Automated Multiple License Plate Recognition (AMLPR) for Dash Cameras

David Peterson

Problem

Over the last decade, an average of 700,000 hit-and-runs (17,000 fatal) were reported in the United States alone every year, causing millions of dollars in damage [1]. Hit-and-runs can occur when a vehicle is being driven or is parked, but the offender is rarely apprehended because their license plate is not acquired. All conventional dash cameras can record driving footage; however, a select few can automatically activate if an impact is detected while parked. Unfortunately, these videos can take a long time to process, have low quality, and have visibility hindrances such as weather and brightness. Emerging technologies allow data to be transmitted in real-time from a dash camera to an application on a mobile device in an attempt to alert an end-user of a hit-and-run remotely. However, the transmission of video files is time-consuming, requires more processing power, and drains the small battery in the dash camera. **Integrated AMLPR in a dash camera could automatically identify relevant data and reduce transmitted data to images, conserving memory and power usage while minimizing transmission times to the end-user and allowing the full recorded videos to be reserved for future review.** By merging dynamic AMLPR software into a conventional dash camera, end-users could actively use artificial intelligence to identify, recognize, and save relevant license plate numbers when an accident occurs, ultimately facilitating the offender's capture because every second matters.

Approach

My approach to creating a fully functioning AMLPR software that efficiently and accurately identifies and recognizes multiple license plate numbers simultaneously is to split the problem into incremental sub-problems. The core of automated license plate recognition software in machine learning is detecting the location of a license plate in an image or video frame, extracting the characters from the license plate, and performing an optical character recognition (OCR) algorithm to classify the characters into specific letters and numerical digits [2]. To facilitate making progress towards the project goal, I plan to obtain and use a United States license plate dataset from UCSD and divide it into the following categories: images with a single license plate location for labels (bounding box coordinates) and images with a single license plate number for labels [3]. **Since the overarching objective of this project is to attain multiple license plate detection and recognition, meaning there is actively one or more license plates visible in the current frame, I believe relaxing the problem to single license plate detection and recognition for training is requisite.** This approach will enable me to thoroughly understand the details of each module's construction and functionality in the completed AMLPR software system [4, 5].

Following the acquisition of a suitable dataset, I plan to utilize a convolutional neural network (CNN) as the backbone of this project to detect the location of a license plate in an image or video frame and facilitate cropping the image to only the license plate. The state-of-the-art algorithm I plan to utilize to develop a model is YOLO (You Only Look Once), specializing in efficient and real-time multiple object detection. **I plan to implement Version 3 of the YOLO algorithm because Version 3 has the most thorough documentation, and later versions of the algorithm are based on Version 3; however, I plan to perform further optimizations to the model to increase its performance and efficiency for the specified use case.** To briefly summarize YOLOv3, this algorithm functions by scoring regions of an image through several convolutional layers, where higher-scoring regions are noted as positive detections of whatever class the regions most closely identify with [6]. YOLOv3 is recognized for maintaining similar performance to alternatives while boasting substantial reductions in processing times, as visualized in **Figure 1.**, which is crucial for an application, such as a dash camera, that would process live data [7].

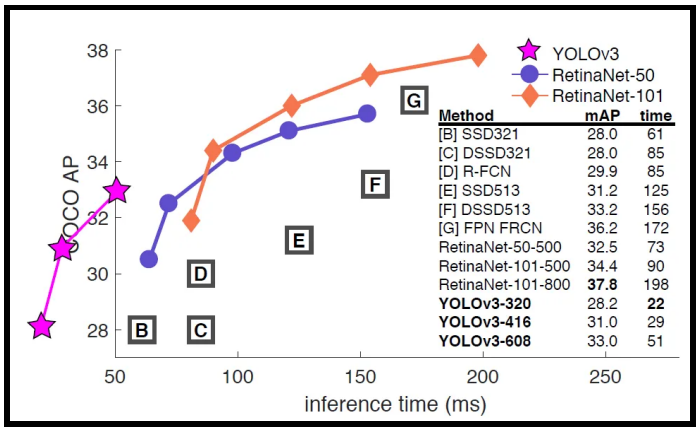

Figure 1. YOLOv3 Performance Comparisons [7]

The selected UCSD dataset does not have the locations of the individual characters on the license plates labeled, meaning YOLO cannot be implemented to perform character segmentation. Alternatively, I plan to use an OCR package to recognize, separate, and identify the characters on the license plate after applying various post-processing techniques to the detected and cropped license plate image. One possible package is PyTesseract, which serves as a "wrapper for Google's Tesseract-OCR engine" that can be optimized for different text applications and allows additional trained models to be specified for particular fonts, such as the font used on license plates [8]. Following the application of the OCR algorithm, I plan to develop a filtering algorithm to apply to the prediction output in an attempt to probabilistically remove any undesired extra character predictions aside from the license plate number. **In the big picture, the overall approach to this project will ultimately result in an integration of interconnected sub-modules comprised of uniquely tuned models/algorithms working together: a module to detect the license plate, a module to perform image processing, a module to perform classification of the recognized characters, and a module to perform filtering on the predicted license plate number.** I plan to utilize version control through Google Colabs and GitHub to manage updates for these sub-modules and test their performance individually before merging them into a streamlined program. Additionally, I plan to store the large weight files for the trained license plate detection model on Zenodo. By adhering to this design approach for completing the proposed project, I will be able to thoroughly understand the machine learning practices that go into creating an application associated with automated object detection and interpretation.

Project Milestones

Milestone	Target Completion Date	Status
Multiple License Plate Detection on Image	03/21/2022	Completed
Multiple License Plate Recognition on Image	04/04/2022	Completed
Multiple License Plate Detection and Recognition on Video	04/18//2022	Completed
Create Custom Dataset of Images and Videos	04/25/2022	Completed
Execute Performance Evaluations	05/02/2022	In Progress
Write Final Report	05/04/2022	In Progress

Evaluation Metrics

The following objective evaluation metrics are derived from the evaluation protocol of a YOLOv2-based automatic license plate recognition system [9] trained and tested on multiple license plate datasets, including the UCSD dataset I plan to utilize. The same evaluation metrics will be generated as described below for the constructed YOLOv3-based automatic license plate recognition system and each corresponding field (detection, recognition, and efficiency) will be appropriately compared to the results of the previous study, as found in the *Results and Discussion* section of [9]. **Lastly, as a final applicable evaluation challenge, I plan to create a custom dataset of images and videos to stream to my AMLPR software to observe its performance in a practical use case for a dash camera.**

Metric	Description
License Plate Location Detection Accuracy	Number of detected boxes that yield an Intersection over Union (IoU) greater than 0.5 with the ground-truth bounding box divided by the total number of license plates in the UCSD dataset [3].
License Plate Number Recognition Accuracy	Number of correctly recognized license plates (all characters are correctly segmented and classified) divided by the total number of license plates in the dataset.
Overall AMLPR Software Efficiency	Processing time of entire system between input to predicted license plate number(s) output for a single image or video frame.

References

[1] Hit-and-Run Crashes: Prevalence, Contributing Factors, and Countermeasures
[2] OpenCV: Automatic License/Number Plate Recognition (ANPR) with Python
[3] UCSD/Calit2 Car License Plate, Make and Model Database
[4] Performance Enhancement Method for Multiple License Plate Recognition (Paper)
[5] A Multi-tiered Automatic License Plate Recognition Strategy Using YOLOv2 Detector (Paper)
[6] YOLOv3: Real-Time Object Detection Algorithm (What's New?)
[7] YOLOv3: An Incremental Improvement (Paper)
[8] Python-Tesseract Python Wrapper for Google's Tesseract-OCR Engine
[9] An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO Detector (Paper)