***Agentic Vision System***



***Submitted by:***

Faizan Rasool (041)

Anoosha Alina (037)

**Section :**

A3

***Submitted To:***

Daniyal Adeeb

### **1. Project Overview**

This project presents a web-based Vision System powered by AI models to perform a range of image and video analysis tasks. Users can upload files and interact with a chatbot to perform operations such as image captioning, object detection, emotion recognition, drone detection, accident detection, video summarization, and real-time object tracking.

The system uses both pretrained and custom-trained models in a modular architecture, integrated with prompt-based AI assistance, to intelligently classify user queries and perform relevant analysis on uploaded media. This allows even non-technical users to access advanced vision capabilities through natural language interaction.

### **2. Key Functionalities**

* **Image Captioning**: Generates a descriptive sentence summarizing the visual content of an image.
* **Scene Description**: Uses object detection and captioning results to compose a detailed and coherent description of the image context.
* **Object Detection**: Detects and highlights multiple objects within an image or video frame using YOLOv8 models.
* **Object Counting**: Returns a dictionary of object types along with their frequencies in the media.
* **Emotion Detection**: Analyzes facial expressions in images or video frames using DeepFace to classify emotions like happy, sad, angry, etc.
* **Drone Detection**: Utilizes a custom-trained YOLOv8 model to detect drones in aerial and ground surveillance videos.
* **Accident Detection**: Uses a specialized YOLOv8 model trained on accident-related datasets to flag potential road mishaps in videos or dashcam footage.
* **Video Summarization**: Extracts objects and emotions from keyframes to generate a high-level narrative of the video.
* **Real-Time Tracking**: Tracks objects across video frames and outputs a video showing object movement and identification.
* **Question Answering**: Interprets natural language questions and answers them using the media context (e.g., "what are people doing in this image?").

### **3. Technologies Used**

* **Frontend & Server**: FastAPI (web server), Jinja2 (template engine), CORS (API access control)
* **Machine Learning Models**:
  + **YOLOv8** (for object detection, drone detection, accident detection, and real-time tracking)
  + **BLIP** (for image captioning)
  + **DeepFace** (for facial emotion detection)
  + **DeepSeek** (used via OpenRouter for prompt classification and scene description)
* **APIs**:
  + OpenRouter API (for DeepSeek-based chat and classification)
  + Pyngrok (to expose local server publicly)
* **Libraries**: OpenCV (video/image handling), PIL (image processing), Transformers, Torch, NumPy, Uvicorn

### **4. System Architecture**

* **Frontend**: Displays the upload interface and chatbot interaction. Handled using HTML with Jinja2 templates.
* **Backend (FastAPI)**: Manages API endpoints, routes user prompts to ML modules, and handles media file processing.
* **AI Task Modules**: All models are globally initialized for efficient access across multiple requests. Each task is modularized as a function that can be triggered by classified user intent.
* **Prompt Classification**: Prompts are sent to the DeepSeek API, which returns one of the pre-defined task types. This allows the system to intelligently dispatch to the correct vision module.

### **5. Workflow**

1. **User Uploads Media File** (Image or Video)
2. **System Classifies File Type** (Image vs Video)
3. **User Enters a Text Prompt** (e.g., "What do you see in this?" or "Track objects")
4. **Prompt Sent to DeepSeek via OpenRouter**
5. **Classification Output Mapped to Task** (e.g., object\_detection, video\_summarization)
6. **Task Function Invoked**
7. **Result Returned as Text or Annotated Media** (Caption, Scene Description, Summary, etc.)

### **6. Results & Examples**

* **Case 1: Image Analysis**
  + Upload: A photo of a city street
  + Prompt: "Describe the image"
  + Output: Caption - "A crowded urban street with people and vehicles"
  + Prompt: "How many people?"
  + Output: Object Count - {'person': 7}
* **Case 2: Video Analysis**
  + Upload: Classroom video
  + Prompt: "Summarize the video"
  + Output: "Students are seated and learning, emotions detected: happy, focused"
* **Case 3: Drone Detection**
  + Upload: Surveillance video from airport perimeter
  + Prompt: "Are there any drones?"
  + Output: "Detected drone presence at 3 time intervals"
* **Case 4: Accident Detection**
  + Upload: Dashcam footage
  + Prompt: "Detect accidents"
  + Output: "Collision detected in frames 214-230"
* **Case 5: Tracking**
  + Upload: Traffic footage
  + Prompt: "Track moving objects"
  + Output: Downloadable video showing object movement frame-by-frame

### **7. Limitations**

* Real-time tracking performance depends on video size and available memory
* Scene descriptions are abstracted and may miss nuanced details
* DeepSeek may misclassify vague or ambiguous prompts occasionally
* Requires GPU for optimal speed; slower on CPU-based hosting
* Custom models (e.g., drone/accident detection) need further tuning for generalization

### **8. Conclusion**

This project successfully demonstrates an interactive and intelligent system for handling a broad range of computer vision tasks using natural language commands. It bridges the gap between end-users and complex machine learning models through a simplified, modular, and conversational interface. The system is extendable and has potential applications in surveillance, traffic monitoring, public safety, education, accessibility tools, and media analysis.

### **9. Future Enhancements**

* Add Optical Character Recognition (OCR) for detecting text in images
* Integrate Text-to-Speech for vocalizing responses
* Build a modern frontend interface using React or Vue for better UX
* Add support for image segmentation
* Extend question-answering with more context-aware models (e.g., GPT-4 Turbo)
* Expand training datasets for drone and accident detection for higher accuracy
* Introduce multilingual support for broader accessibility

**Appendix:**

* Development environment: Google Colab (with GPU support enabled)
* File handling: Uploads saved to Google Drive for persistent access
* Performance: Models initialized globally to minimize repeated loading and improve execution speed