "fly in a square that has a length of 100m"
Low Level	Action	The most specific commands (APIs) that directly control the system	move_forward(x), rotate(x)

As shown in the figure above, higher-level commands are more abstract and lack instructions on how to fly.

C. Designing a Scalable and Stable Pipeline for Autonomous Flight
The goal is to convert all high- and mid-level commands into low-level commands for execution in AirSim. After flying, the system will output:

1. A 20-fps UAV aerial video
2. All user Instructions and their corresponding translated low-level actions

The actual pipeline is illustrated below:



First, a filter judges whether the user Instruction is a Predefined Mission. If so, it forwards the Instruction to that mission's dedicated sub-pipeline. Otherwise, it goes through two stages of decision-making. The first stage works with motion planning to determine "how" to fly, while the second stage evaluates "whether" the user's commands have been achieved.

D. Designing Sub-Pipelines for Predefined Missions

In this study, three Predefined Missions are implemented (but it's easy to extend):

1. Return Flight
2. Areal Scan
3. Static-object (Path) Following

Details are discussed in the appendix (supplementary slide P.33) for simplicity.

3. Experiments and Results

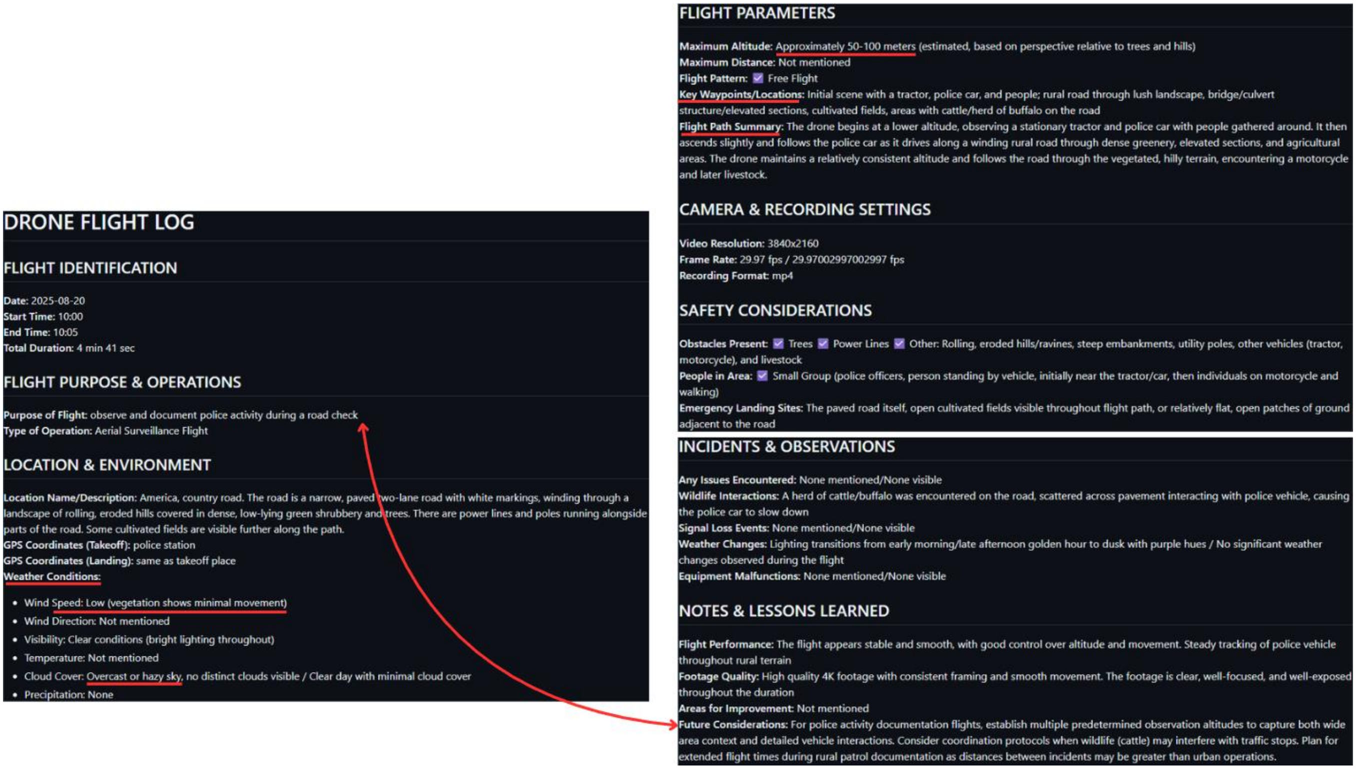
Since the outputs are long flight logs and videos, only brief examples are shown here. Full results are available on my GitHub. The demonstrations include:

1. Flight-log generation from real aerial footage
2. Mid-level Instruction: Navigations
3. High-level Instruction: Vision Tasks
4. High-level Instruction: Predefined Missions

Demo 1: Real-world Aerial Footage – Flight Log (Following a Police Car)

Below are manually selected representative frames:

The resulting flight log is shown as the following:



DRONE FLIGHT LOG

FLIGHT IDENTIFICATION

Date: 2025-08-20
Start Time: 10:00
End Time: 10:05
Total Duration: 4 min 41 sec

FLIGHT PURPOSE & OPERATIONS

Purpose of Flight: observe and document police activity during a road check
Type of Operation: Aerial Surveillance Flight

LOCATION & ENVIRONMENT

Location Name/Description: America, country road. The road is a narrow, paved two-lane road with white markings, winding through a landscape of rolling, eroded hills covered in dense, low-lying green shrubbery and trees. There are power lines and poles running alongside parts of the road. Some cultivated fields are visible further along the path.
GPS Coordinates (Takeoff): police station
GPS Coordinates (Landing): same as takeoff place
Weather Conditions:

- Wind Speed: Low (vegetation shows minimal movement)
- Wind Direction: Not mentioned
- Visibility: Clear conditions (bright lighting throughout)
- Temperature: Not mentioned
- Cloud Cover: Overcast or hazy sky, no distinct clouds visible / Clear day with minimal cloud cover
- Precipitation: None

FLIGHT PARAMETERS

Maximum Altitude: Approximately 50-100 meters (estimated, based on perspective relative to trees and hills)
Maximum Distance: Not mentioned
Flight Pattern: ☑ Free Flight
Key Waypoints/Locations: Initial scene with a tractor, police car, and people; rural road through lush landscape, bridge/culvert structure/elevated sections, cultivated fields, areas with cattle/herd of buffalo on the road
Flight Path Summary: The drone begins at a lower altitude, observing a stationary tractor and police car with people gathered around. It then ascends slightly and follows the police car as it drives along a winding rural road through dense greenery, elevated sections, and agricultural areas. The drone maintains a relatively consistent altitude and follows the road through the vegetated, hilly terrain, encountering a motorcycle and later livestock.

CAMERA & RECORDING SETTINGS

Video Resolution: 3840x2160
Frame Rate: 29.97 fps / 29.97002997002997 fps
Recording Format: mp4

SAFETY CONSIDERATIONS

Obstacles Present: ☑ Trees ☑ Power Lines ☑ Other: Rolling, eroded hills/ravines, steep embankments, utility poles, other vehicles (tractor, motorcycle), and livestock
People in Area: ☑ Small Group (police officers, person standing by vehicle, initially near the tractor/car, then individuals on motorcycle and walking)
Emergency Landing Sites: The paved road itself, open cultivated fields visible throughout flight path, or relatively flat, open patches of ground adjacent to the road

INCIDENTS & OBSERVATIONS

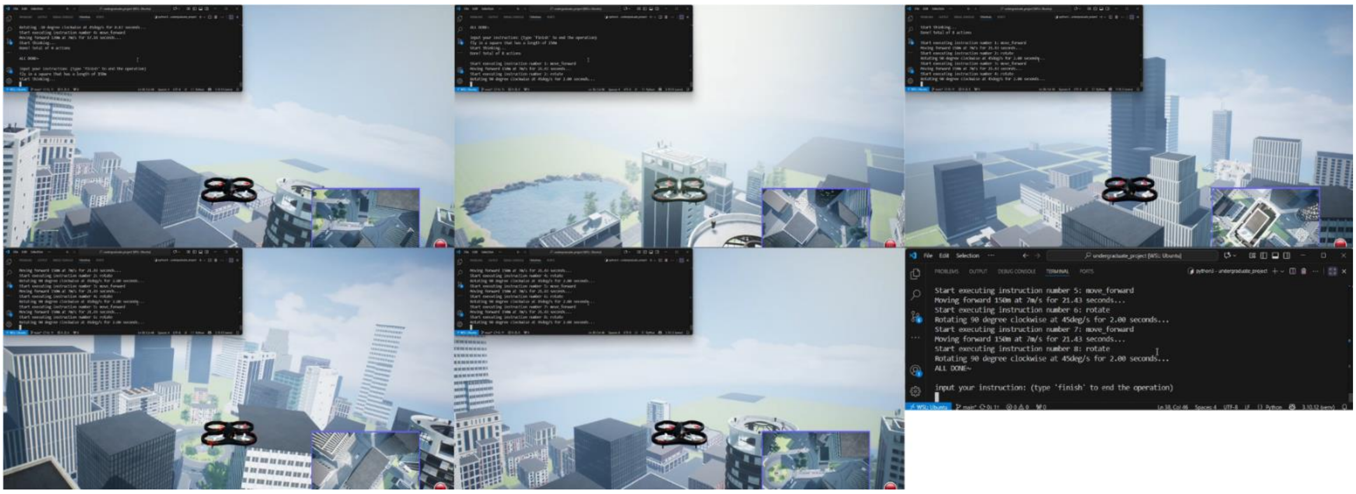
Any Issues Encountered: None mentioned/None visible
Wildlife Interactions: A herd of cattle/buffalo was encountered on the road, scattered across pavement interacting with police vehicle, causing the police car to slow down
Signal Loss Events: None mentioned/None visible
Weather Changes: Lighting transitions from early morning/late afternoon golden hour to dusk with purple hues / No significant weather changes observed during the flight
Equipment Malfunctions: None mentioned/None visible

NOTES & LESSONS LEARNED

Flight Performance: The flight appears stable and smooth, with good control over altitude and movement. Steady tracking of police vehicle throughout rural terrain
Footage Quality: High quality 4K footage with consistent framing and smooth movement. The footage is clear, well-focused, and well-exposed throughout the duration
Areas for Improvement: Not mentioned
Future Considerations: For police activity documentation flights, establish multiple predetermined observation altitudes to capture both wide area context and detailed vehicle interactions. Consider coordination protocols when wildlife (cattle) may interfere with traffic stops. Plan for extended flight times during rural patrol documentation as distances between incidents may be greater than urban operations.

Although the lawn mower was not mentioned, the content is generally correct and can even roughly infer altitude.

Demo 2: Mid-Level Instruction (Navigation) – "Fly in a square of length 150 m"

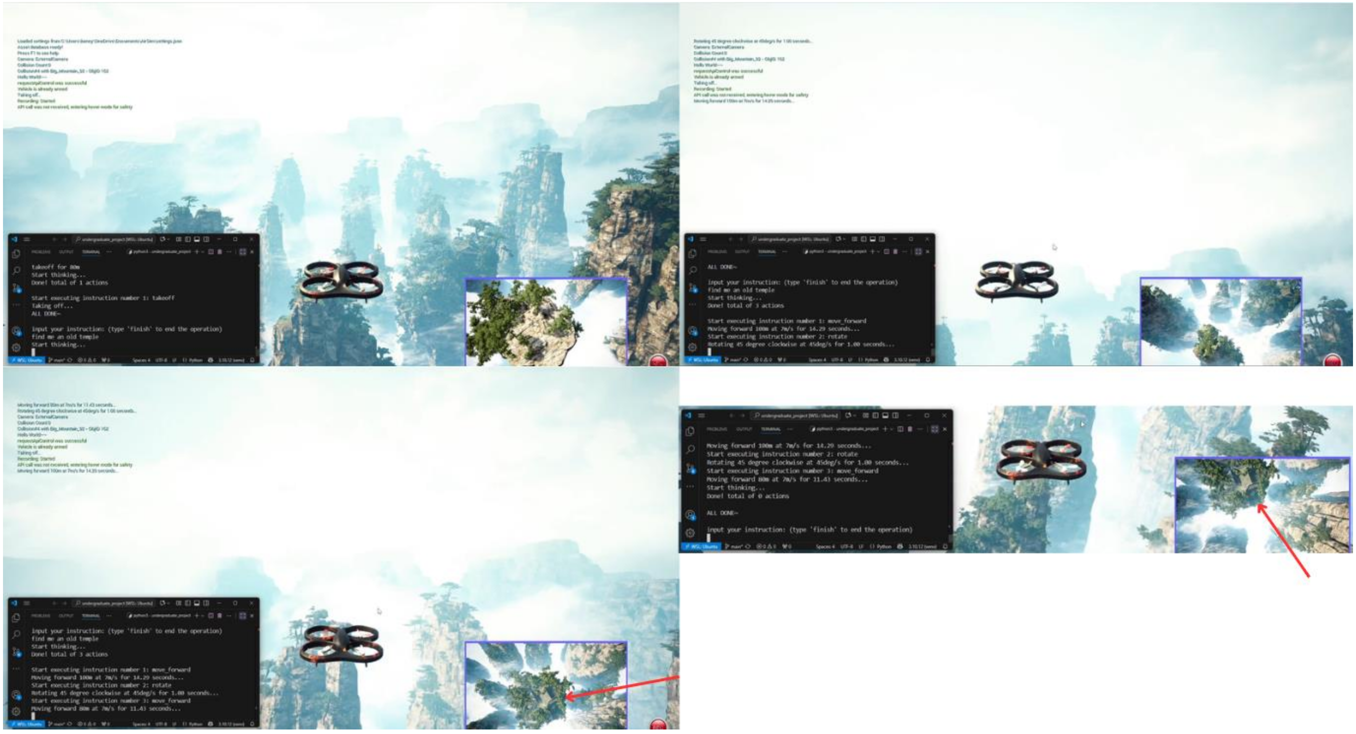


```
Start executing instruction number 5: move_forward
Moving forward 150m at 7m/s for 21.43 seconds...
Start executing instruction number 6: rotate
Rotating 90 degree clockwise at 45deg/s for 2.00 seconds...
Start executing instruction number 7: move_forward
Moving forward 150m at 7m/s for 21.43 seconds...
Start executing instruction number 8: rotate
Rotating 90 degree clockwise at 45deg/s for 2.00 seconds...
ALL DONE~

input your instruction: (type 'finish' to end the operation)
```

In the figure, the upper-left section displays the terminal used to issue commands to the drone, while the lower-right presents the actual aerial footage captured by the drone's onboard camera. It can be observed that the UAV successfully executed a square trajectory and returned to its initial position.

Demo 3: High-Level Instruction (Vision Task) – “Find me an old temple”



```
Loaded settings from C:\Users\[illegible]\OneDrive\Documents\AirSim\settings.json
Asset database ready!
Press F1 to see help
Camera: ExternalCamera
Collision Count:0
CollisionAt with Big_Mountain_32 - ObjID 152
Hello World~~
requestApiControl was successful
Vehicle is already armed
Taking off...
Recording: Started
API call was not received, entering hover mode for safety
```

```
takeoff for 80m
Start thinking...
Done! total of 1 actions

Start executing instruction number 1: takeoff
Taking off...
ALL DONE~

input your instruction: (type 'finish' to end the operation)
find me an old temple
Start thinking...
```

```
Rotating 45 degree clockwise at 45deg/s for 1.00 seconds...
Camera: ExternalCamera
Collision Count:0
CollisionAt with Big_Mountain_32 - ObjID 152
Hello World~~
requestApiControl was successful
Vehicle is already armed
Taking off...
Recording: Started
API call was not received, entering hover mode for safety
Moving forward 100m at 7m/s for 14.29 seconds...
```

```
ALL DONE~

input your instruction: (type 'finish' to end the operation)
find me an old temple
Start thinking...
Done! total of 3 actions

Start executing instruction number 1: move_forward
Moving forward 100m at 7m/s for 14.29 seconds...
Start executing instruction number 2: rotate
Rotating 45 degree clockwise at 45deg/s for 1.00 seconds...
```

```
Moving forward 80m at 7m/s for 11.43 seconds...
Rotating 45 degree clockwise at 45deg/s for 1.00 seconds...
Camera: ExternalCamera
Collision Count:0
CollisionAt with Big_Mountain_32 - ObjID 152
Hello World~~
requestApiControl was successful
Vehicle is already armed
Taking off...
Recording: Started
API call was not received, entering hover mode for safety
Moving forward 100m at 7m/s for 14.29 seconds...
```

```
input your instruction: (type 'finish' to end the operation)
find me an old temple
Start thinking...
Done! total of 3 actions

Start executing instruction number 1: move_forward
Moving forward 100m at 7m/s for 14.29 seconds...
Start executing instruction number 2: rotate
Rotating 45 degree clockwise at 45deg/s for 1.00 seconds...
Start executing instruction number 3: move_forward
Moving forward 80m at 7m/s for 11.43 seconds...
```

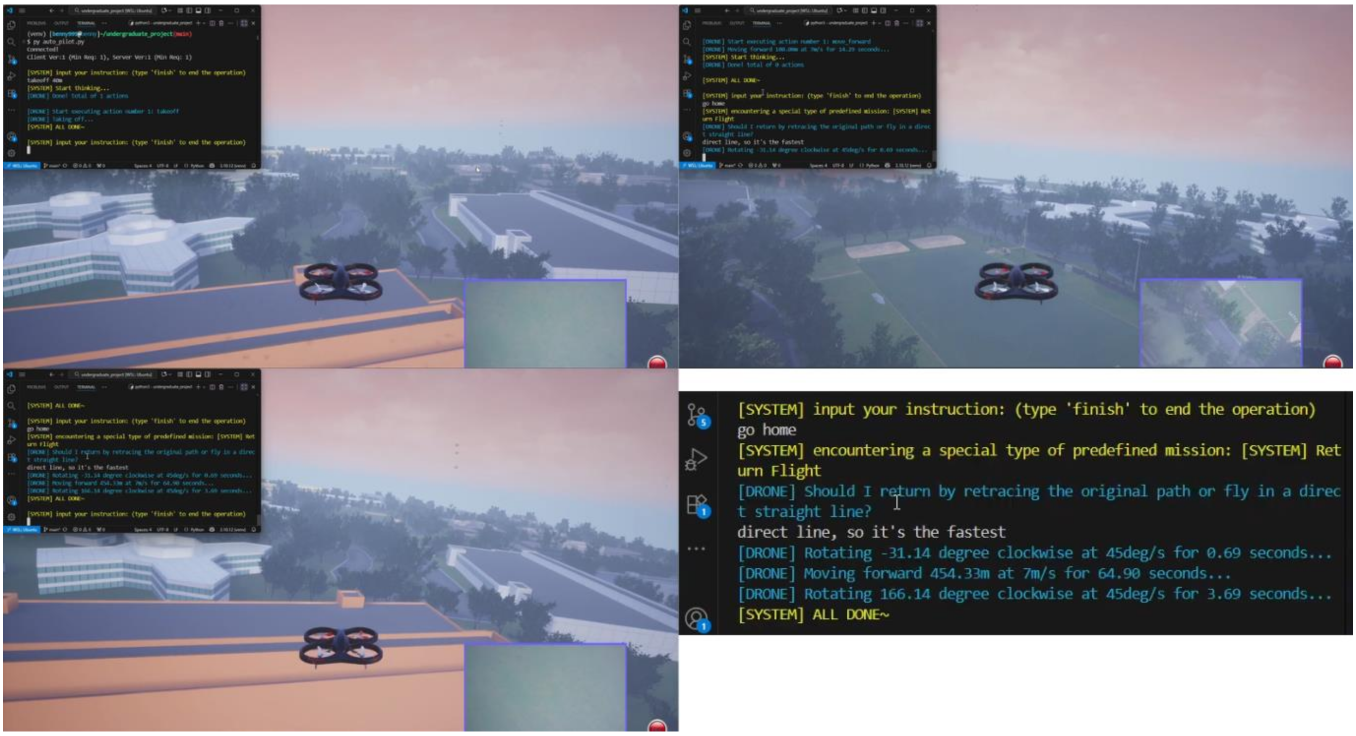
```
Moving forward 100m at 7m/s for 14.29 seconds...
Start executing instruction number 2: rotate
Rotating 45 degree clockwise at 45deg/s for 1.00 seconds...
Start executing instruction number 3: move_forward
Moving forward 80m at 7m/s for 11.43 seconds...
Start thinking...
Done! total of 0 actions

ALL DONE~

input your instruction: (type 'finish' to end the operation)
```

The drone stopped upon finding the temple.

Demo 4: High-Level Instruction (Predefined Mission) – “Go home”



```
(venv) [benny999@benny]~/undergraduate_project(main)
$ py auto_pilot.py
Connected!
Client Ver:1 (Min Req: 1), Server Ver:1 (Min Req: 1)

[SYSTEM] input your instruction: (type 'finish' to end the operation)
takeoff 40m
[SYSTEM] Start thinking...
[DRONE] Done! total of 1 actions

[DRONE] Start executing action number 1: takeoff
[DRONE] Taking off...
[SYSTEM] ALL DONE~

[SYSTEM] input your instruction: (type 'finish' to end the operation)
```

```
[DRONE] Start executing action number 1: move_forward
[DRONE] Moving forward 100.00m at 7m/s for 14.29 seconds...
[SYSTEM] Start thinking...
[DRONE] Done! total of 0 actions

[SYSTEM] ALL DONE~

[SYSTEM] input your instruction: (type 'finish' to end the operation)
go home
[SYSTEM] encountering a special type of predefined mission: [SYSTEM] Ret
urn Flight
[DRONE] Should I return by retracing the original path or fly in a direc
t straight line?
direct line, so it's the fastest
[DRONE] Rotating -31.14 degree clockwise at 45deg/s for 0.69 seconds...
```

```
[SYSTEM] ALL DONE~

[SYSTEM] input your instruction: (type 'finish' to end the operation)
go home
[SYSTEM] encountering a special type of predefined mission: [SYSTEM] Ret
urn Flight
[DRONE] Should I return by retracing the original path or fly in a direc
t straight line?
direct line, so it's the fastest
[DRONE] Rotating -31.14 degree clockwise at 45deg/s for 0.69 seconds...
[DRONE] Moving forward 454.33m at 7m/s for 64.90 seconds...
[DRONE] Rotating 166.14 degree clockwise at 45deg/s for 3.69 seconds...
[SYSTEM] ALL DONE~

[SYSTEM] input your instruction: (type 'finish' to end the operation)
```

```
[SYSTEM] input your instruction: (type 'finish' to end the operation)
go home
[SYSTEM] encountering a special type of predefined mission: [SYSTEM] Ret
urn Flight
[DRONE] Should I return by retracing the original path or fly in a direc
t straight line?
direct line, so it's the fastest
[DRONE] Rotating -31.14 degree clockwise at 45deg/s for 0.69 seconds...
[DRONE] Moving forward 454.33m at 7m/s for 64.90 seconds...
[DRONE] Rotating 166.14 degree clockwise at 45deg/s for 3.69 seconds...
[SYSTEM] ALL DONE~
```

The upper left one is when the takeoff took place, the command “go home” was issued at the upper right’s time, and the drone successfully returned to the origin (lower left).

4. Conclusion

This study demonstrates that vision-language models establish a new bidirectional interface connecting humans with unmanned aerial vehicles, effectively bridging the gap between visual perception and language interaction. Our framework validates the advantages and potential of VLMs in robotic control through two core capabilities:

1. **Flight log generation**, providing a simple, low-effort approach for evaluating UAV footage to help users quickly assess flight status.
2. **Autonomous Flight**, transforming user intent into actionable instructions. Crucially, this process features a highly flexible architecture, allowing for the easy expansion of additional sub-methods without altering the core framework. Furthermore, as AirSim integrates Unreal Engine and ROS2, both the simulation environment and UAV input signals achieve high fidelity. Consequently, once hardware requirements are met, this architecture can be directly transitioned to real-world applications.

However, current VLMs are still limited in visual reasoning. One predefined mission, static-object following, was affected by this constraint. When determining the necessary angular corrections to continuously follow static features such as rivers or lanes—tasks that are intuitive for the human eye—the system frequently deviated from the correct trajectory, eventually losing the target. Although current VLMs can clearly identify the presence of these static objects, there remains significant room for improvement regarding fine-grained control, such as interpreting the extension angles and precise directionality of a lane.

Reference

[1] Boyi Li et al. (2024) Wolf: Dense Video Captioning with a World Summarization Framework