Parking Solution for WKU: Car Counting with OpenCV

Patrick O'Boyle
patrick.oboyle773@topper.wku.edu
Computer Science
Western Kentucky University

Dr. Michael Galloway
jeffrey.galloway@wku.edu
Computer Science
Western Kentucky University

Abstract

Parking pains ultimately cost US drivers $95.7 billion annually due to the costs of search times, overpayment, and parking fines. These issues extend to college campuses, where parking can be especially troublesome. Western Kentucky University (WKU) is no exception. New dorms opened in 2018, but parking problems persisted. Though smart parking solutions are becoming more common, the smart parking market is dominated by companies that use expensive sensors and proprietary software. The objective of this research project was to investigate computer vision as an alternative smart parking solution for WKU. Computer vision is underutilized in the smart parking industry, but it can be a more cost-effective and versatile method for capturing data. For this study, a car counting system was deployed at Cherry Lot on the WKU campus to monitor the lot occupancy count (number of cars/other vehicles in the parking lot). Data was successfully collected from 9:00am to 4:00pm for seven days. This research is a contribution to open source, and it serves as a foundation for collecting and mining transportation data at WKU to improve campus life.

Introduction

WKU is demoing Park Smarter, an IPS Group service. The infrastructure is expensive. A single parking meter costs $900 [8]. Most CCTV cameras, however, cost less than $50. Cameras are able to monitor entire parking lots, making computer vision parking solutions potentially more cost-effective than methods used by mainstream smart parking vendors [3].

DIY digital image processing applications are becoming more widespread thanks to expanding machine learning resources and techniques. OpenCV is one of the many free tools available for computer vision applications.

MobileNets serve as a base layer in their convolutional neural networks. MobileNets simplify the network to meet the harsher resource constraints of less powerful machines, like Raspberry Pis [1].

MobileNet-SSD is the combination of MobileNet v1 and the SSD object detection network. MobileNet-SSD takes advantage of the accuracy of Fast R-CNN (part of SSD) while ensuring low-power usage [4].

Methodology

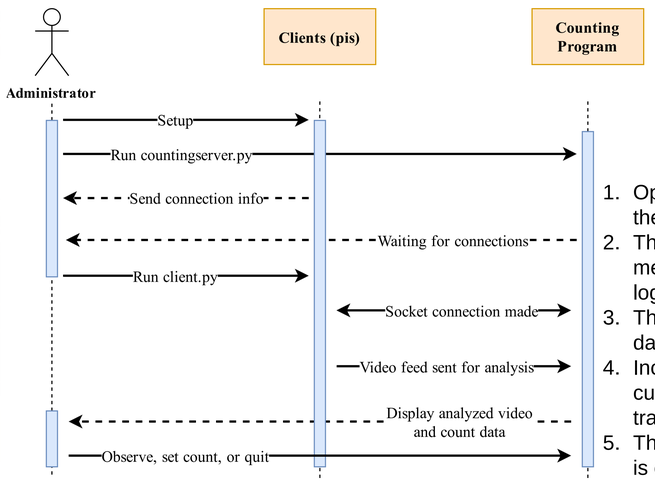
After in-lab tests, the counting application was deployed at Cherry Lot on the WKU campus. Data collection ran from 9:00am-4:00pm for 7 days. The counting server component and the client script were modified from a combination of different guides for OpenCV real-time streaming, video storage, and pedestrian counting [2, 5-7]

Event Recognition Algorithm

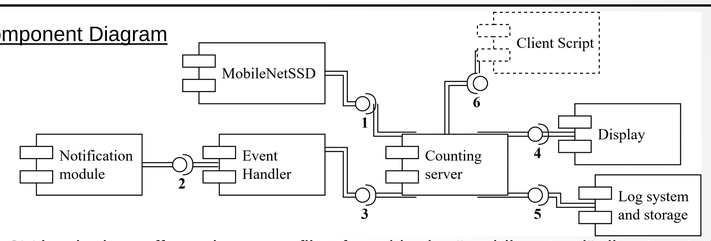
```
1:  procedure (centroids[])                    ▷ Analyze vehicle positions
2:      current ← centroids[0]                 ▷ Last position (x,y)
3:      position ← current.location            ▷ Vehicle is on left or right
4:      direction ← current − centroids.mean()
5:      if direction is left & position is left then
6:          entry
7:      else if direction is right & position is right then
8:          exit
9:      end if
10: end procedure
```

Exit Event

Sequence Diagram

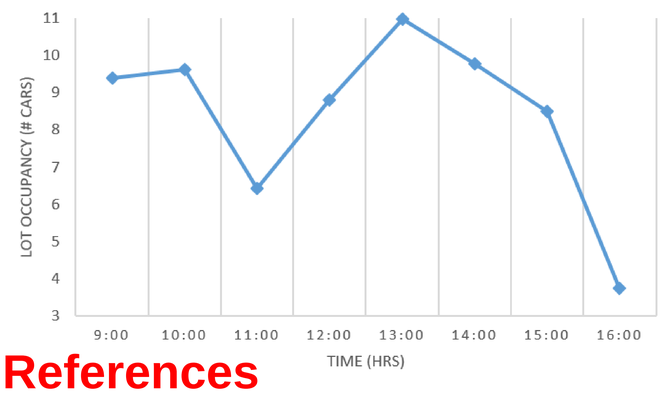


Component Diagram

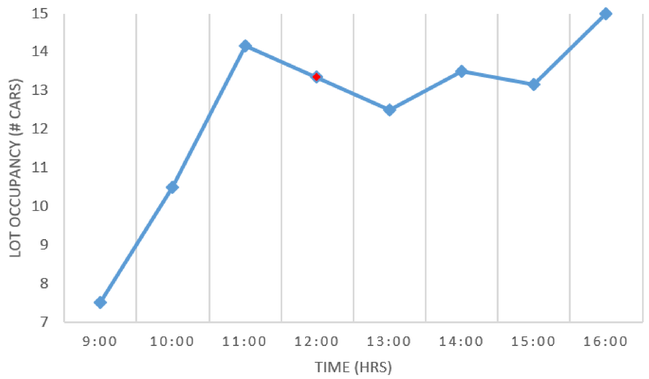


1. OpenCV loads the caffe and prototxt files found in the "mobilenetssd" directory to initialize the model used for object detection.
2. The Notification Module is a simple python script that uses Google's SMTP servers to send messages. The Event Handler module uses the notification module to send lot occupancy logs.
3. The Event Handler is integrated into the counting server script to manage lot occupancy data, notifications, and user interaction.
4. Incoming frames from the pi client are displayed along with the timestamp, client name, current vehicle count, vehicle centroids, and the current status (waiting, detecting, or tracking).
5. The Event Handler outputs log data that simple shell scripts organize. Written video footage is organized as well.
6. After establishing a TCP socket connection, the client script sends frames over ZeroMQ to the counting server for analysis.

Monday, Wednesday, Friday



Tuesday, Thursday



Results & Conclusions

The average lot occupancy counts from 9am-4pm on each day of the week were determined. The days of the week were grouped together according to the WKU class schedule (MWF and TR).

The measured car counts were compared to the actual number of vehicles in Cherry Lot twice a day. The count was usually off by only 1-2 vehicles. The missing or extra vehicles represent drift, the deviance from the real number of vehicles.

The WKU class time frames for Monday, Wednesday, and Friday are 8:00-8:55, 9:10-10:05, 10:20-11:15, 11:30-12:25, 12:40-1:35, and 1:50-2:45. The time frames for Tuesday and Thursday are 8:00-9:20, 9:35-10:55, 11:10-12:30, 12:45-2:05, and 2:20-3:40. No apparent correlations to the WKU class time frames were found for the current data set.

The deployment of the Cherry Lot car counting application was successful. This research has shown the effectiveness of computer vision approaches for smart parking systems. The open source angle of this research is a step in the right direction for opening up the smart parking industry.

References

[1] Andrew G Howard, Weijun Wang, Menglong Zhu, Tobias Weyand, Bo Chen, Marco Andreeto, Dmitry Kalenichenko, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. (2017). arXiv:arXiv:1704.04861v1
[2] Caleb Ogbonnaya. 2019. Pedestrian Detection and Tracking - A People Counting Based Application Using Embedded OpenCV. TopScholar (2019).
[3] Vijay Paidi, Hasan Fleyeh, Johan Håkansson, and Roger G. Nyberg. 2018. Smart parking sensors, technologies and applications for open parking lots: A review. IET Intelligent Transport Systems 12, 8 (2018), 735–741. https://doi.org/10.1049/ietits. 2017.0406
[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks. 8828, c (2016), 1–14. https://doi.org/10.1109/TPAMI.2016.2577031
[5] Adrian Rosebrock. 2016. Writing to video with OpenCV. pyimagesearch.
[6] Adrian Rosebrock. 2018. OpenCV People Counter. pyimagesearch.
[7] Adrian Rosebrock. 2019. Live video streaming over network with OpenCV and ImageZMQ. pyimagesearch.
[8] Michael Wilson. 2019. IPS Sample Quote