**Project Report: End-to-End Image Segmentation & Object Tracking Pipeline** **for Labeller AI**

**Candidate Name:** Kiranjot Kaur

**SID:** 23106059

**College:** PEC, Chandigarh

**Submission Date:** 23rd September 2025

**Table of Contents**

**1. Introduction**

**2. Methodology**

2.1 Dataset Creation & Annotation

2.2 Model Training

**3. Problems Faced and Resolutions**

3.1 Challenge 1: Class Imbalance

3.2 Challenge 2: Small Object Detection

**4. A Guide to Building Your Own Object Tracker with Labellerr AI**

**5. Conclusion**

**6. References**

**Introduction**

This report documents the development of an end-to-end computer vision pipeline for vehicle and pedestrian tracking, as outlined in the Labellerr AI technical assessment. The project's primary objectives were to demonstrate proficiency in:

* Data annotation and management using the Labellerr Platform.
* Training a YOLO-Seg instance segmentation model via transfer learning.
* Integrating the model with a robust tracking algorithm, ByteTrack, to process video data.
* Creating a live demo to showcase the final system.

**Methodology**

The project followed a structured, iterative approach to develop the computer vision system.

2.1 Dataset Creation & Annotation

To fulfil the assignment's requirement of using a custom dataset, I collected approximately 150 raw images of vehicles and pedestrians from various sources. These images were specifically chosen to present challenges such as occlusions, diverse lighting conditions, and varying object sizes.

* A new project was created on the Labellerr platform for data annotation.
* Manual Annotation: A total of 100 images were meticulously annotated with polygon masks for the vehicle and pedestrian classes. The annotation process was expedited using Labellerr's features.
* Data Export: The annotated data was exported from Labellerr in the YOLO format, ready for model training.

2.2 Model Training

A YOLOv8-seg model was chosen for its balance of speed and accuracy. The model was trained using a fine-tuning approach with pre-trained weights from the yolov8n-seg.pt checkpoint. The training was conducted for 100 epochs as required.

* Training Environment: All training was performed on a Google Colab notebook, utilising a free GPU to handle the computational load.
* Hyperparameters: Key hyperparameters such as learning rate and batch size were left at their default values for the baseline model, but adjustments were made during the refinement phase.

**Problems Faced and Resolutions**

Throughout the project, two key challenges were identified and addressed, demonstrating a hands-on approach to debugging.

3.1 Challenge 1: Class Imbalance.

* Problem: The initial dataset had a severe class imbalance, with a disproportionately large number of vehicles compared to pedestrians. This caused the baseline model to perform poorly on detecting pedestrians.
* Resolution: I collected and annotated an additional 20 images focused solely on pedestrians in various settings (e.g., crowded scenes, occluded views). This balanced the dataset and significantly improved the model's recall for the pedestrian class.

3.2 Challenge 2: Small Object Detection.

* Problem: When images were resized to the default 640x640 during training, small objects in the distance (e.g., a car far down the road) were compressed to a point where the model could not detect them.
* Resolution: I re-trained the model with a higher input image size (imgsz=1024), which provided more pixel information for small objects. This resulted in a noticeable improvement in the model's performance on distant objects.

**A Guide to Building Your Own Object Tracker with Labellerr**

This section provides a high-level guide to help fellow developers build a similar project.

1. Data Curation is Key: Do not underestimate the importance of your data. Start by identifying the specific object types and challenging scenarios you need to address. Collect raw images that represent these real-world conditions.
2. Annotate with Precision: Use the Labellerr platform for your annotations. The polygon mask tool is precise, and the platform's features can greatly speed up the process. Pay close attention to consistency in your labels.
3. Leverage Transfer Learning: For most custom projects, fine-tuning a pre-trained model is the most effective strategy. Use a framework like Ultralytics YOLO to load a pre-trained segmentation model and train it on your custom data.
4. Integrate Detection with Tracking: Don't reinvent the wheel. Use a purpose-built tracking algorithm like ByteTrack. It's designed to work seamlessly with object detectors and handles common issues like occlusions.
5. Build a Feedback Loop**:** The most critical step is to use your model's predictions to refine your dataset. Upload predictions to a new Labellerr project and use a human-in-the-loop approach to correct mistakes. This iterative process is how you achieve a truly high-performing model.

**Conclusion**

The project successfully demonstrated the complete machine learning lifecycle, from data creation to a deployed tracking demo. The combination of the Labellerr platform for data management, a fine-tuned YOLO-Seg model for accurate segmentation, and ByteTrack for robust tracking resulted in a powerful and functional computer vision system. The problems faced and their resolutions proved that a practical approach to data and model refinement is essential for success.

**References**

**i. *https://www.labellerr.com/blog/how-to-implement-bytetrack***

**ii. *https://datature.io/blog/introduction-to-bytetrack-multi-object-tracking-by-associating-every-detection-box***