***A logo for a university

AI-generated content may be incorrect.***

Name: Omar Nagy

ID:227750

***Anomaly Detection CCTV***

***1. Introduction***

**Anomaly detection in CCTV footage has become an essential tool for modern security systems. Traditionally, human operators would manually review hours of video, a time-consuming and error-prone task. However, with advancements in computer vision and machine learning, automated systems can now detect suspicious activities in real time, making surveillance more efficient and responsive. This report explores a computer vision-based anomaly detection system using YOLOv8 for detecting loitering and unusual activities in CCTV footage.**

**The aim of this project is to design a system that can analyze real-time video streams or recorded footage, identify individuals, track their movements, and detect anomalies, specifically loitering, using YOLOv8-based object detection. This system can be deployed in various surveillance applications, alerting security personnel to potential threats, enabling proactive responses to prevent harm.**

**2. *Literature Review***

**Existing Techniques**

**Traditional anomaly detection methods in CCTV footage include background subtraction, optical flow, and frame differencing. These methods analyze the differences between frames to detect motion, but they are susceptible to high false positive rates in dynamic environments. More advanced methods leverage machine learning, particularly deep learning, to improve accuracy. CNNs (Convolutional Neural Networks) and object detection models, such as YOLO (You Only Look Once), have become popular due to their ability to perform real-time object detection with high accuracy.**

***YOLOv8***

**YOLO is a deep learning-based object detection model designed to be fast and efficient for real-time processing. YOLOv8, the latest version of the YOLO series, improves upon previous versions with enhanced accuracy and speed. It is widely used in surveillance systems for detecting various objects in video frames, including people, vehicles, and other potential threats.**

***Loitering Detection***

**Loitering, or the act of lingering in a specific area without any apparent purpose, is often considered a sign of suspicious activity. Detecting loitering behavior in surveillance footage requires monitoring a person's position over time and determining whether they stay within a specific area for an extended period. This type of detection plays a critical role in identifying potential threats in environments such as shopping malls, airports, and public spaces.**

***3. Methodology***

**System Design**

**The system is designed to process CCTV footage (or webcam video), detect people using YOLOv8, and track their movements across frames. When a person stays in the same area for an extended period, they are flagged for loitering. The system also provides real-time feedback in the form of bounding boxes, tracking trails, and an information panel that shows the number of active tracks and loitering alerts.**

**Tools and Libraries Used**

**The system leverages the following tools:**

* **OpenCV: A library for computer vision tasks such as video processing, image manipulation, and object tracking.**
* **NumPy: A library for numerical computations, used for mathematical operations in the code.**
* **YOLOv8: An object detection model from the ultralytics package, used for detecting people in video frames.**

***4. Implementation***

**Code Explanation**

**The CCTV Anomaly Detection class is at the core of the system. Here's a breakdown of the key functions:**

1. **Initialization (\_\_init\_\_):**
   * **Sets up the video source, output directory, and frame interval.**
   * **Initializes YOLOv8 for people detection and sets tracking parameters.**
2. **People Detection (\_detect\_people):**
   * **Converts each frame to RGB and uses YOLOv8 to detect people.**
   * **Extracts the bounding boxes and centers of detected people.**
3. **Tracking (\_track\_people):**
   * **Matches new detections with existing tracked people based on Euclidean distance.**
   * **Updates the tracks with new detections.**
4. **Loitering Detection (\_detect\_loitering):**
   * **Detects loitering by checking if a person stays in the same area for a specified time threshold (loitering\_time\_threshold).**
   * **Alerts are triggered when the person stays within a small area for longer than the threshold.**
5. **Drawing Tracking Trails (\_draw\_tracking\_trail):**
   * **Draws lines between consecutive detection points to visualize the movement of tracked individuals.**
6. **Information Panel (\_add\_info\_panel):**
   * **Displays an overlay with real-time information about the active tracks, loitering alerts, and total people detected.**
7. **Video Processing (process\_video):**
   * **Captures the video feed, processes each frame, detects people, tracks their movements, and detects loitering.**
   * **Saves the processed video with overlays to the output directory and displays the frames in real time**

***Code Example:***

**Here’s a simplified snippet of the code that detects people and tracks them across frames:**

**def \_detect\_people(self, frame):**

**rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)**

**results = self.yolo\_model.predict(source=rgb\_frame, imgsz=416, conf=0.5)**

**people = []**

**for result in results:**

**for box in result.boxes:**

**cls = int(box.cls[0])**

**conf = float(box.conf[0])**

**if cls == 0 and conf > 0.5:**

**x1, y1, x2, y2 = map(int, box.xyxy[0])**

**people.append({'bbox': (x1, y1, x2-x1, y2-y1), 'center': (x1 + (x2-x1)//2, y1 + (y2-y1)//2)})**

**return people**

***5. Results and Discussion***

**Evaluation**

**The system was evaluated based on its ability to detect people, track their movements, and identify loitering. The YOLOv8 model provided accurate people detection, with bounding boxes drawn around detected individuals. The tracking mechanism efficiently followed people across frames, even in crowded scenes. Loitering detection worked well, triggering alerts for individuals who remained in the same area for more than the specified threshold.**

**Performance Optimization**

**To optimize performance, the frame processing interval was adjusted based on the type of video source (webcam or video file). The YOLOv8 model was configured with a frame size of 416x416, balancing accuracy and speed. Further optimizations could include using a smaller model variant like YOLOv8n for faster processing in real-time applications.**

**Possible Improvements**

* **Advanced Tracking: Integrating advanced tracking algorithms such as DeepSORT or SORT could improve person re-identification and tracking accuracy.**
* **False Positives: The system occasionally flagged non-loitering individuals as loiterers due to short pauses. Implementing additional behavior analysis (e.g., direction changes) could reduce false positives.**
* **Scalability: The system could be expanded to handle multiple cameras and larger surveillance areas, with a centralized monitoring dashboard for security personnel.**

***6. Conclusion***

**In conclusion, the CCTV Anomaly Detection system based on YOLOv8 has successfully demonstrated the potential of using deep learning for real-time surveillance and anomaly detection. By combining object detection with tracking and loitering detection, the system provides valuable tools for security personnel to identify suspicious behavior. While the system works effectively in controlled environments, further optimization and expansion are needed to handle more complex real-world scenarios, such as crowded spaces or varying environmental conditions.**