Georgia Institute of Technology

CE 8900 - SPECIAL PROBLEMS

ROBOTICS & INTELLIGENT CONSTRUCTION AUTOMATION LAB

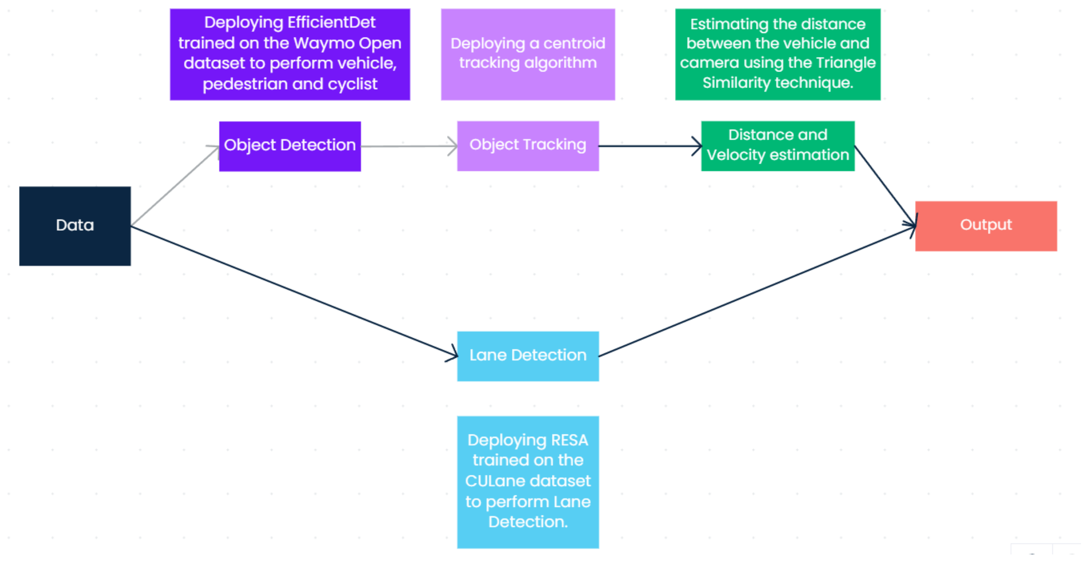
VEHICLE PROXIMITY DETECTION SYSTEM

Nimisha Pabbichetty

April 2023

1 System Description

The aim of this project was to develop a system that issues alerts and warnings for construction worker safety on the field. It needs to be capable of detecting vehicles approaching the construction zone, identify the vehicles of interest, calculate an estimated distance and velocity and issue a warning if either of the values indicate danger to the workers. The below flowchart describes the design of the pipeline.



Deploying EfficientDet trained on the Waymo Open dataset to perform vehicle, pedestrian and cyclist
Deploying a centroid tracking algorithm
Estimating the distance between the vehicle and camera using the Triangle Similarity technique.
Object Detection
Object Tracking
Distance and Velocity estimation
Data
Output
Lane Detection
Deploying RESA trained on the CULane dataset to perform Lane Detection.

Figure 1: Flowchart of the proposed system

1.1 Object Detection

To design such a system, the first block of the pipeline needs to be an object detection model that can detect the vehicles in the camera feed. An object detection model is a type of computer vision model that is designed to locate and identify objects within an image or video. The model works by analyzing the pixels of an image or video frame and attempting to classify each pixel as part of an object or background. The objects of interest that have been identified in the image or video are marked using a bounding box around their estimated position. To train and deploy such a model, the first step is to identify the right dataset.

1.1.1 Vehicle Detection Dataset Literature Survey

The summary of the literature survey is below. Based on the recorded key insights, the 'Waymo open dataset - 2D camera labels' is used.[1]

Dataset	Number of Images	Key highlights
Kaggle Car object detection dataset	Training - 1001, Testing - 175	Images of vehicles driving in various views
Vehicles - openImages dataset (Roboflow)	627 images	Close up images of 5 classes - car, bus, motorcycle, ambulance and truck
Boxy dataset	200,000	Images are taken from a vehicle on the move, catered towards identifying the rear and sides of the vehicle (3D boxes for side).
KITTI 2D benchmark	14000	Contains variety of views, classes include car, van, pedestrian and cyclist.
TRANCOS	1244	Traffic scenes, higher angle view, features a lot of cars in each shot (42k annotations)
Stanford car dataset	19618	Closeups of different cars
Waymo open dataset - 2D camera labels	800+ video segments	Classes - vehicles, pedestrians and cyclists
PKU vehicleID	200,000+	Closeups of different cars
Vehicle 1M	900,000	Closeups of different cars
MOCS	41668	Used for detecting moving objects in construction sites. 13 classes such as worker, tower crane etc
Veri	50000	Unconstrained surveillance footage. 12 classes include different makes of cars.
VRIC	60430	Surveillance view but consists of more real-time like images (near and far angles, blurred images of vehicles, day and night etc)
UA-DETRAC	140,000	Traffic surveillance footage
GRAM Traffic		Benchmark for testing models

Table 1: Summary of Vehicle Detection Datasets

1.1.2 Detection Model

The pre-trained model that was deployed is the EfficientDet model by a team from Google [2]. The backbone of this architecture is the EfficientNet convolutional neural network. In addition to this they also implemented a bi-directional feature network (BiFPN) enhanced with fast normalisation which enables fast feature fusion. Compound scaling is also implemented where a compound coefficient to jointly scale up all the dimensions of backbone, BiFPN, class prediction network and resolution. The input resolution is set to 1024 * 1024. The model is implemented using Tensorflow's Object Detection API.

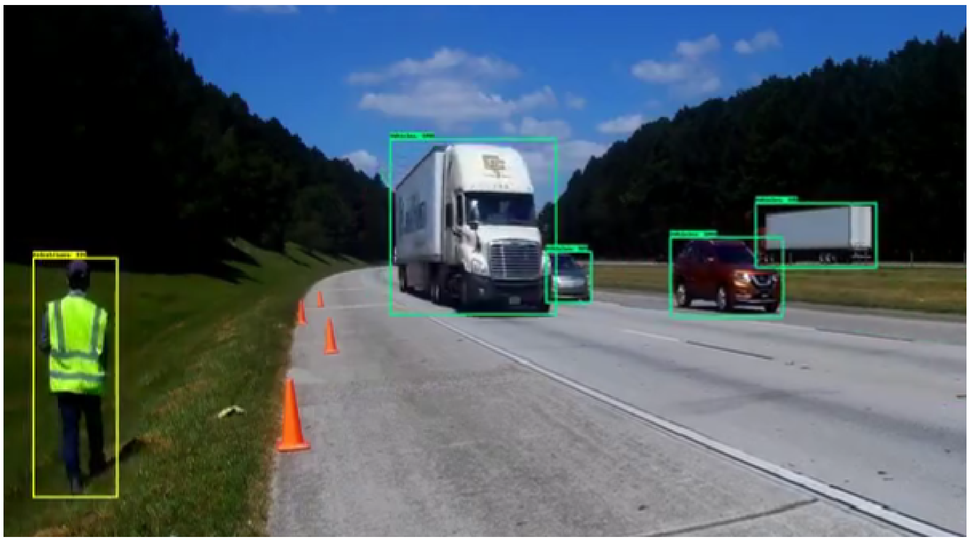


Figure 2: Output of the Object Detection model

1.2 Object Tracking

The goal of object tracking is to determine the bounding boxes and identities of objects in videos. It takes a series of initial object detections, creates a visual model for the objects, and then tracks them as they move about in a video. Object tracking also allows us to assign a unique ID to each tracked object, allowing us to count the number of unique objects in a video. Centroid tracking algorithm was deployed. Centroid tracker algorithms are all about tracking the coordinates by defining some threshold values to be called as same (centroid) points and then the Euclidean distance of these centroids of all detected objects in subsequent frames is calculated. The main idea of using a centroid is that the same object will be moved a minimum distance compared to others points in the subsequent frames. Therefore, whichever centroid has the minimum distance pair will be assigned with the same ID as the previous centroid with the least euclidean distance. In the figure below, the unique IDs being assigned to each vehicle and the centroids (green dots) can be seen.[3]

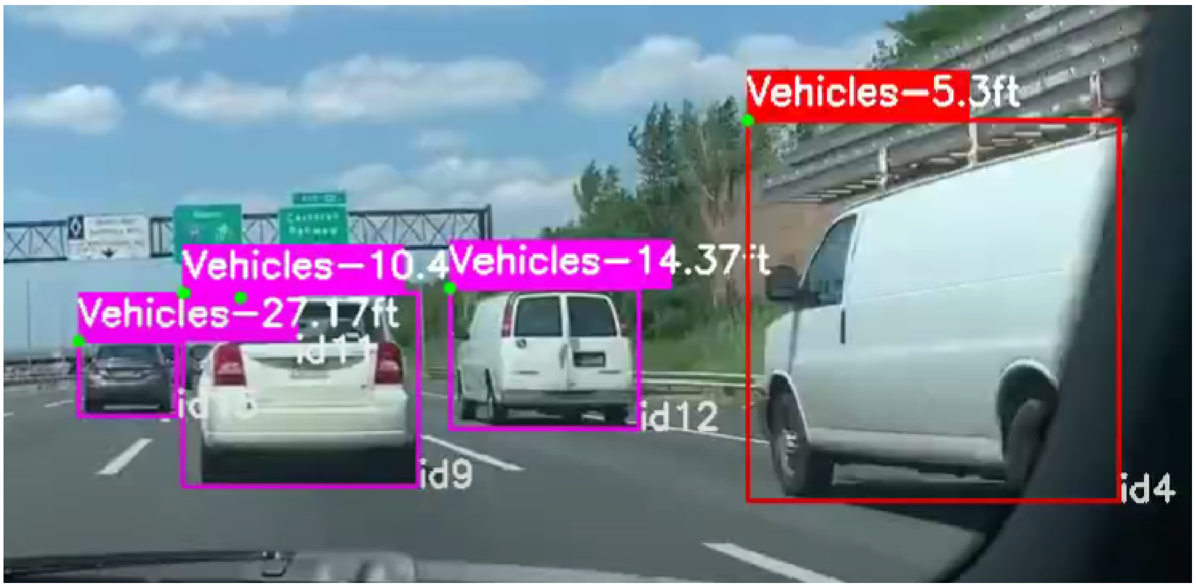


Figure 3: Output of the Object Detection and Tracking models

1.3 Lane Detection

The second block of the pipeline is the Lane Detection Network. Lane detection is a computer vision technique that involves detecting and tracking the lane markings on a road. To train and deploy such a model, the first step is to identify the right dataset.

1.3.1 Literature Survey on Lane Detection datasets

Dataset Name	Images	Key Highlights
CULane	134000	Several modes to the dataset - 1 normal and 8 challenging categories.
TUSimple	6408	Has images from different weather conditions.
LLAMAS	100,000	Lane markers were generated on Lidar
CurveLanes	150,000	Catered towards difficult scenarios like curves and multi-lanes.

Table 2: Summary of Lane Detection Datasets

1.3.2 Model Analysis

Since a smaller model could perform better on a model, in interest of keeping the pipeline from becoming too computationally intensive, the TUSimple dataset was chosen [4]. The following pre-trained models were tested on the TUSimple dataset's test set. The Condlane model alone was tested on the CULane dataset (this model ranks higher than the others on the Leaderboard for that dataset) but when applied to our data, it fails to perform well. The model is implemented using PyTorch.

Model Name	Accuracy
LaneAtt (ResNet 18 backbone)	94.62
LaneAtt (ResNet 34 backbone)	94.99
RESA (ResNet 18 backbone)	96.72
RESA (ResNet 34 backbone)	96.85
SCNN (ResNet 18 backbone)	96.04
UFLD (ResNet 18 backbone)	95.32

Table 3: Comparison of Accuracy of different models

In accordance with these results, the RESA model with the ResNet 34 backbone is used in the pipeline to detect lanes.[5]

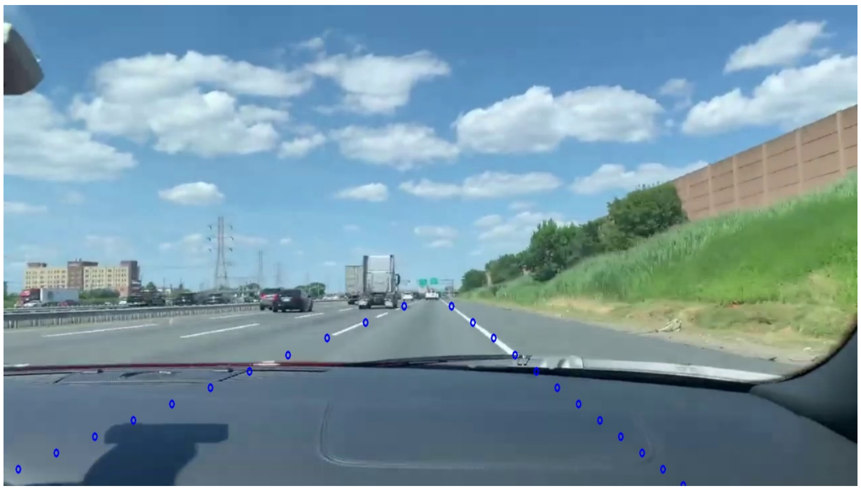


Figure 4: Output of the Lane Detection model

1.4 Post-Processing

The model is capable of detecting 3 classes - vehicles, pedestrians and cyclists. Each class is visualised with different coloured bounding boxes. Monocular depth estimation is performed using height estimation based on projection using this formula:

$$\tan(\alpha) = \frac{w}{d} = \frac{p}{f} \tag{1}$$

The velocity is estimated by comparing the distance across frames. If the velocity is higher than a set threshold (80 km/hr) then the bounding box changed colour and a warning is issued in the terminal. The velocity of the vehicle in the center of the screen is monitored (vehicle of interest in the deployment scenario is assumed to be in the center of the feed). The vehicle of interest is visualised in a different colour (green as opposed to pink which is the default colour for all vehicles).

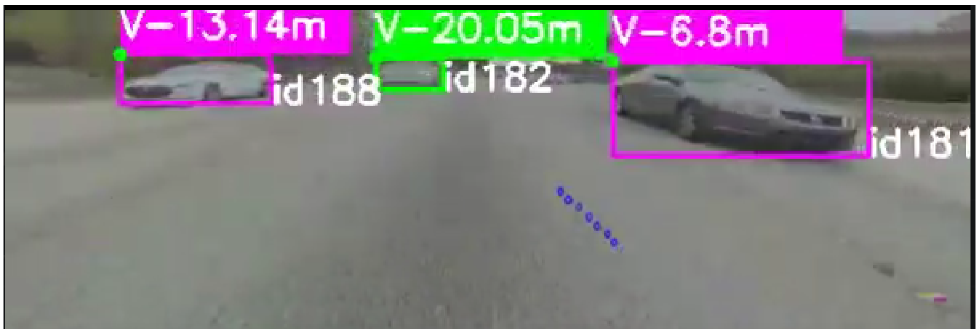


Figure 5: Output of the final pipeline

2 Github Reference Repositories

1. https://github.com/raz4/waymo-object-detection
2. https://github.com/adipandas/multi-object-tracker
3. https://github.com/Turoad/lanedet

References

[1] P. Sun *et al.*, *Scalability in perception for autonomous driving: Waymo open dataset*, 2020. arXiv: 1912.04838 [cs.CV].

[2] M. Tan, R. Pang, and Q. V. Le, *Efficientdet: Scalable and efficient object detection*, 2020. arXiv: 1911.09070 [cs.CV].

[3] A. M. Deshpande, *Multi-object trackers in python*, version v1.0.0, Jul. 2020. DOI: 10.5281/zenodo.3951169. [Online]. Available: https://doi.org/10.5281/zenodo.3951169.

[4] *Tusimple*, https://github.com/TuSimple/tusimple-benchmark, 2017.

[5] T. Zheng *et al.*, "Resa: Recurrent feature-shift aggregator for lane detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 3547–3554.