**Real-Time Object Change Detection Using YOLOv8**

**Project Description**

The objective of this project is to develop a real-time object change detection system that identifies newly appeared and missing objects from a live video stream or recorded video.  
The model uses YOLOv8 and is optimized for CPU-only systems while maintaining high FPS and robustness.

**FPS Achieved**

| **Mode** | **FPS (Approximate)** |
| --- | --- |
| Without Video Saving | ~28–30 FPS |
| With Video Saving | ~18–22 FPS |

FPS is measured using OpenCV's timer mechanism during runtime.

**Hardware Configuration**

* **CPU:** Intel Core i5-1135G7 @ 2.40GHz
* **GPU:** Integrated Intel Iris Xe Graphics
* **RAM:** 8 GB
* **Operating System:** Windows 11 / Ubuntu 20.04
* **Python Version:** 3.10

**Techniques, Optimizations, and Architectural Decisions**

* **YOLOv8 Model:** Used a lightweight model for real-time performance.
* **Efficient Frame Processing:** Resized frames to 480x360 to accelerate detection.
* **Selective Frame Saving:** Only save frames when new objects appear or disappear to reduce disk writing overhead.
* **Robust Missing Object Detection:** Implemented counter-based missing detection to avoid false alarms due to momentary detection loss.
* **Optional Output Video Writing:** Video writing can be disabled during testing to further boost FPS.
* **Dockerization:** Containerized the entire application for environment-independent deployment.

**Sample Outputs**

* Event frames are stored inside the output\_frames/ folder.
* The output video output\_video.mp4 shows the real-time event detection.

**Docker Instructions**

**Build the Docker image:**

docker build -t object-change-detection .

**Run the container:**

docker run --rm -it --device=/dev/video0 object-change-detection

*Note: On Windows, ensure Docker Desktop is configured properly for webcam access.*

**Conclusion**

The system successfully detects appearing and missing objects in real-time with high performance on CPU devices.  
It is optimized for robustness and can be easily deployed via Docker for production use.