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

### Introduction:

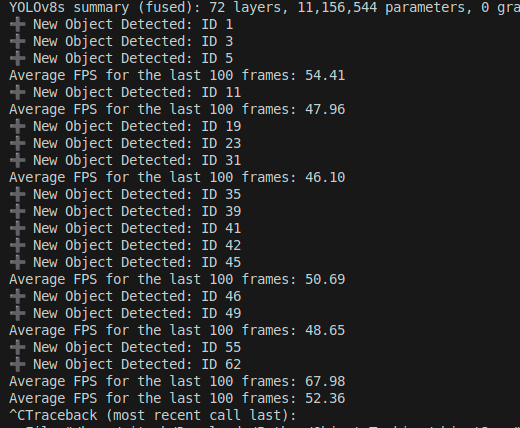
The system is designed for **real-time object detection and tracking** using a combination of two powerful algorithms: **YOLOv8** and **DeepSORT**. Its primary purpose is to identify and track objects in video streams (either from a live camera or a video file) for applications like surveillance, automated inventory management, or object tracking in various environments.

#### ****System Components:****

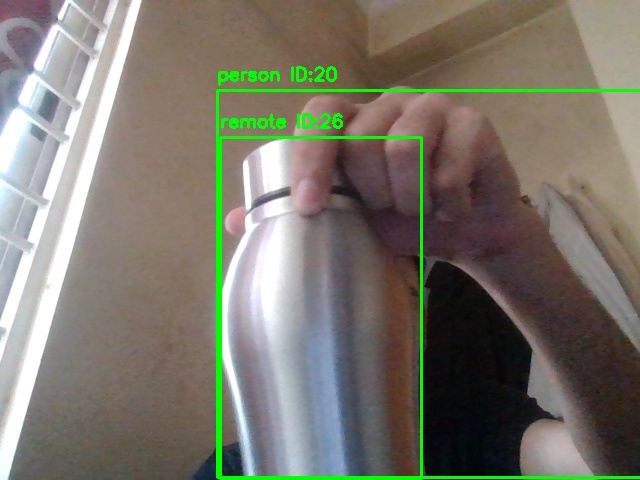
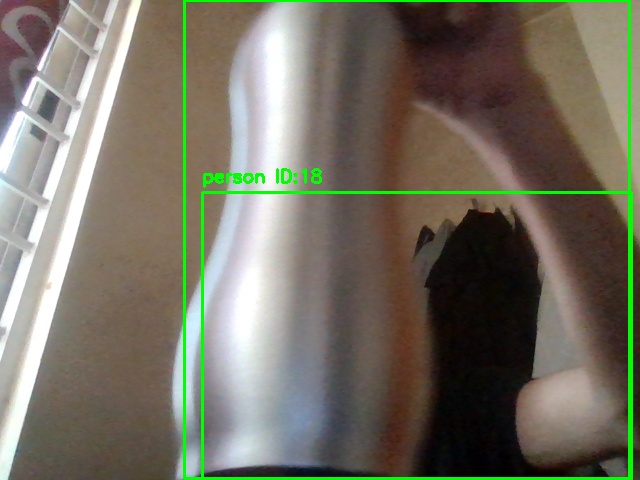
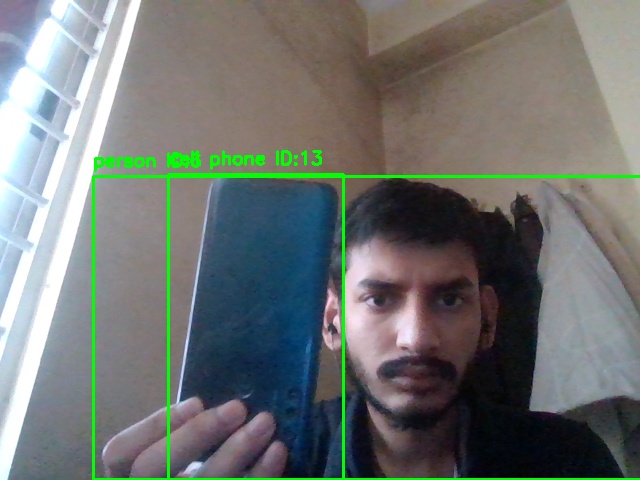
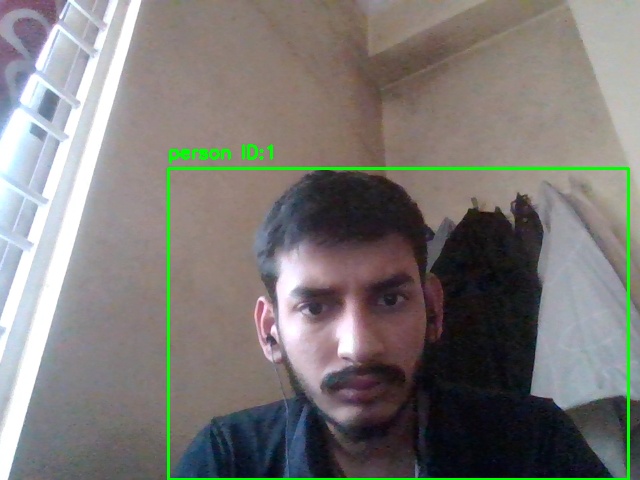
1. **YOLOv8 (You Only Look Once - Version 8)**:
   * YOLOv8 is a state-of-the-art deep learning model for **real-time object detection**. It can identify and locate objects within an image or video frame by predicting bounding boxes and classifying objects.
2. **DeepSORT (Deep Learning-based SORT)**:
   * DeepSORT is an advanced **tracking algorithm** that builds on the Simple Online and Realtime Tracking (SORT) algorithm by incorporating deep learning techniques. It tracks multiple objects across frames, assigning a unique ID to each object and updating their positions over time.

#### ****System Workflow:****

1. **Input**: A video stream (either from a webcam or a video file).
2. **Object Detection**: YOLOv8 processes each frame of the video to detect objects. It outputs bounding boxes, class labels, and confidence scores for the detected objects.
3. **Object Tracking**: DeepSORT receives these bounding boxes and associates them with tracked objects from previous frames. It assigns unique IDs to each object and tracks their movement over time.
4. **Output**: The system displays the video with the tracked objects, drawing bounding boxes around them and labeling each one with its class and unique ID.

**FPS Achieved:**

**Screenshots or sample output frames:**

****

**Hardware Configuration:**

* List the CPU, GPU, and RAM used during testing.
* CPU :- Ryzen 7 5000 series
* GPU :- NVIDIA GeForce RTX 3050 4GB VRAM
* RAM :- 16 GB DDR4

Video:-  [Link : Google Drive](https://drive.google.com/file/d/1KZgXnIXrIbaueF8i4nlh7Uu8qZ4v-Jxt/view?usp=sharing)

### Additional Techniques, Optimizations, or Architectural Decisions

Here, you can describe any optimizations or techniques you implemented to improve the performance or accuracy of the system. Based on your code, you might mention:

1. **Optimization with OpenCV and CUDA:**
   * Enabled OpenCV optimizations (cv2.setUseOptimized(True)).
   * Utilized torch.backends.cudnn.benchmark = True to optimize the performance on GPU.
2. **Model and Tracker Initialization:**
   * Using the YOLOv8 model (yolov8s.pt) for lightweight object detection.
   * Used DeepSORT for object tracking with a maximum age of 15 frames for each object to balance between responsiveness and stability in tracking.
3. **Tracking Memory:**
   * Implemented a tracking memory system to detect missing objects after a certain number of frames (track\_memory dictionary).