Project Report: Real-Time AI-Powered Surveillance System

Hackathon: Honeywell Campus Connect Program

Project Title: AI-Powered Surveillance System for Suspicious Activity Detection

Name: Madhav Mahajan

1. Executive Summary

This report details the design and implementation of a real-time, AI-powered surveillance system developed to automatically detect suspicious activities in video feeds. The system addresses critical security needs in public and private spaces by identifying two key event types: **abandoned objects** and **unusual movements (loitering)**. Our solution leverages a state-of-the-art "Tracking-by-Detection" architecture, integrating the high-performance YOLOv5 object detector with the robust SORT multi-object tracking algorithm. 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. Introduction

The increasing need for security in public areas such as airports, train stations, and commercial centers has led to a massive deployment of surveillance cameras. However, manual monitoring of these video feeds is inefficient, prone to human error, and not scalable. This project tackles this challenge by creating an intelligent system that automates the surveillance process, acting as a "second pair of eyes" to flag potentially dangerous situations that a human operator might miss.

Our system focuses on two primary threat vectors:

1. **Object Abandonment:** An unattended bag or package could potentially contain hazardous materials. Our system is designed to detect objects that remain stationary for a prolonged period without a person nearby.
2. **Unusual Movement:** Individuals loitering in restricted areas or for extended periods can be a precursor to criminal activity. Our system identifies and flags such behavior based on user-defined zones and time thresholds.

By automating the detection of these events, our system enhances situational awareness and allows security personnel to respond more quickly and effectively to potential threats.

3. System Architecture

The core of our project is a modular, end-to-end pipeline built on the **"Tracking-by-Detection"** paradigm. This modern approach is more robust than classical techniques (like background subtraction) because it focuses on the semantic understanding of objects rather than low-level pixel changes, making it resilient to environmental variations like lighting changes.

The data processing pipeline consists of the following stages:

1. **Video Ingestion:** The system accepts a standard video file (MP4, AVI) uploaded by the user.
2. **Object Detection:** Each frame is processed by a pre-trained **YOLOv5** model, which identifies and localizes all relevant objects (e.g., 'person', 'backpack', 'suitcase') with high accuracy.
3. **Multi-Object Tracking:** The detections are passed to the **SORT (Simple Online and Realtime Tracking)** algorithm. SORT assigns a unique, persistent ID to each object and tracks its movement across frames using a Kalman Filter for motion prediction. This is the crucial step that enables behavioral analysis.
4. **State Analysis Engine:** The tracked object data (IDs, positions, classes) is fed into our custom logic engine, which analyzes object states over time to identify suspicious behavior.
5. **Event Generation & Alerting:** When the logic engine's rules are met (e.g., an object is abandoned), it generates a timestamped alert.
6. **Real-Time Dashboard:** A **Streamlit** web application visualizes the live video feed with overlaid bounding boxes and track IDs, and displays a running log of all generated alerts.

4. Core Technologies

Our technology stack was chosen to prioritize rapid development, high performance, and ease of integration, which is critical in a hackathon environment.

- **Language: Python 3.9**
- **AI / ML Framework: PyTorch** (for running the YOLOv5 model)
- **Object Detection: YOLOv5s** (A small, fast, and accurate pre-trained model)

- **Object Tracking: SORT Algorithm** (Provides a robust balance of speed and occlusion handling)
- **Video Processing: OpenCV** (The primary tool for reading and manipulating video frames)
- **Dashboard UI: Streamlit** (A pure-Python framework for building interactive web applications with minimal effort)

5. Feature Implementation

5.1. Abandoned Object Detection

Our logic for detecting abandoned objects is based on a robust, two-factor rule system that minimizes false alarms. An object is flagged as "abandoned" only when both conditions are met:

1. **The Stationarity Rule (Temporal):** The object must remain motionless for a user-defined period (e.g., 10 seconds). Our system tracks the position of each object, and if its centroid does not move beyond a small pixel threshold, a stationary_timer is incremented.
2. **The Unattended Rule (Spatial):** The object must be spatially distant from any person. Once an object's stationary_timer exceeds the abandonment threshold, the system performs a proximity check. It calculates the distance to all detected persons. If no person is within a configurable radius (e.g., 150 pixels), the object is officially flagged as abandoned.

This object-centric approach is highly effective because it directly analyzes the behavior of tracked items, unlike older methods that were easily confused by environmental noise.

5.2. Unusual Movement (Loitering) Detection

To detect loitering, we implemented a trajectory-based analysis method. This allows for the creation of specific, rule-based security policies.

1. **Zone Definition:** A static "no-loitering" zone is defined as a polygon on the video frame. In a future version, this could be drawn interactively by the user.
2. **Trajectory Analysis:** The system tracks the history of each person's location.
3. **Loitering Logic:** For each person, the system checks if their current position is inside the defined loitering zone. If it is, a loitering_timer for that individual is incremented. If the person remains in the zone for a duration exceeding the user-defined threshold (e.g., 20 seconds), a "Loitering Detected" alert is triggered.

6. Results and Demonstration

The final system is presented as an intuitive web dashboard. The user can upload their own video file and configure key detection parameters in real-time via sliders and input boxes in the sidebar.

The main interface is split into two sections:

- **Live Video Feed:** Displays the processed video stream. Detected objects are enclosed in bounding boxes, color-coded to indicate their status (Green: Tracked, Orange: Stationary Candidate, Red: Alert). Each box is labeled with the object's unique track ID.
- **Real-Time Alerts:** A log on the side of the video feed instantly displays timestamped alerts as they are detected, ensuring operators are immediately notified of suspicious events.

The application successfully demonstrates a complete, end-to-end workflow from video input to actionable intelligence, proving the viability of our architectural approach.

7. Conclusion and Future Work

In this project, we successfully developed a functional prototype of an AI-powered surveillance system capable of detecting abandoned objects and loitering behavior in real-time. By leveraging powerful open-source tools like YOLOv5, SORT, and Streamlit, we were able to build a robust and user-friendly application within the time constraints of a hackathon.

For future development, the system could be extended in several ways:

- **Live Camera Feeds:** Integrate support for RTSP streams from IP cameras.
- **Advanced Anomaly Detection:** Implement more complex behavior analysis, such as detecting fights, falls, or crowd panic using optical flow analysis.
- **Improved Tracking:** Upgrade the tracker to **DeepSORT**, which uses appearance features to better handle long-term occlusions, albeit at a higher computational cost.
- **Synthetic Data Augmentation:** Use Generative Adversarial Networks (GANs) to create synthetic training examples of rare events, further improving the robustness of the detection models.
- **Interactive Zone Definition:** Allow users to draw and save custom detection zones directly on the dashboard UI.