End-to-End Object Tracker Development Report

1. Introduction

This document outlines the complete process of building an end-to-end image segmentation and object tracking pipeline as required by the technical assessment. The project focused on creating a semantic/instance segmentation workflow for vehicles and pedestrians using

YOLO-Seg and **Byte Track**, with the **Labellerr platform** serving as a central hub for data management and quality control. This report details the journey, challenges faced, and a guide for reproducing the workflow.

---

2. The Project Journey

The development process followed a structured machine learning lifecycle from data creation and annotation to model training, deployment, and quality assurance.

1. **Data Collection and Preparation**: The first step involved collecting images for the dataset. Following the assignment guidelines, permissibly licensed images of vehicles and pedestrians were gathered from online sources. A folder structure was created on a local machine to organize these images, which were then zipped for efficient upload to Google Drive. It is also encouraged to create your own synthetic dataset or mix it with real data.
2. **Labellerr Project Setup**: Two separate projects were created on the Labellerr platform as required by the assignment.
    - **Training Project**: A project with images intended for manual annotation. This project was set up with "vehicle" and "pedestrian" as object classes using the

Polygon annotation type for segmentation.

    - **Test Project**: A separate project for the test set, configured with the same object classes but without manual annotations. This project serves as the destination for model-generated predictions for later review.
3. **Annotation & Data Export**:
    - Images in the training project were manually annotated using the polygon tool to create segmentation masks for all instances of vehicles and pedestrians. A minimum of 100 images were annotated to meet the project's requirements.
    - Upon completion, the annotated data was exported from the Labellerr platform in a format suitable for YOLO training. This includes the images and their corresponding

.txt label files in the necessary structure.

4. **Model Training**:
    - A

Google Colab notebook was used for the training process to leverage its free GPU resources. The exported data was unzipped in the Colab environment.

- The data.yaml configuration file was updated to point to the correct file paths within the Colab file system.
- An

Ultralytics YOLOv8-seg model was fine-tuned for approximately **100 epochs** using the custom annotated dataset.

5. **Inference, Evaluation, and Demo**:
    - The trained model was used to run inference on the test images. The results, including bounding boxes, class names, and confidence scores, were saved.
    - The model's performance was evaluated using standard metrics such as a confusion matrix and a PR curve.
    - Finally, the trained model was integrated with

Byte Track to track objects in a video. A simple Python script was built to run the tracking pipeline and export the results to a

results.json file containing object IDs, bounding boxes, and frame numbers.

---

3. Problems Faced and Resolutions

Throughout the process, several common challenges were encountered and successfully resolved:

- **Problem**: ModuleNotFoundError: No module named 'ultralytics'.
    - **Resolution**: The ultralytics library was not installed in the Colab environment. The issue was resolved by running !pip install ultralytics in a separate code cell before attempting to import the library.
- **Problem**: FileNotFoundError: Dataset '...' images not found.
    - **Resolution**: The training script was failing because the relative paths in the data.yaml file were incorrect for the Colab environment. The issue was fixed by writing a Python script to dynamically update the data.yaml file with the correct absolute paths to the image and label folders in Colab.
- **Problem**: The exported zip file from Labellerr was missing image files, containing only labels.
    - **Resolution**: This was a configuration issue during the export process. The problem was circumvented by manually zipping the images and labels into a single archive with the correct folder structure on a local computer. This new zip file was then uploaded to Google Drive for use in the Colab notebook.

---

4. A Guide to Building an Object Tracker with Labellerr and YOLO

This guide provides a step-by-step walkthrough for a fellow developer to build a similar end-to-end pipeline.

Step 1: Data Collection & Preparation

- **Collect Images**: Gather a minimum of 100 images of vehicles and pedestrians from permissibly licensed sources.
- **Organize Data**: Create a folder structure with subfolders for images and labels.
- **Labellerr Setup**: Create a training project on the Labellerr platform and upload your images.

Step 2: Annotation

- **Configure Classes**: In Labellerr, define "vehicle" and "pedestrian" as object classes, and set the annotation type to **Polygon**.
- **Annotate Images**: Use the polygon tool to manually draw segmentation masks around each object in the training images.
- **Review and Accept**: Once annotations are complete, use the "Review" tab to accept all files.

Step 3: Export and Training Setup

- **Export Data**: From the Labellerr "Export" tab, create an export of your annotated data. Select a YOLO-compatible format and ensure you choose to include the source images in the export.
- **Google Colab**: Upload your exported zip file to Google Drive and open a new Colab notebook.

Step 4: Model Training and Inference

- **Unzip Data**: In Colab, mount your Google Drive and unzip your dataset.
- **Train Model**: Use the ultralytics library to train a yolov8n-seg.pt model on your dataset for 100 epochs.

Step 5: Video Tracking

- **Get a Video**: Upload a video file of your choice to your Google Drive.
- **Run Tracking Script**: Use a Python script to run your trained model on the video with the bytetrack.yaml tracker. The script should iterate through each frame, track objects, and save the results to a

results.json file.

Step 6: Deliverables

- **GitHub**: Upload all your code, including the Colab notebook and the sources.md file, to a GitHub repository.
- **Documentation**: This report, detailing your journey, problems, and solutions, is a final deliverable.
- **Demo**: Create a live demo of your video tracking application for users to experience.

Final Summary

This project successfully demonstrates a complete computer vision pipeline, highlighting the importance of data quality, proper file management, and debugging. The use of the Labellerr platform provided a streamlined process for annotation and export, while Ultralytics and Byte Track enabled the training and deployment of a functional object tracker. The journey from a raw dataset to a video tracking demo provides a solid foundation in machine learning development.