Car Counting Project

Brian Li, Taeyoon Kim, Alex D'Souza

June 4, 2024

1 Introduction

In an era where urbanization and population growth are accelerating, efficient transportation systems have become crucial for the sustainability and livability of cities. As the number of vehicles on the roads continues to rise, managing traffic congestion, improving road safety, and planning future infrastructure developments are pressing challenges. Accurate and real-time data on vehicle movement is essential to address these issues effectively. Our Car Counting Project aims to provide a robust solution for real-time vehicle detection and counting, offering critical insights for transportation planning and traffic management.

2 Motivation

2.1 Traffic Flow Analysis

Traffic flows can be analyzed if we have an accurate estimation of the number of vehicles within a certain image frame. This data allows for the identification of peak traffic hours, understanding of traffic patterns, and optimization of traffic signal timings to reduce congestion and improve overall traffic efficiency.

2.2 Parking Management

Parking management can also be tracked using a camera to analyze how many cars enter and leave a designated parking lot, and calculate the amount of vacant slots. This information can be used to guide drivers to available parking spaces, reduce the time spent searching for parking, and optimize the use of parking resources.

2.3 Transportation Planning

Transportation planning and infrastructure management can be planned based on traffic volumes. It can be used to prioritize heavily used roads, based on traffic analysis from a well-defined car counting program. Accurate traffic data ensures that infrastructure investments are made where they are needed most, leading to improved road conditions and better allocation of public funds.

2.4 Automated Cars

Automated cars need to be able to quickly and accurately detect cars in front and behind them. Being able to detect trucks and buses can also be a benefit, in order to pursue more cautious behaviors. This capability enhances the safety and reliability of autonomous vehicles, enabling them to make informed decisions in real time and navigate complex traffic scenarios more effectively.

3 Methods

Our dataset contains images of highways with labels of motorcycles, Sedans, SUVs, trucks, vans, and pickups. In our vehicle detection algorithms, we ignored all motorcycles and summed up the instances of Sedans, SUVs, vans, and pickups to be the number of cars.

We initially attempted to implement vehicle detection using techniques purely in computer vision. Many of these algorithms have limitations and require specific conditions to work correctly. Among them, we tried edge detection and background subtraction algorithms.

3.1 Edge Detection

- Gray → Gaussian blur (remove noises) → Get edges (Canny Edge) → Find contour

We used the Canny Edge detection algorithm to identify the boundaries of an image. After detecting edges, we use a contour detector to find continuous edges that form shapes, which might be vehicles. Vehicle detection using edge detection performed poorly because of the complexity of vehicle shapes, ambiguity in edge formation, and inadequate contour formation. Edge detection algorithm struggles to accurately delineate vehicle boundaries, which leads to inaccurate contours.

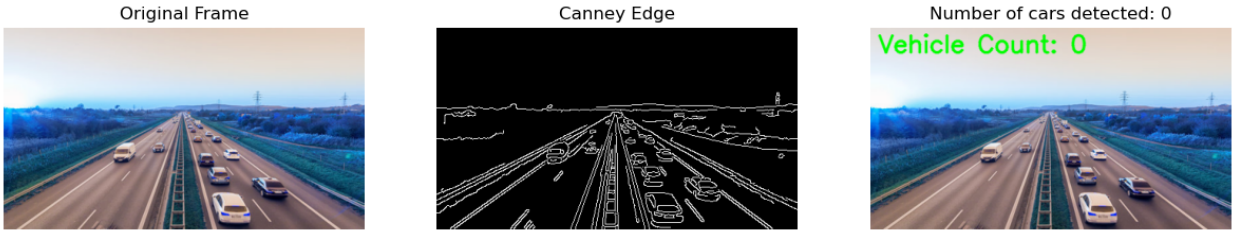


Figure 1: Canny Edge

3.2 Background Subtraction

- Gray → Gaussian blur (remove noises) → Threshold to remove shadows → Apply some morphological operations to make sure you have a good mask (getStructuringElement, erode, dilate) → Find contour

Background subtraction can indeed be limited in its effectiveness for vehicle detection in images. One major challenge arises from the need for a consistent and accurate background model. Background subtraction techniques rely on the assumption that the scene's background remains relatively static over time. However, in real-world scenarios, backgrounds can vary due to changes in lighting conditions, moving objects, or environmental factors.

In the context of vehicle detection from a single picture, background subtraction becomes especially challenging. Unlike in video surveillance applications where a background model can be continuously updated, analyzing a single image lacks the temporal context necessary for accurate background modeling. Without prior knowledge of the background or a reference image captured from the same angle without any vehicles present (often referred to as an empty road picture), background subtraction may yield inaccurate results.

Figure 2: Background Subtraction

Background subtraction and edge detection are fundamental techniques used for vehicle detection in images, each with its own set of challenges and limitations. Vehicle detection using edge detection performs poorly due to the complexity of vehicle shapes, ambiguity in edge formation, and inadequate contour formation. Similarly, background subtraction technique is significantly limited when applied to single images due to the need for a consistent and accurate background model. Without an updated background model or a reference image without vehicles, background subtraction often yields inaccurate results. In contrast, cv2.createBackgroundSubtractorMOG(), a Gaussian Mixture-based Background/Foreground Segmentation Algorithm in CV2, performs well with video data by continuously updating the background model.

3.3 Detection using ML

To find a better vehicle detector, we tried machine learning-based methods like You-Only-Look-Once (YOLO). YOLO is a popular object detection algorithm known for its speed and accuracy. The architecture of YOLO involves dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell simultaneously. This allows YOLO to detect multiple objects in an image in real-time.

Initially, we chose to use YOLOv3. The YOLOv3 object detection model runs a deep learning convolutional neural network (CNN) on an input image to produce network predictions from multiple feature maps. Then, the object detector gathers and decodes predictions to generate the bounding boxes. From these bounding boxes, YOLOv3 uses anchor boxes, where the model predicts these three attributes for each anchor box:

- Intersection over union (IoU) — Predicts the objectness score of each anchor box.
- Anchor box offsets — Refine the anchor box position.
- Class probability — Predicts the class label assigned to each anchor box.

We compared the performance of YOLOv3 to other YOLO models, such as YOLOv2 and YOLOv4. From version two to four, improvements to object detection were made in both speed and accuracy through various machine learning techniques and changes in architecture. Since these are pretrained models, we combined all the images into one directory and tested on that.

4 Experimental Results

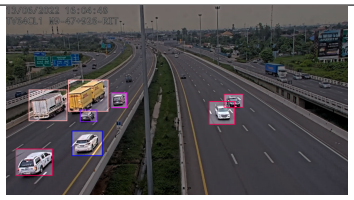
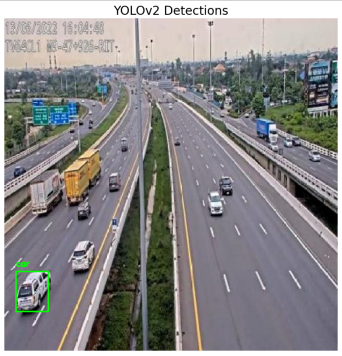
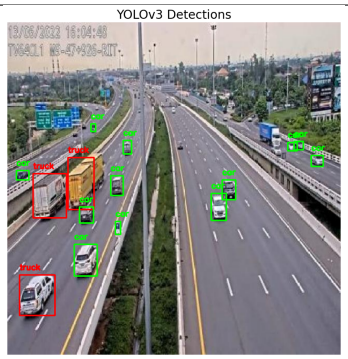
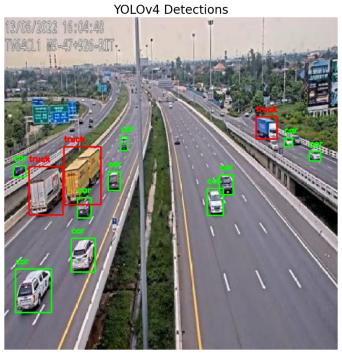
Ground Truth or YOLO model	Detection	Label Instances	Accuracy	Precision, Recall, and F1-Score
Ground Truth		6 cars 0 buses 2 trucks	N/A	N/A
YOLO v2		1 car 0 buses 0 trucks	0% for car 100% for buses 0% for trucks	Precision = Recall = F1-Score =
YOLO v3		10 cars 0 buses 3 trucks	0% for car 100% for buses 0% for trucks	Precision = Recall = F1-Score =
YOLO v4		12 cars 0 buses 3 trucks	0% for car 100% for buses 0% for trucks	Precision = Recall = F1-Score =

Table 1: Comparison of YOLO Models

Model	Car Accuracy (%)	Bus Accuracy (%)	Truck Accuracy (%)
YOLOv2	18.11	78.19	2.06
YOLOv3	34.57	62.55	2.06
YOLOv4	46.50	49.79	1.65

Table 2: Performance of YOLO Models on the Test Set

5 References

- Canny Edge: https://medium.com/neurosapiens/edge-detection-65d57c1af118
- Background Subtraction: https://www.overleaf.com/project/664bebf6ef8f9e89f32fe121
- Reference paper on YOLOv3: https://arxiv.org/abs/1804.02767
- YOLOv2, v3, and v4 models: https://github.com/kiyoshiiriemon/yolov4_darknet?tab=readme-ov-file#yolo-v4-v3-and-v2-for-windows-and-linux
- Information about YOLO: https://pjreddie.com/darknet/yolo/
- Dataset: https://universe.roboflow.com/cc-pintel/car-counting

6 Contributions