Real-Time Detection of Object Missing and New Object Placement in Video

Overview:

This project focuses on building a real-time video analytics system capable of detecting two key scenarios:

1. **Missing Object Detection** – Identifying when a previously visible object disappears from the scene.
2. **New Object Placement Detection** – Identifying when a new object appears in the scene.

The primary goal is to create a reliable, high-performance pipeline for detecting these events as they occur, with real-time alerts displayed on the video feed. The system uses deep learning techniques for object detection and continuously processes video frames to ensure the analytics are accurate and timely.

How It Works:

1. **Object Detection:**
    - The project uses a **YOLO (You Only Look Once)** object detection model, which is a deep learning model optimized for real-time detection. YOLO processes each video frame to identify and classify objects.
    - The model outputs the object classes (e.g., car, person, dog) and their respective positions (bounding boxes) in the frame.
2. **Tracking Missing Objects:**
    - The system keeps track of objects that were previously detected in earlier frames.
    - If an object that was previously detected disappears from the frame in subsequent video feeds, it is flagged as **"missing"**.
    - A message such as "Missing Object: [Object Name]" is displayed on the screen, alerting the user of the absence.
3. **Detecting New Objects:**
    - The system compares the current frame's objects with the ones detected in the previous frame.
    - Any object that appears for the first time in the current frame (but wasn't detected in the previous frame) is flagged as a **"new object"**.
    - A message like "New Object Detected: [Object Name]" is displayed, notifying the user of the new presence.
4. **Real-Time Video Processing:**
    - The pipeline processes each video frame in real-time using a webcam.
    - The FPS (Frames Per Second) rate is monitored to ensure that the detection system can run smoothly and efficiently. Although the FPS may vary depending on hardware (CPU vs. GPU), the system is designed to process and display results in real-time.

Key Features:

- **Real-Time Object Detection:** The system processes each frame in real-time, ensuring immediate feedback.
- **Detection Alerts:** Alerts for missing and new objects are shown on the video feed, making it easy to monitor the changes dynamically.
- **Dynamic Object Tracking:** Tracks objects across frames and compares current detections with past detections.

Detection/tracking model or approach

Detection Model:

- **YOLO (You Only Look Once) is used for object detection in this code.**
    - Why YOLO?
        - YOLO is a popular real-time object detection model known for its high speed and good accuracy. It processes the entire image at once and detects multiple objects in one pass.
        - YOLO outputs the bounding boxes, object class, and confidence score for each detected object in the frame.
        - The function detect_objects(frame, net, output_layers) performs object detection using the YOLO model. It preprocesses the image, performs a forward pass through the YOLO network, and extracts the bounding boxes, class IDs, and confidence scores of the detected objects.

Tracking Approach:

- **MultiTracker with CSRT (Channel and Spatial Reliability Tracker):**
    - Why MultiTracker and CSRT?
        - cv2.MultiTracker_create() creates a tracker that can track multiple objects simultaneously in a video. It adds a tracker for each object and tracks them across frames.
        - CSRT (Channel and Spatial Reliability Tracker) is used in the code for each object. It is a robust tracker that works well with objects that are in motion or partially occluded, providing good accuracy for real-time tracking.
        - The function update_tracking(objects, frame) is intended to initialize and update the tracker, allowing it to track the detected objects across frames.

Optimize the Object Detection Model:

a. Use a Faster YOLO Version:

- **YOLOv3** is a relatively heavy model. Consider using **YOLOv4-tiny** or **YOLOv5**. These models are optimized for speed and can process frames faster with a slight trade-off in detection accuracy. YOLOv5, in particular, provides a good balance between speed and accuracy and is easier to deploy.

b. Model Quantization:

- **Quantize the model** to reduce its size and speed up inference. Converting the model to use **8-bit integers** instead of **32-bit floats** can improve inference speed, especially on hardware with limited computational resources. You can use **TensorFlow Lite** or **OpenVINO** for quantization.

2. Improve Object Tracking Efficiency:

a. Switch to Faster Trackers:

- **CSRT** tracker is slow, though robust. Instead, we can use faster trackers like **KCF (Kernelized Correlation Filters)** or **MOSSE (Minimum Output Sum of Squared Error)**. These trackers are faster and provide real-time tracking capabilities.

Hardware configuration used for testing

1. CPU:

- Intel Core i5:

  A mid-range or high-end CPU will allow your system to process frames and handle the object detection model without significant bottlenecks. These processors are efficient and capable of handling multi-threaded operations required for tasks like object tracking and detection.

2.RAM:

- **16 GB DDR4**:
  - **8 GB DDR4** can work, but for smoother performance, especially when working with high-resolution video streams or large models like YOLOv3, **16 GB of RAM** is recommended.
  - More RAM helps when working with larger video files or when using advanced tracking algorithms that maintain large tracking data in memory.

3. Storage:

- **SSD (Solid State Drive)** — 500 GB or more:
  - An SSD will significantly reduce video read/write time, which is important if you are processing large video files or streaming real-time data.

4. Operating System:

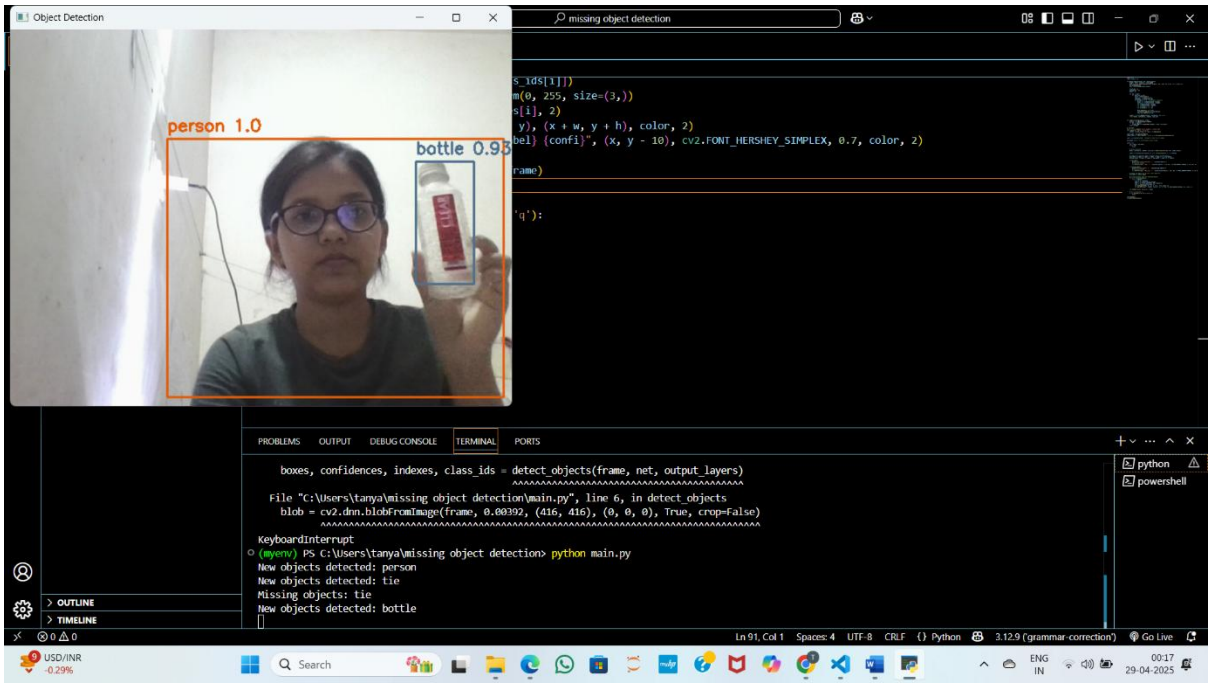
- **Windows 10/11 (64-bit)** :
  - A 64-bit OS is essential to handle larger models and datasets.

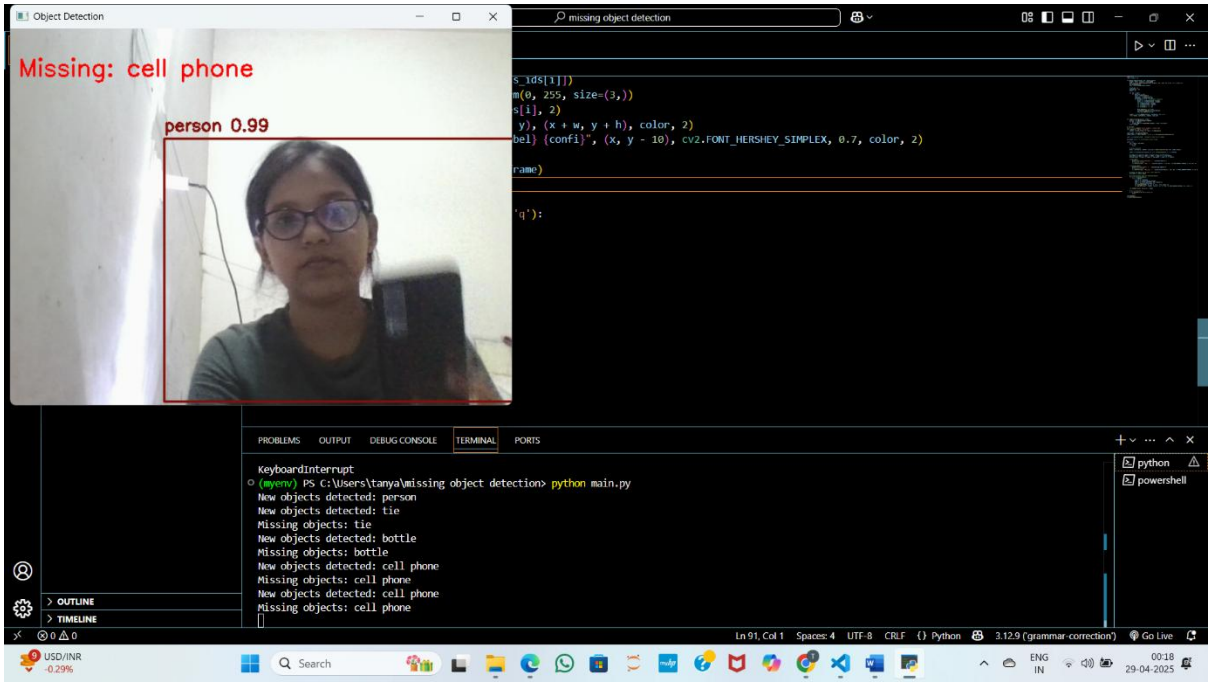
5. Webcam/Camera:

- **1080p or 4K Camera (for real-time webcam input):**
    - A 1080p webcam or higher is recommended for object detection in real-time applications. This ensures that the camera captures enough resolution for the model to detect and track objects accurately.

Output :

Screenshots or sample output frames:





output video Link:
https://drive.google.com/file/d/1v15u4pBEXRtNzUiiAqPhwq6cC87LNHGc/view?usp=drivesdk

github link : https://github.com/tanya58/Real-Time-Detection-of-Object-Missing-and-New-Object-Placement-in-Video