Computer Vision Based Smart Parking Lot System

Mark Stiller, Sairam Tangirala, and Tae Song Lee
Applied Physics Laboratory, School of Science and Technology,
Georgia Gwinnett College, Lawrenceville, GA 30043

Abstract

In the institutions and venues having multiple parking decks, it can sometimes be difficult to locate empty parking spots. Examples include academic institutions, commercial parking decks, etc. Research on the prevalence of smart parking lots indicates a recent focus on the monitoring of electric charging stations within the traditional parking lots, rather than a real time monitoring of all the parking spots inventory. Our work focuses on designing, developing and prototyping an implementation of a hybrid computer-vision and client-facing app system to eliminate user's guesswork and to improve efficiency of the parking lot system. In our approach, the real-time video-feed of a parking lot's entrance/exit is processed using a portable, independent, computing system. The processed information is transmitted to a remote server via telemetry using LoRa (Long Range) protocol. LoRa is a low-power wide-area network protocol developed by Semtech Corporation that works independently without any other network (like Wi-Fi, WAN, LAN) dependencies. The server then processes telemetry-data and updates the parking spots inventory in real-time. The client-facing app then uses the real-time inventory data to provide current parking lot status to the user. Additional features like notifications about full, closed, and busy lots also are also planned to be implemented.

Introduction

- Little research has been published on parking lot systems in recent years. The general trend among existing research is a constant monitoring of the entire parking area.
- One implementation involved the installation of sensor nodes embedded in parking spaces and connected terminals to transmit data over a LAN. The ideal setup for this system is one terminal for every two parking spaces. Another implementation chose to monitor the entire parking lot with a single overhead camera mounted parallel to the orientation of parking spaces, using a lightweight edge-detection algorithm to evaluate spaces.
- Our system operates as individual compact computing systems mounted at the entrances/exits of a parking lot. Each unit tracks the number of vehicles that enter and leave the lot, with object detection being run every 30 frames of incoming footage and processed on a 3x3 depthwise image convolution matrix. The processed telemetry data is then sent to a remote server using the LoRa protocol. The server processes data from the individual devices and updates a parking space inventory for the client-facing app, providing real-time data to users.

Technologies Used

The system resides on a Jetson Nano Dev Board (**Fig. 1**), with an external camera (**Fig. 2**) used to gather video input. OpenCV and other python libraries such as imutils are used to parse incoming video. A MobileNetSSD performs object detection using the previously mentioned 3x3 depthwise image convolution matrix, and detected objects are checked against a trained TensorFlow model for object recognition. Telemetry transmission uses the LoRa protocol, which allows individual systems to operate outside the bounds of a LAN and at a lower power consumption than relying on Wi-Fi.

Fig. 1. Jetson Nano Dev Board

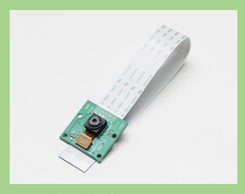


Fig. 2. Camera

Fig. 3. LoRa chip

Developer Kit Technical Specifications	
GPU	128-core NVIDIA Maxwell™
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4GB 64-bit LPDDR4 25.6GB/s

Fig. 4. Jetson Nano Dev Board Specifications

Development laptop technical specifications	
GPU	**Nvidia GeForce 940MX**
CPU	**Intel i7-6500U**
Memory	**12GB**

Fig. 6. Development laptop specifications

OpenCV

TensorFlow

Fig. 5. OpenCV and TensorFlow logos

Our System

- Footage used in testing was taken from an interstate overpass on a smartphone at 1080p, 60FPS. Footage listed as "cropped" in later figures was cropped to half the original width.
- Displayed in Figure 7 (below) is an example frame of the footage being processed.
  - An overlay at the bottom left of the image displays whether the system is waiting for objects to track, tracking detected objects, or (every 30 frames) running object detection.
  - Objects detected are given a sequential ID that follows them until object detection is run again and they are determined to have left the frame.

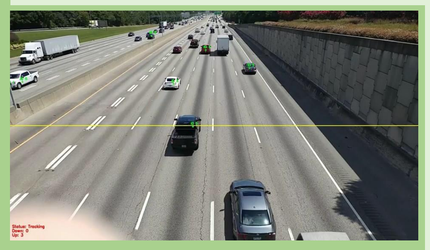


Fig. 7. Testing footage being processed

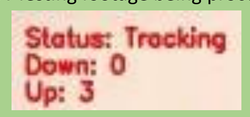


Fig. 8. Enlarged view of the overlay

- With only one compact computer system per entrance/exit to a parking lot, we are able to achieve high accuracy than an overhead camera at a lower cost than a system of embedded sensors and relay terminals.
- LoRa allows the system to cover larger parking lots with no additional setup required (beyond a power source), as well as lots where an institution's LAN does not reach.
- The modular nature of the system and the use of LoRa also makes the system extremely scalable, and it can be used to divide parking lots into sections for more accurate monitoring.
- As the system operates on entrances and exits, it is also ideal for parking garages where overhead monitoring is unfeasible.

Methods

All tests were run on both the development environment (a consumer laptop) and the deployment environment (a Jetson Nano Dev Board) in order to evaluate the performance of the system on the deployment environment.

- 3 video recordings of 60+ seconds in length were used as the base input for reproducibility, with a copy of each being converted to monochrome for testing purposes.
- An additional copy of color and monochrome footage was made and slowed to 25% of the original speed in an attempt to more accurately simulate the speed of vehicles entering and exiting a campus parking area.
- Each video source was processed in increments of 10 seconds (10, 20, 30, 40, 50, and 60 seconds in length), with each increment run 3 times and the average taken to be used for data analysis.
- The process was repeated with a "crop" option active in the script, which only processed the center half of the video sources.

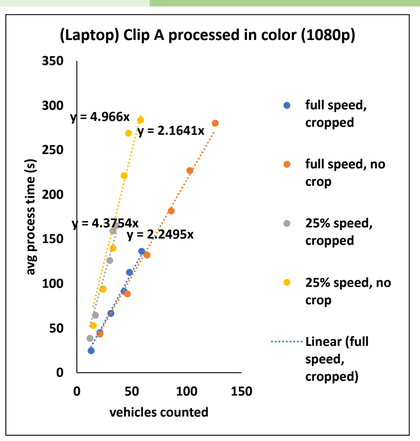
In total, there are 4 major resulting datasets:

- Color footage run on the development environment
- Monochrome footage run on the development environment
- Color footage run on the deployment environment
- Monochrome footage run on the deployment environment

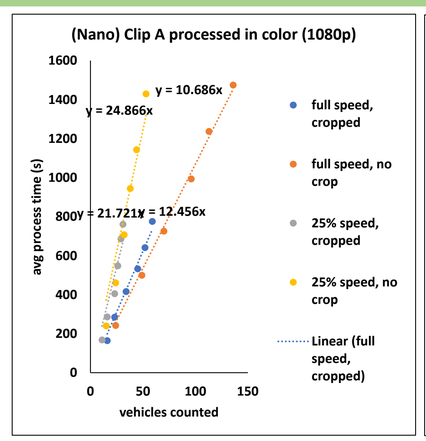
Each of these datasets is further divided into 4 plots:

- Full speed footage, without cropping
- Full speed footage, with cropping
- Slowed footage, without cropping
- Slowed footage, with cropping

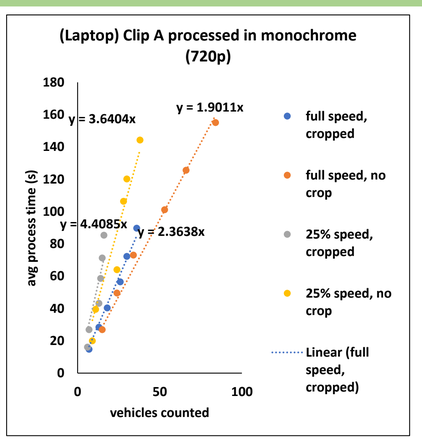
Results



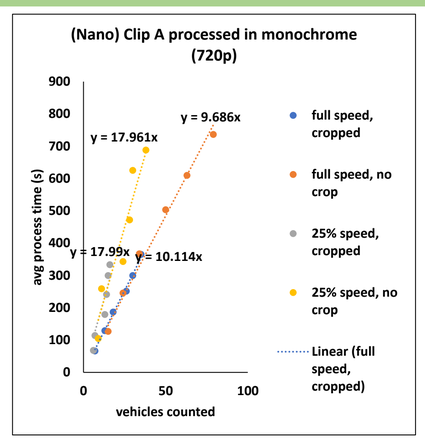
Plot 1
Color footage in development env.



Plot 2
Color footage in deployment env.



Plot 3
Monochrome footage in development env.



Plot 4
Monochrome footage in deployment env.

1. The results from both the development environment (**Plot 1**) and deployment environment (**Plot 2**) show that processing time is directly impacted by the number of vehicles tracked as well as the dimensions of the video source used.
2. The speed of footage used had a minor effect on performance, though slowed footage actually slightly increased the processing time compared to the full speed footage (this is likely due to vehicles spending more time in frame).
3. The Jetson Nano (**Plot 2**) performed approximately 5 times slower than the laptop (**Plot 1**) (a maximum of approx. 1500 seconds compared to approx. 300), however the results were proportionately very similar to those of the laptop.
4. Monochrome footage showed similar results, with a major improvement of halving the processing time taken, with approx. 150 seconds maximum for the laptop (**Plot 3**) and approx. 750 seconds maximum for the Jetson Nano (**Plot 4**).
5. The Jetson Nano saw the greater improvement using monochrome footage (**Plot 4**) over color footage (**Plot 2**) but is still significantly behind the laptop.
6. The number of detected vehicles is slightly lower overall using monochrome footage, but objects lost were those vehicles in the opposite set of interstate lanes (the lanes not intended to be monitored).

Considerations

1. Results shown on this poster only include the first of the video clips taken. However, the other two clips showed very similar results to the first and they were ultimately excluded from the poster for the sake of space.
2. Due to a lack of campus traffic to test the system on (due to the current world situation), the results give a skewed representation of the effectiveness of the system.
   - Even when cropping the footage, the program often picked up vehicles from 5 or more lanes on the interstate. On campus, there will be at most 2 lanes in view of any given system.
   - Interstate traffic levels are closer to the level of traffic seen at peak campus traffic hours, and while it serves as a sort of stress test for the system to use higher-density traffic, this compounds with the previous issue.
3. Currently, the Jetson Nano is using a TensorFlow model that is not designed to take advantage of the board's architecture, which results in the GPU cores sitting mostly idle during processing (around 3-4% usage on average). Once the TensorFlow model is replaced with one that can properly utilize the board's GPU, the system will see significant improvements in performance, and at the very least is expected to equal the processing speed of the laptop.

Conclusions

The number of vehicles detected by the system remained mostly consistent across hardware. There were some small variations among the averaged values of vehicles detected, but these are currently believed to be the result of background processes irregularly interfering with CPU usage and thus the object detection.

Use of monochrome footage and cropping footage to the relevant area results in the fastest possible processing time currently. More testing is needed to ensure that the single color channel does not have an impact on the object detection in a campus environment, though it is not expected to make a significant difference in the detection threshold for the distance the system is planned to be mounted at from the parking area entrances/exits.

The Jetson Nano is currently at a disadvantage due to it being a GPU-focused system using a CPU-focused TensorFlow model. Once the model is updated, testing will be conducted again with the existing footage to evaluate what other optimizations are needed.