■ Code Documentation

■ Project Overview

This project implements a Drone Security Analyst Agent that automates surveillance video analysis through AI. It performs the following core tasks: - Extracts frames from drone video feeds - Captions each frame using the BLIP Vision-Language Model - Detects suspicious activity (e.g., loitering, crash, assault) - Logs events with metadata into a SQLite database - Summarizes video activity using BART - Answers natural language questions using Gemini 1.5 Flash

■ Code Structure

main.py – Frame Processing & Logging

Responsibilities:
- Read input video (fdoor.mp4) - Extract frames using OpenCV - Generate captions using BLIP - Parse captions for objects and alerts - Store frame data in vlm_log.db

■ *Frame Extraction (OpenCV)*

```
cap = cv2.VideoCapture('fdoor.mp4')
frame_rate = 0.5  # Capture every 2 seconds
count = 0

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    if count % int(frame_rate * cap.get(cv2.CAP_PROP_FPS)) == 0:
        frame_path = f"frames/frame_{count}.jpg"
        cv2.imwrite(frame_path, frame)
    count += 1
```

■ *Captioning with BLIP*

```
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image

processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")

inputs = processor(images=Image.open(frame_path), return_tensors="pt")
output = model.generate(**inputs)
caption = processor.decode(output[0], skip_special_tokens=True)
```

■ *Logging to SQLite*

```
import sqlite3

conn = sqlite3.connect('vlm_log.db')
cursor = conn.cursor()
cursor.execute('''
    INSERT INTO caption_logs (frame, timestamp, caption, location, alert)
    VALUES (?, ?, ?, ?, ?)
''', (frame_name, timestamp, caption, location, alert))

conn.commit()
conn.close()
```

### *app.py – Summarization, Search & Q&A;*

Responsibilities:
- Load logs from the SQLite database - Summarize video content using BART - Enable natural language querying via Gemini 1.5 Flash

■ *Summarization with BART*

```
from transformers import pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
summary = summarizer(captions_text, max_length=150, min_length=30, do_sample=False)
print(summary[0]['summary_text'])
```

■ *Natural Language to SQL with Gemini*

```
import google.generativeai as genai

response = model.generate_content(f"Convert the question to SQL: {user_input}")
sql_query = extract_sql(response.text)
```

■ *Executing Gemini SQL Output*

```
cursor.execute(sql_query)
rows = cursor.fetchall()

for row in rows:
    print(row)
```

■■ Database Schema

```
CREATE TABLE caption_logs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    frame TEXT,
    timestamp TEXT,
    caption TEXT,
    location TEXT,
    alert TEXT
);
```

■ Sample Interactions

• Where was the car seen? → ■ Answer: parking • What frames had loitering? → ■ Answer: frame_0012.jpg, frame_0015.jpg • Summarize the video → A motorcycle drove through a tunnel and a man was seen at the gate.

■ Tech Stack

- BLIP – Image captioning from video frames - BART – Summarizing multi-frame captions - Gemini 1.5 – Converts natural questions to SQL - OpenCV – Frame extraction from video - SQLite – Local database for indexed captions - dotenv – API key handling