Project Report: AI-Powered Surveillance System

Project Title: AI-Powered Surveillance System for Suspicious Activity Detection
Name: Madhav Mahajan
GitHub Repository: Madhav-Mahajan-13/AI-Powered-Surveillance-System: This project is a complete implementation of an AI-powered surveillance system using Python. It leverages YOLOv5 for object detection, the SORT algorithm for tracking, and custom logic to detect suspicious activities like abandoned objects and loitering. The entire system is presented through a real-time dashboard built with Streamlit.

---

1. Executive Summary

This report details the design, implementation, and results of a real-time, AI-powered surveillance system developed to automatically detect suspicious activities in video feeds. Our solution addresses critical security needs by identifying abandoned objects and unusual movements (loitering).

The system is built on a modular **"Tracking-by-Detection"** architecture, integrating the high-performance **YOLOv5** object detector with the robust **SORT** multi-object tracker.

The entire system is delivered through an interactive, web-based dashboard built with **Streamlit**, providing security personnel with a user-friendly interface for live monitoring, parameter configuration, and real-time event alerting.

The result is a **modular, efficient, and scalable prototype** that successfully demonstrates a practical application of modern computer vision for enhancing public safety.

---

2. System Architecture

The core of our project is a modular, end-to-end pipeline built on the **"Tracking-by-Detection"** paradigm. This approach is resilient to environmental variations like lighting changes by focusing on the semantic understanding of objects.

Data processing pipeline stages:

1. **Video Ingestion:** Accepts standard video files (MP4, AVI) uploaded via Streamlit dashboard.
2. **Object Detection:** Each frame is processed by pre-trained YOLOv5, detecting objects like `person` or `backpack`.
3. **Multi-Object Tracking:** Detections are passed to **SORT**, which assigns persistent IDs and tracks movement using a Kalman Filter.
4. **State Analysis Engine (`logic_engine.py`):** Analyzes object states to identify suspicious behavior.
5. **Event Generation & Alerting:** Generates timestamped alerts when suspicious behavior is detected.
6. **Real-Time Dashboard (`main_app.py`):** Visualizes live video with analytics and displays a running log of alerts.

---

3. Core Technologies

- **Language:** Python 3.9
- **AI / ML Framework:** PyTorch
- **Object Detection:** YOLOv5s (small, fast, accurate)
- **Object Tracking:** SORT Algorithm (`sort_tracker.py`)
- **Video Processing:** OpenCV
- **Dashboard UI:** Streamlit

---

4. Code Implementation Highlights

Our codebase is modularized into four key files for clarity and maintainability.

4.1 Main Application (`main_app.py`)

Handles Streamlit UI, video processing loop, and module integration.

UI and Configuration:

```
import streamlit as st



st.sidebar.header("Configuration")

confidence_threshold = st.sidebar.slider("Confidence Threshold", 0.0, 1.0,
0.5, 0.05)

abandonment_time = st.sidebar.slider("Abandonment Time (seconds)", 5, 60,
10)

proximity_radius = st.sidebar.slider("Proximity Radius (pixels)", 50, 300,
150)

loitering_time = st.sidebar.slider("Loitering Time (seconds)", 10, 120,
20)
```

Core Processing Loop:

```
while cap.isOpened():

    ret, frame = cap.read()

    if not ret:

        break
    # 1. Detection

    results = model(frame)

    detections_df = results.pandas().xyxy[0]

    detections_df = detections_df[detections_df['confidence'] >
confidence_threshold]



    # 2. Tracking

    detections_for_sort = detections_df[['xmin', 'ymin', 'xmax', 'ymax',
'confidence']].values

    tracked_objects = tracker.update(detections_for_sort)



    # 3. State Analysis

    logic_engine.update_states(tracked_objects, class_names_map)

    abandoned_alerts, abandoned_ids =
logic_engine.check_abandoned_objects()

    loitering_alerts, loitering_ids =
logic_engine.check_loitering(loitering_zone)



    # 4. Visualization and Display

    frame = draw_tracked_objects(...)

    frame_placeholder.image(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

4.2 Logic Engine (`logic_engine.py`)

Contains core intelligence for detecting suspicious behavior.

Abandoned Object Detection:

```
def check_abandoned_objects(self):

    alerts = []

    persons = [s for s in self.object_states.values() if s['class_name']
== 'person']

    person_centroids = [s['history'][-1] for s in persons if s['history']]



    for track_id, state in self.object_states.items():

        if state['class_name'] != 'person' and not state['alerted']:

            if state['stationary_frames'] > self.abandonment_threshold:

                is_abandoned = True

                if person_centroids:

                    current_centroid = state['history'][-1]

                    for p_centroid in person_centroids:

                        dist = np.linalg.norm(np.array(current_centroid) -
np.array(p_centroid))

                        if dist < self.proximity_threshold:

                            is_abandoned = False

                            break
                if is_abandoned:

                    alerts.append(f"ALERT: Abandoned object (ID:
{track_id}) detected!")

                    state['alerted'] = True

    return alerts, [...]
```

4.3 Drawing Utilities (`drawing_utils.py`)

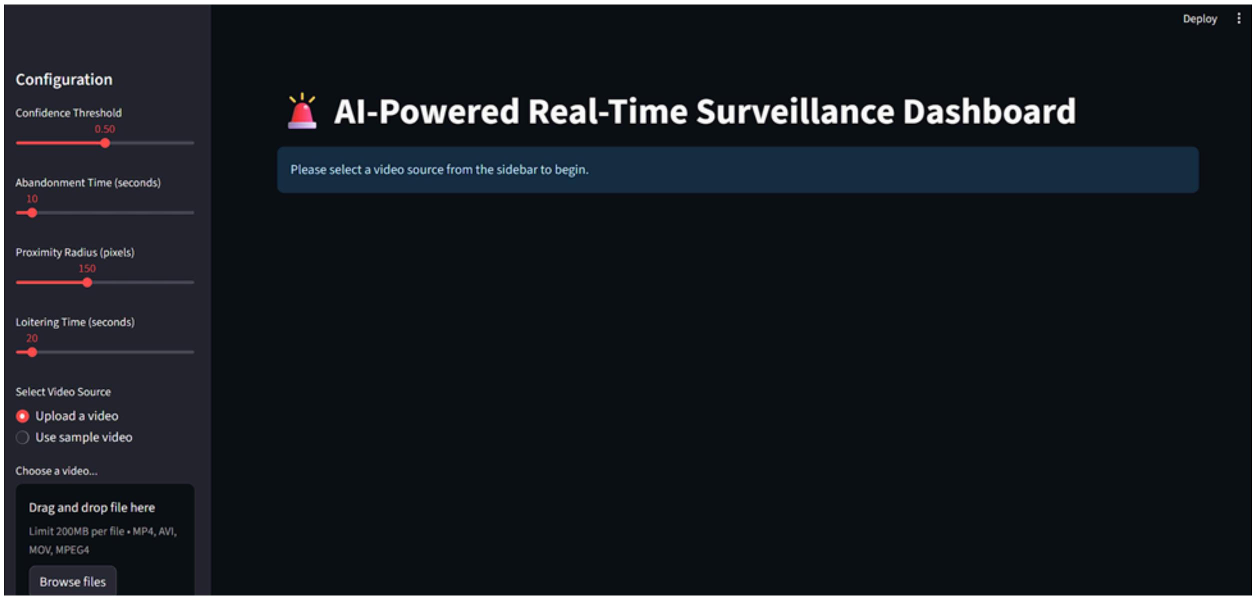
Handles visual overlays such as bounding boxes, labels, and zones.

```
def draw_tracked_objects(frame, tracked_objects, ..., abandoned_ids,
loitering_ids):

    for obj in tracked_objects:

        x1, y1, x2, y2, track_id = map(int, obj)

        class_name = class_names_map.get(track_id, 'object')



        color = (0, 255, 0)  # Green by default

        status = "Normal"



        if track_id in abandoned_ids:

            color = (0, 0, 255)  # Red

            status = "ABANDONED"
        elif track_id in loitering_ids:
            color = (0, 165, 255)  # Orange
            status = "LOITERING"

        cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
        label = f"ID:{track_id} {class_name} - {status}"
        cv2.putText(frame, label, (x1, y1 - 5), ...)
    return frame
```

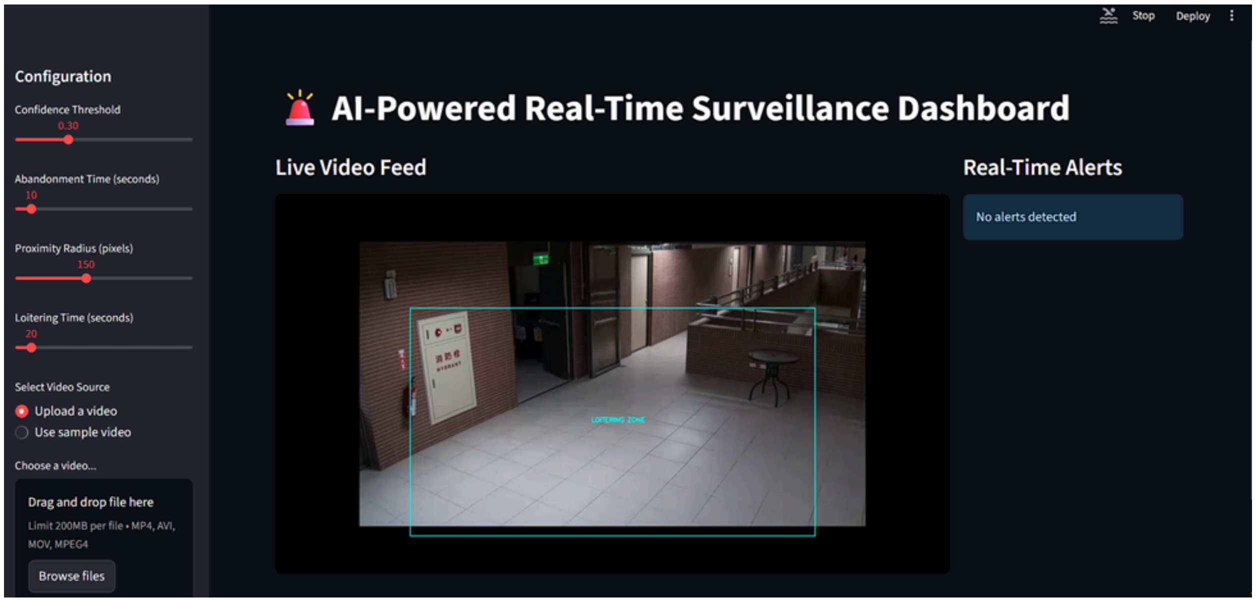
---

5. Results and Demonstration

- **Initial State:** Dashboard prompts video upload; sidebar allows parameter configuration.


Deploy
Configuration
Confidence Threshold
0.50
Abandonment Time (seconds)
10
Proximity Radius (pixels)
150
Loitering Time (seconds)
20
Select Video Source
Upload a video
Use sample video
Choose a video...
Drag and drop file here
Limit 200MB per file • MP4, AVI, MOV, MPEG4
Browse files
AI-Powered Real-Time Surveillance Dashboard
Please select a video source from the sidebar to begin.

- **Video Processing:** Processes frames in real-time; visualizes “Loitering Zone” with a cyan polygon.



Configuration

Confidence Threshold 0.30

Abandonment Time (seconds) 10

Proximity Radius (pixels) 150

Loitering Time (seconds) 20

Select Video Source

[✓ Upload a video]

Use sample video

Choose a video...

Drag and drop file here

Limit 200MB per file • MP4, AVI, MOV, MPEG4

Browse files

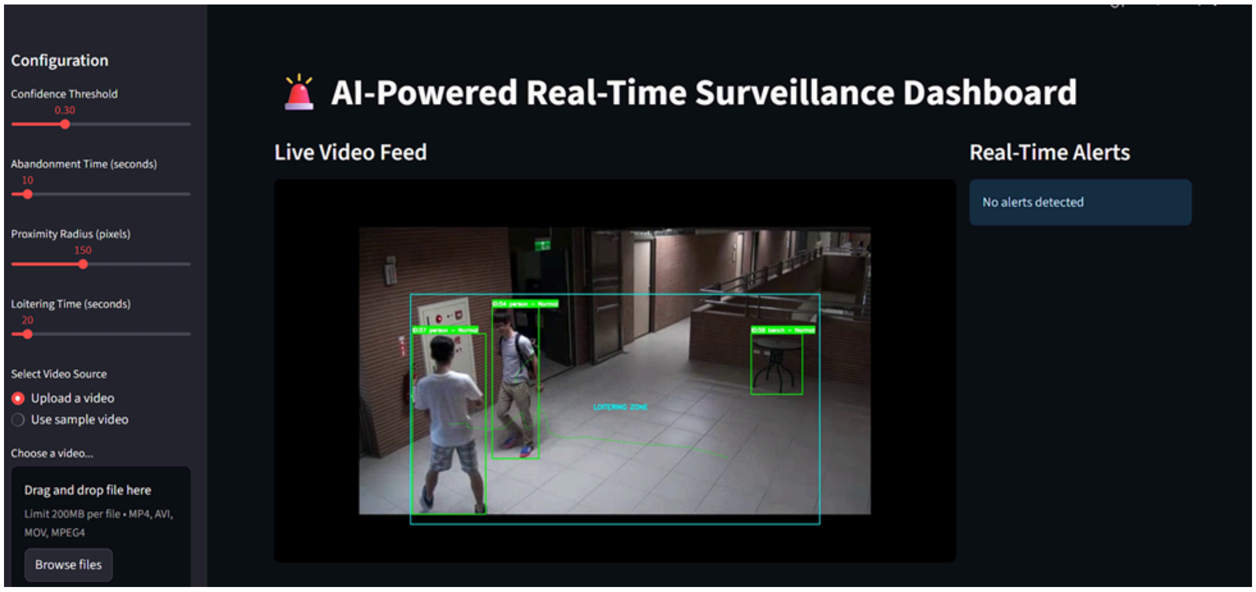
AI-Powered Real-Time Surveillance Dashboard

Live Video Feed

Real-Time Alerts

No alerts detected

- **Object Tracking:** Detects and tracks multiple objects, assigning unique IDs.



Configuration

Confidence Threshold 0.30

Abandonment Time (seconds) 10

Proximity Radius (pixels) 150

Loitering Time (seconds) 20

Select Video Source

[✓ Upload a video]

Use sample video

Choose a video...

Drag and drop file here

Limit 200MB per file • MP4, AVI, MOV, MPEG4

Browse files

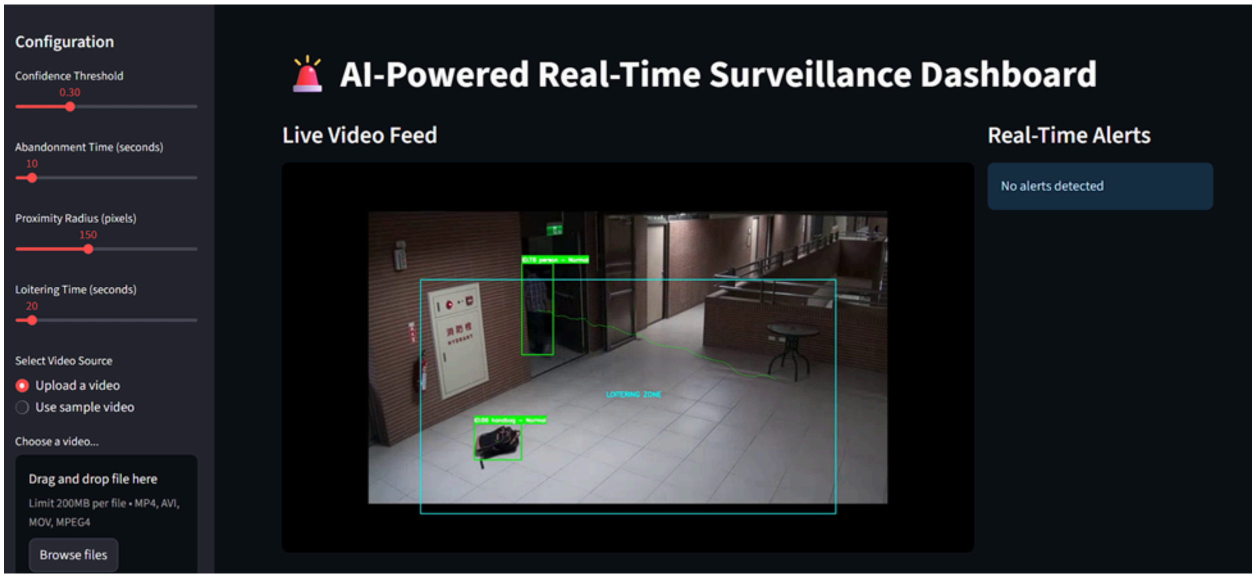
AI-Powered Real-Time Surveillance Dashboard

Live Video Feed

Real-Time Alerts

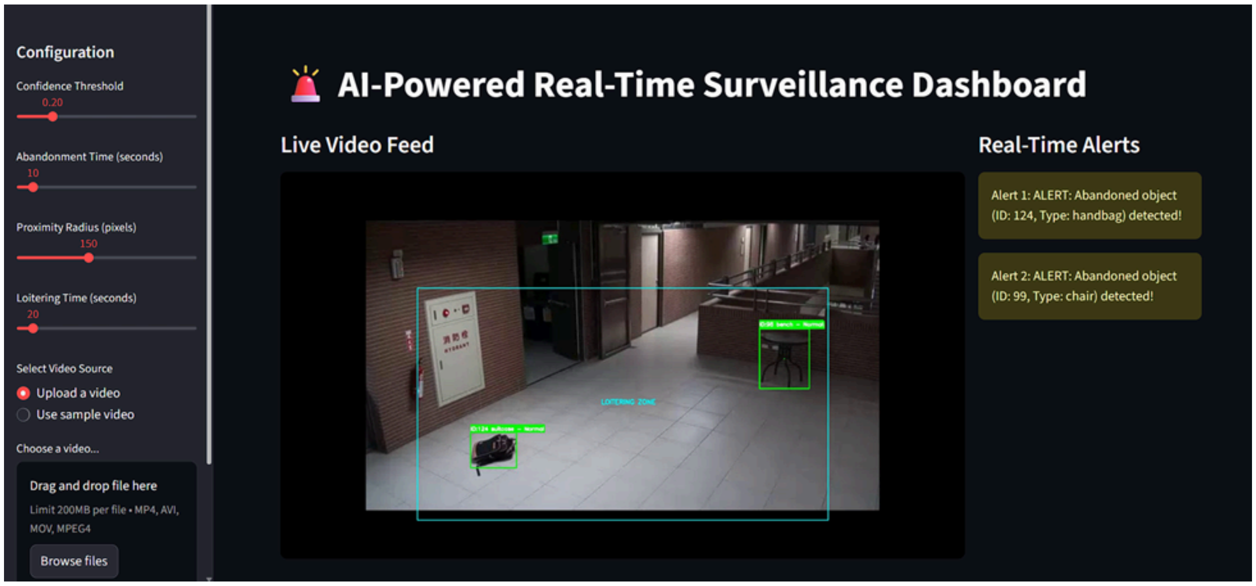
No alerts detected

- **Trajectory Visualization:** Shows recent paths of moving objects.



No alerts detected

- **Alert Generation:** Flags abandoned objects with red bounding boxes and logs alerts.



Alert 1: ALERT: Abandoned object (ID: 124, Type: handbag) detected!

Alert 2: ALERT: Abandoned object (ID: 99, Type: chair) detected!

6. Conclusion and Future Work

We successfully developed a functional prototype of an AI-powered surveillance system for detecting abandoned objects and loitering.

Future Enhancements:

1. Support for **live RTSP camera feeds**.
2. Advanced **anomaly detection** like fights, falls, or crowd panic.
3. Upgrade to **DeepSORT** for better long-term tracking.
4. **Interactive zone definition** directly on the dashboard UI.