Speed Estimation on a Highway

Applied Research Thesis

2_DLBAIPCV01_Project_Computer_Vision

Computer Vision

25.06.2024

Author: Raphael Braunstein

Matriculation number: 9226563

Tutor: Konstantinos Amplianitis

Table of Contents

1. Introduction

1.1. Problem Definition and Initial Conditions

The modern highway system is a crucial infrastructure component for transportation, yet it faces significant challenges related to traffic management and safety. One critical issue is the ability to monitor and control vehicles to reduce accidents and improve traffic flow. Traditional methods, such as speed cameras and police patrols, are often resource-intensive and not always effective in real-time speed monitoring and enforcement. (Federal Highway Administration, 2019).

The problem at hand is the need for an efficient, accurate, and real-time speed tracking system for highways. Such a system should be able to detect vehicles, estimate their speeds, and trigger alerts when speed limits are exceeded. This capability is essential for enhancing road safety, reducing accidents, and improving overall traffic management.

1.2. Project Objectives

The primary objective of this project is to develop a speed tracking system for highways that can:

- Detect cars accurately in real-time.
- Estimate the speed of each detected car.
- Trigger alerts for cars that exceed a predefined speed limit.

The system should leverage state-of-the-art computer vision algorithms and tools to achieve high accuracy and real-time performance.

1.3. Preparatory Work

Preparatory work for this project involved recording video footage of a predominantly straight section of a highway, selected to facilitate easier processing. Additionally, measurements of this highway section were obtained using Google Maps for further transformation of the polygon zone. This aspect will be thoroughly explained in the section discussing the speed estimation methodology. The original unprocessed video can be seen here:

https://drive.google.com/file/d/1h4SoOfiXCuHfLQn32L96kg-11-35TVL7/view?usp=sharing

This project report details the development of a speed estimation tool using YOLOv8 in combination with Byte Track. It also includes a comparison of three state-of-the-art approaches to implement such a tool and explains the methodology employed for implementing the speed estimation.

2. Main Part

2.1. Comparison of three state of the art approaches

Given the numerous algorithms available for object detection and tracking, this report will compare the three most used algorithms YOLO, SSD and Faster R-CNN. Initially, the three algorithms will be briefly described, highlighting their pros and cons in terms of accuracy and speed. Subsequently, the decision behind the selected algorithm, based on the specific requirements of this project, will be explained.

2.1.1. YOLO (You ONLY LOOK Once)

YOLO (You Only Look Once) is a real-time object detection system that frames object detection as a single regression problem, directly mapping image pixels to bounding box coordinates and class probabilities. Unlike traditional object detection methods that apply classifiers or localizers to various regions of an image, YOLO processes the entire image with a single neural network, making predictions simultaneously for multiple bounding boxes and class probabilities (Redmon et al., 2016). This unified approach significantly enhances processing speed, enabling real-time performance.

YOLO is incredibly fast because it processes the image in one pass (one look), making it highly suitable for real-time applications. However, despite its speed, YOLO sometimes trades off accuracy, particularly when detecting smaller objects, leading to lower localization accuracy compared to other methods (Redmon et al., 2016). The high frame rate of YOLO allows for frequent updates on car positions, providing more data points for speed calculation, which is a significant advantage. On the downside, the lower localization accuracy can result in errors in distance measurements, affecting the overall speed accuracy (Redmon, n.d.).

2.1.2. SSD (Singe Shot MultiBox Detector)

SSD (Single Shot MultiBox Detector) is a more balanced approach that achieves a good compromise between speed and accuracy in object detection. Unlike traditional methods that may require multiple stages of processing, SSD uses a single deep neural network to detect objects in images. This network simultaneously predicts bounding boxes and object classes at multiple scales, which allows it to handle objects of varying sizes effectively (Liu et al., 2016).

One of the key features of SSD is its use of feature maps at different layers of the neural network to detect objects at different scales. This multi-scale approach ensures that both large and small objects can be detected with reasonable accuracy. SSD's architecture allows for faster processing compared to methods like Faster R-CNN, while still maintaining a high level of accuracy, making it suitable for real-time applications (Liu et al., 2016).

SSD is faster than Faster R-CNN and generally more accurate than YOLO, particularly in detecting smaller objects. While it strikes a good balance between speed and accuracy, it may not be as fast as YOLO in some real-time applications. The better accuracy in detecting smaller objects and improved localization accuracy enhance the precision of speed calculations. However, the slightly lower speed compared to YOLO might reduce the number of data points in real-time applications, though it remains generally effective (Liu et al., 2016).

2.1.3. Faster R-CNN (Region-based Convolutional Neural Network)

Faster R-CNN achieves high object detection accuracy through a two-stage process. First, a Region Proposal Network (RPN) identifies candidate regions in an image likely to contain objects. These high-quality proposals are crucial for subsequent steps. In the second stage, a network refines these regions, classifying objects and adjusting bounding boxes for precise localization (Ren et al., 2015). This approach balances coarse and fine-grained detection, ensuring accurate results.

However, Faster R-CNN's complex pipeline makes it slower than single-stage detectors like YOLO and SSD. This limits its real-time applicability but enhances performance in tasks requiring detailed object analysis, such as automated inspection and precise object localization in autonomous systems (Ren et al., 2015).

Faster R-CNN excels in accuracy, particularly in precise object localization and classification, establishing itself as one of the foremost algorithms for object detection (Ren et al., 2015). However, its comparative slowness compared to YOLO and SSD may pose limitations in real-time applications. The high accuracy in object localization and detection significantly enhances the precision of speed calculations, minimizing errors. Nevertheless, the slower processing speed may reduce the frequency of updates, potentially impacting the effectiveness of real-time speed tracking (Ren et al., 2015).

2.1.4. Final decision

YOLO's combination of speed, accuracy, and efficiency makes it a highly effective choice for detecting and tracking cars on highways, enabling robust real-time performance essential for modern traffic surveillance and management systems.

2.2. Technologies used in this Project

This project leverages several advanced technologies to achieve robust car detection and tracking on highways. The primary technologies employed include:

Yolov8x: YOLOv8x serves as the core machine learning model for real-time object detection. This model is renowned for its speed and accuracy in identifying objects within images, making it ideal for dynamic environments such as highway traffic monitoring.

ByteTrack by Supervision: ByteTrack is utilized for object tracking, enhancing the continuity and reliability of car tracking across consecutive frames. By employing advanced algorithms for object motion prediction and trajectory estimation, ByteTrack ensures robust tracking performance.

Polygon Zone Transformation: The project incorporates polygon zone transformation, facilitated by tools such as Google Maps and the roboflow’s polygon zone tool, to define specific areas of interest along the highway. This transformation aids in spatially constraining the detection and tracking processes, focusing computational resources on relevant regions and improving efficiency.

2.3. Implementation of the project

The implementation of the car detection and tracking tool on a highway section involved several key steps, integrating YOLOv8 for object detection, ByteTrack with Supervision for tracking, and polygon zones for spatial constraint.

2.3.1. Object Detection

To begin, necessary libraries and tools were installed including YOLO by ultralytics, Supervision's Bytetrack, and OpenCV for image processing.

Initially, several instances, such as the YOLOv8 model and the frame generator, are initialized to process frames from the input video. The inference is then run on each frame within a for loop, converting the results into supervision detection objects. The Bounding Box Annotator is used to draw boxes around the detected objects on the frame. A copy of the current frame is created and updated with the results from the Bounding Box Annotator's annotate method. Finally, OpenCV is used to display the annotated frame on the screen.

2.3.2. Multi-Object Tracking

Object detection alone is insufficient for accurate speed estimation. To calculate the distance travelled by each car, continuous tracking of each vehicle's position across multiple frames is essential. This project utilizes ByteTrack, part of the Supervision package, to achieve reliable object tracking. Object tracking is crucial for speed estimation because it maintains the identity of each car across consecutive frames, allowing us to measure the displacement over time. Without tracking, it would be impossible to determine whether detected objects in successive frames correspond to the same vehicle, thus hindering accurate distance and speed calculations.

In this implementation, an instance of ByteTrack is created with the necessary information, such as the frame rate, to facilitate accurate tracking. During the processing loop, ByteTrack's update_with_detections method is called for each frame to update the tracking information. This method ensures that the detected objects are consistently tracked across frames, enabling precise calculation of each car's speed.

2.3.3. Filtering Detections with Polygon Zone

To eliminate unwanted detections, particularly those flickering in the background as cars become smaller, Supervision's Polygon Zones Filtering feature is employed. This approach focuses object detection on a specific area within the video frames, improving accuracy and efficiency.

First, the exact coordinates for the polygon zone within the frame are needed. These coordinates will form the corners of the polygon zone. The RoboFlow PolygonZone tool (https://roboflow.github.io/polygonzone/) can be used to identify these points accurately. By clicking on the desired spots in the frame, the tool provides a numpy array of these coordinates.

In the code, the polygon zone is initialized with this numpy array and the frame resolution information. During the detection process, this polygon zone acts as a filter. After detecting objects, but before tracking them, a detection filter is applied to determine whether each detection falls inside or outside the polygon zone. This ensures that only relevant detections within the defined area are considered for tracking, thus enhancing the overall performance of the system.

2.3.4. Perspective transformation and Speed Estimation

Speed measures how fast an object moves, calculated by dividing the distance travelled by the time taken. Simply counting the pixels covered by cars in a specified time can result in inaccurate speed measurements due to the camera's angle and distortion. Objects farther from the camera cover fewer pixels over time compared to those closer to the camera, leading to perspective distortion.

To accurately estimate a vehicle's speed, raw image coordinates must be transformed into actual road coordinates. This transformation corrects perspective distortion using OpenCV. A trapezoidal polygon zone is defined as the region of interest and then converted into a rectangular shape representing the actual dimensions of the road, which measures 35 meters in width and approximately 230 meters in length, as determined using Google Maps' distance measurement tool.

To achieve this transformation, coordinate systems are converted. The highway's trapezoidal shape may extend beyond the frame, with coordinates A (740, 420), B (1150, 420), C (2100, 830), and D (-300, 830). These coordinates are transformed into a rectangle with coordinates A' (0,0), B' (34,0), C' (34,229), and D' (0,229), maintaining the true dimensions of the road section. The coordinates are specified as width and length minus one because array indexing starts at 0.

A new class, ViewTransformer, will be implemented to handle the perspective transformation. The constructor of this class takes the source and target regions of interest and passes them to OpenCV's

getPerspectiveTransform function. OpenCV's Perspective Transform expects points to be defined in three-dimensional space rather than on a two-dimensional plane. Therefore, additional data is added to our points as a placeholder for the third dimension, allowing the points to undergo perspective transformation. Once the transformation is complete, the placeholder data for the third dimension is removed.

Subsequently, OpenCV will be used to convert the bounding boxes into a list of points. These points are then transformed using an instance of the ViewTransformer, converting coordinates from the source region of interest to the target region of interest.

The transformed coordinates can now be used to calculate the speed of moving vehicles. With the vehicles moving in a rectangular frame, the positions are aligned such that the x-coordinates remain relatively constant, and only the y-coordinates change significantly. To determine the distance travelled by a vehicle over the last second, the y-coordinate at the previous point is subtracted from the y-coordinate at the current point. This difference represents the vehicle's speed in meters per second. To convert this speed to kilometres per hour, the result is multiplied by 3.6.

In the implementation, the coordinates of each frame are stored in a dictionary. Within this dictionary, a deque is utilized to retain the car's coordinates over the last second, at a rate of 25 updates per second. Subsequently, a loop iterates over each tracker ID and updates the current y-coordinate value in this dictionary.

To calculate the speed, the difference between the first and last elements in the coordinates deque is computed and multiplied by 3.6. This speed value is then displayed alongside the cars using labels.

2.3.5. Alerting

After calculating the speed, vehicle speed alerts are implemented using simple if clauses. Initially, bounding boxes are displayed in white. Upon speed calculation, the bounding box colour changes: green if the car travels under 100 km/h, yellow if it exceeds 100 km/h, and red if it exceeds 140 km/h, indicating excessive speed.

2.4. Difficulties during the project

Throughout the project, a significant challenge arose while transforming the trapezoidal shape into a rectangular shape for speed calculation. The section of the highway exhibits a slight curve to the right, making the trapezoidal shape less than ideal for the polygon zone. This curvature causes a compression during transformation, particularly affecting vehicles further from the camera. As a result, these vehicles appeared to travel shorter distances due to the distortion, potentially leading to inaccurate speed calculations.

To mitigate this distortion, adjustments were made by stretching the right top corner of the trapezoidal shape slightly outward. This modification aimed to minimize the compression effect during transformation, thereby improving the accuracy of speed calculations. While the solution remains imperfect, these adjustments significantly enhanced performance in handling the curvature of the highway section.

2.5. Result

As a result of this project, a comprehensive vehicle tracking and speed estimation system has been developed, successfully fulfilling all required functionalities including car detection, real-time speed estimation, and alert triggering for exceeding predefined speed limits. The system performs adequately overall; however, there is room for improvement. Future enhancements could involve using a more suitable shape than the trapezoidal region of interest or choosing a highway section with fewer curves for the input video to mitigate issues related to curvature. The resulting video can be seen here:

https://drive.google.com/file/d/1AamTJwAR2mR0gcQXtlRov6-Bk1Ig2HGP/view?usp=sharing

3. Conclusion

This project has successfully developed a vehicle tracking and speed estimation system that meets the required functionalities of detecting cars, estimating their speed over time, and triggering alerts when a car exceeds a predefined speed limit. The system integrates advanced technologies like YOLOv8 for object detection, ByteTrack for object tracking, and polygon zone transformation to enhance accuracy. Overall, the project demonstrates a robust approach to monitoring vehicular speed on highways, with satisfactory performance in real-world conditions.

From the results and findings presented in the main sections, several key conclusions can be drawn. Firstly, the integration of YOLOv8 and ByteTrack has proven to be effective in maintaining a high detection and tracking accuracy while operating at real-time speeds. The system's ability to transform image coordinates to actual road coordinates using polygon zones significantly reduces perspective-related distortions, thereby improving the precision of speed estimation. However, the project also highlighted areas for potential improvement, particularly in handling road curvature and optimizing the polygon zone transformation for more accurate speed calculations.

The project has provided valuable hands-on experience with state-of-the-art machine learning models and computer vision techniques, which are highly relevant in the field of intelligent transportation systems. The challenges encountered and the solutions devised have deepened my understanding of practical issues in object detection and tracking, as well as the importance of data preprocessing and transformation in achieving accurate results.

In conclusion, the vehicle tracking, and speed estimation system developed in this project serves as a foundational step towards more advanced applications in traffic monitoring and management. While the current implementation meets the project's objectives, future work will focus on refining the system to handle more complex scenarios and improve overall accuracy.

4. Bibliography

- Federal Highway Administration. (2019). Traffic monitoring guide. U.S. Department of Transportation. https://www.fhwa.dot.gov/policyinformation/tmguide/tmg_fhwa_pl_17_003.pdf

- Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. Proceedings of the IEEE International Conference on Computer Vision (ICCV). https://doi.org/10.1109/ICCV.2017.324

- Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T. K. (2014). Multiple object tracking: A literature review. arXiv preprint arXiv:1409.7618. https://arxiv.org/abs/1409.7618

- Redmon, J. (n.d.). YOLO: Real-time object detection. YOLO. https://pjreddie.com/darknet/yolo/

- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788. https://doi.org/10.1109/CVPR.2016.91

- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems (NeurIPS). https://arxiv.org/abs/1506.01497

- Zhang, Y., Wang, C., Wang, X., Zeng, W., & Liu, W. (2020). FairMOT: On the fairness of detection and re-identification in multiple object tracking. International Journal of Computer Vision (IJCV). https://doi.org/10.1007/s11263-021-01407-8

- Zhou, X., Wang, D., & Krähenbühl, P. (2020). Tracking objects as points. Proceedings of the European Conference on Computer Vision (ECCV). https://doi.org/10.1007/978-3-030-58565-5_45