**📄 Final Project Report: Drone Security Analyst System**

***(With VLM + LangChain Agent for Scene Monitoring)***

**🔹 Overview & Problem Understanding**

**The growing need for real-time surveillance, especially in autonomous systems like drones, demands smart, flexible, and interactive software pipelines. This project simulates a drone security analyst system that uses a mobile device as a pseudo-drone. It captures live video and telemetry data, detects objects, generates descriptive event summaries, indexes all frames, and allows natural-language querying through an LLM-powered agent.**

**The pipeline is designed to be modular, adaptable, and hardware-light, suitable for both development and testing on non-drone platforms. The challenge was to build a meaningful multimodal data processing pipeline and LLM-driven interface that provides semantic access to security events—all under the constraints of local hardware and real-time input simulation.**

**🔹 Assumptions Made**

**To simulate a production drone pipeline without physical drone hardware, we assumed:**

* **Mobile-as-Drone: A real Android phone acted as the pseudo-drone, streaming live video via IP Webcam and live telemetry via Phyphox.**
* **Real-Time Input: Instead of using pre-recorded datasets, live sensor + camera streams were used to mimic the drone environment and ensure realism.**
* **Scene Analysis: Object detection per frame is sufficient to extract visual meaning; hence we used per-frame analysis over video-level processing.**
* **Telemetry Syncing: Matching frame timestamps with telemetry records within ~100ms was considered “synchronized” for indexing purposes.**

**🔹 Justification of Tool & Model Choices**

**🔸 Dual Detection Model: YOLOv8 + CLIP**

**We opted for a dual-mode object detection pipeline:**

* **YOLOv8 was chosen for high-speed, high-accuracy real-time detection with bounding boxes.**
* **CLIP provided lightweight and semantically aware object identification, acting as a fallback in low-resource contexts.**

**This dual-mode gives flexibility to balance speed vs semantic expressiveness based on hardware availability.**

**🔸 Language Model Choice: ChatGPT-2 (Fallback from Phi-2)**

**Initially, we planned to use Phi-2 due to its balance of quality and lightweight nature. However, due to incomplete or corrupted downloads during local setup, Phi-2 could not be used in time.**

**We reverted to using ChatGPT-2, a smaller, well-tested transformer-based model that can run locally. Though less refined than Phi-2, it provided sufficiently fluent descriptions and maintained frame-to-description performance within acceptable latency.**

**Note: This fallback was purely due to infrastructure limitations, not performance issues with Phi-2. All prompt logic and integration had already been set up to support either model.**

**🔸 LangChain Agent**

**LangChain was used to build a conversational agent that reads structured .jsonl log files, extracts relevant records, and responds in natural language. We chose LangChain because:**

* **It integrates smoothly with LLMs**
* **It supports conversational memory, enabling follow-up questions (e.g., “What happened next?”)**
* **It’s extensible, lightweight, and transparent**

**No vector database was used; instead, we relied on JSONL parsing and prompt templates to answer queries efficiently and clearly.**

**🔹 Results & Examples**

**🔸 Frame Detection**

**The system captures and processes each frame using YOLOv8 or CLIP. Here’s an example of a logged output:**

**{**

**"frame\_id": 108,**

**"timestamp": 1713689745.12,**

**"lat": 12.930183,**

**"lon": 77.621110,**

**"alt": 912.9,**

**"heading": 274,**

**"description": "A person is standing near a white car in front of a building."**

**}**

**Descriptions were generated using ChatGPT-2 via a structured object list (e.g., ['person', 'car', 'building']), passed through a one-sentence prompt pattern.**

**🔸 LangChain Q&A**

**Queries like:**

**“Show me all events involving a refrigerator”**

**Would return responses like:**

**1. At 12:14 PM, a refrigerator was detected near the hallway. Frame: 113, Video: session\_003.mp4**

**2. At 12:17 PM, a person placed an item on top of the refrigerator near the kitchen entrance. Frame: 125, Video: session\_003.mp4**

**These were compiled from indexed logs, matched to object descriptions, and answered via ChatGPT-2, invoked through LangChain.**

**🔸 Alert Simulation**

**Alerts were conditionally triggered based on key object-location combos. Example:**

**"truck" + "gate" → triggers:**

**ALERT: Unauthorized vehicle spotted at 12:46 PM near entry gate. Session: session\_005.mp4**

**🔸 Emergency Stop System**

**Two types of failures were monitored:**

* **Camera disconnect → Automatically halts the pipeline and prints:  
  "Camera lost. Drone returning to dock."**
* **Telemetry stop or malformed data → Also halts the pipeline with:  
  "Telemetry failed. Drone returning to dock."**

**These simulate safe fallback protocols in real drone systems.**

**🔹 What Could Be Improved (Given More Time)**

**If time and compute were not constraints, we would extend this system with:**

* **Video Summarization: Generating natural summaries for an entire session—e.g., “Session\_008 contained 3 trucks, 2 people, and one alert event.”**
* **Phi-2 Completion: With more time and stable internet/hardware, Phi-2 could be successfully installed, offering better descriptions and faster inference than ChatGPT-2.**
* **AirSim Integration: Real drone simulation using AirSim or Gazebo would validate system performance under navigation.**
* **More Intelligent Alerts: Use of pre-trained action recognition or scene analysis models (e.g., SLOWFAST) to detect behavioral cues like loitering, object drop, or running.**
* **Interactive Dashboard: UI to live monitor logs, telemetry heatmaps, search outputs, and flagged alerts.**

**🔹 How AI Tools Helped**

**This project was heavily AI-assisted, especially in terms of developer productivity:**

**✅ ChatGPT:**

* **Helped design the architecture from scratch**
* **Built out the LangChain agent chain and memory handling**
* **Rewrote large blocks of inefficient GPT-2 prompt logic**
* **Helped create debugging flows for telemetry/HTTP problems**
* **Generated skeleton code for JSONL indexing and captioning**

**✅ GitHub Copilot:**

* **Provided inline code suggestions for:**
  + **API fetching**
  + **Data formatting**
  + **Exception handling**
  + **Class structure templates**
* **Made coding faster, especially for threading, socket logic, and prompt chaining**

**✅ CLIP + ChatGPT-2:**

* **CLIP was used to provide object semantics with low overhead**
* **ChatGPT-2 returned descriptive, fluent one-liner captions based on object context**
* **Together they formed a lightweight, interactive detection pipeline that could be queried by the LangChain agent**

**✅ Final Notes**

**Despite time and resource constraints, this project implemented a full, intelligent drone simulation system:**

* **Live video and telemetry ingest**
* **Dual-path detection model**
* **Descriptive captioning using LLMs**
* **JSONL-based indexing with timestamps**
* **Queryable interface using LangChain agent**
* **Fail-safety via crash detection logic**

**While we reverted from Phi-2 to ChatGPT-2 due to environment limitations, the fallback was successful and still met the assignment’s scope. This system is ready for further scaling and integration into real-world drone simulations or deployments.**