Technical Report CS 2019-05

Analyzing the Pedestrian and Vehicle Traffic of an Unsignaled Crosswalk

Michael John Dilla

Submitted to the Faculty of
The Department of Computer Science

Project Director: Dr. Oliver Bonham-Carter
Second Reader: Dr. Janyl Jumadinova

Allegheny College
2019

I hereby recognize and pledge to fulfill my responsibilities as defined in the Honor Code, and to maintain the integrity of both myself and the college community as a whole.

Michael John Dilla

MICHAEL JOHN DILLA. Analyzing the Pedestrian and Vehicle Traffic of an Unsignaled Crosswalk.
(Under the direction of Dr. Oliver Bonham-Carter.)

ABSTRACT

Vehicles present a danger to pedestrians. Often in urban areas, the two share space. This paper presents a tool for analyzing pedestrian and vehicle traffic in a shared space. The tool uses the Tensorflow Object detection API to count pedestrians and vehicles in an unsignaled crosswalk. Data from the tool can then be used to analyze trends in traffic. More complete knowledge of a pedestrian and vehicle shared space will then allow for more efficient use of the space.

Acknowledgements

I would like to thank my parents for giving me the opportunity to be at this school and for all the work they have put into raising me and all the love and support as well.

I would like to thank Dr. Oliver Bonham-Carter and Dr. Janyl Jumadinova for all the help guidance they have given me not just through the process of writing my thesis, but throughout my career at Allegheny.

Finally, I would like to thank Thea Gamboa for being constantly supportive and a source of encouragement and love.

Contents

List of Figures

Chapter 1

Introduction

Pedestrians and vehicles do not share space well. Since many cars weigh over 2 tons and travel at high enough speeds to cause serious harm to humans when struck, it is simple to see how problems can arise when the two are in close proximity. If there were a better way to monitor interactions between vehicles and pedestrians and analyze them for dangerous behavior, then we could make progress towards making the relationship between vehicles and pedestrians more symbiotic.

1.1 Motivation

1.1.1 Pedestrians Safety

Pedestrian fatalities have remained at a 25 year high for the past 2 years. This is also part of a 27% increase in pedestrian deaths from 2007 to 2016. In addition, pedestrian fatalities have increased in the United State's 10 largest cities by 28% [20]. Higher quality information is needed on pedestrians if officials are to effectively combat the rise in pedestrian fatalities. If city authorities had access to higher quality data on areas where pedestrians and vehicles come in close contact with one another, they would be able to implement different solutions to reduce or make safer such contact such as using sky-walks, closing certain roads to vehicles or pedestrians

Figure 1.1: An example of a barrier in a used in Paris, France to exclude vehicle traffic for a pedestrian only area

at certain times during the day, or improving road or crosswalk design. However, in order to make sure these solutions are cost effective and being implemented at the proper location, city authorities must know when and how much traffic a certain area is experiencing.

1.1.2 Urban Livability

Higher quality information on areas with high pedestrians and vehicle traffic can increase the livability of a city. If a certain area of a city is observed to have low vehicle traffic, compared to pedestrian traffic at a certain time of day, a portion of those roads could be closed to vehicles during that time. This way pedestrians can more quickly and comfortably navigate the area. This could also be positive for vehicles traffic as well. Instead of being delayed by pedestrians crossing the road, they could be rerouted to an area with less pedestrian traffic, decreasing delays.

The idea of closing roads to vehicles has already been implemented in many European cities (see fig. 1.1 for an example). Roads designed before the invention of the modern automobile tend to be smaller, hence making it difficult for pedestrians and vehicles to share such roads. Closing roads to vehicles has created a much more friendly space for pedestrians. Such spaces can then be used to improve the quality

of life of those that use them. Also, if the data for a crosswalk shows that vehicle traffic is far greater than pedestrian traffic at a given time, the crosswalk could be closed to pedestrians at that time. Then delays would be minimized for vehicles since they would not need to stop to accommodate pedestrians that wish to cross the road.

1.1.3 Traffic Delay Predictions

Data of pedestrian traffic can also be used to improve traffic predictions of GPS software. If more pedestrians are using an area at a certain time, that will cause a greater delay for vehicles. Then the GPS software would be able to better predict delays from pedestrians and avoid roads that have many pedestrians at the given time. Any delays in vehicle traffic carry a monetary toll as well. In the US traffic congestion has cost 305 billion dollars in 2017, which is an increase of 10 billion from 2016 [7]. This cost comes from the loss of productivity of workers standing in traffic, increase in delivery costs, and wasted fuel to name some factors. If we had GPS that could better predict pedestrian traffic. Although it would not have a massive effect on traffic congestion as a whole, it would lead to some amount of improvement. Gathering better data on pedestrian and vehicle interactions will lead to a better environment for both parties. It will be a safer environment for pedestrians, less pedestrian fatalities, and will create a more livable urban environment. It will make it so that there are more spaces for vehicles to operate without the interference of pedestrians and vice-versa. Lastly, better data will allow for better data for predicting delays cause by pedestrians for vehicles. This will cause less traffic delays and lessen their economic impact.

Figure 1.2: Market Square in Pittsburgh, Pennsylvania. An area that has high pedestrian traffic but low vehicles traffic that would benefit from limiting vehicle traffic

1.2 Current State of the Art

1.2.1 Tools for Vehicle Traffic Analysis

There are many different tools that work to analyze traffic. However, these tools do not focus on the relationship of pedestrians and vehicles. There are three main types of traffic analysis tools: 1)evaluating, simulating, or optimizing the operations of transportation facilities and systems, 2)modeling existing operations and predicting probable outcomes for proposed design alternatives, 3)and evaluating various analytical contexts, including planning, design, and operations/construction projects [6]. The different tools can be used for many different purposes. It allows those that design road systems to make more informed decisions about how to best design roads for maximum efficiency. It also helps to assign priorities to competing projects. Also, analysis tools can be used to evaluate designs for different road networks without the need to physically implement the designs. There are tools that can simulate road networks so that designs can be evaluated under different scenarios, which can be difficult to do experimentally since road conditions, such as traffic or weather, cannot be reliably predicted or reproduced when needed. They

Figure 1.3: An example of the graphic output from the TransModeler traffic simulation model

also reduce disruptions to traffic by estimating the effects that management strategies will have on a system before they are actually deployed. Tools also exist to optimize and maximize traffic flow in a network. In addition, many tools on the market today utilize a graphic display (as seen in fig. 1.2) which can be used to present solutions to those that are not well versed in traffic simulations. Traffic tools can monitor road performance and can then be linked to analytical tools to provide real time analysis on road performance.

There are a great many uses and benefits for traffic analysis tools; however, they do come with their own draw backs as well. There is a shortage of quality data on roadways. Collection of this data can also be very costly. This makes it difficult to implement many simulation type analysis tools. Furthermore, results obtained from less data intensive tools tend to be more general and less powerful. In addition to the cost of collecting data, the tools themselves are expensive. Even if the group conducting the study can afford them, the tools often have a steep learning curve and those that use them can be under pressure to minimize the time taken for the study, hence they do not take the time to train themselves properly. Thus, they do not utilizing the tool to its full potential.

1.2.2 Tools for Pedestrian Traffic Analysis

Pedestrian analysis tools are not nearly as well developed as vehicle analysis tools. There are numerous, well developed vehicle traffic analysis software available for purchase; however, tools for pedestrian analysis and data collection are still experimental. The pedestrian analysis tools that are available are tools that simulate crowd behaviors and dynamics. However, in order to train simulations for specific areas, for highest accuracy, there should be data from that specific area. Data collection on pedestrian behaviors is typically done manually by having a person either watching an area directly or having someone analyzing video footage. They would then watch the area and record the number of pedestrians and their behaviors that are in question. If one were to have enough footage to be an accurate image of the behavior of pedestrians of that intersection, then analyzing the footage would be very expensive both monetarily and in time. However, there are experimental methods of pedestrian analysis that involve the use of computer vision algorithms.

Using computer vision, one can automatically analyze pedestrian footage far faster than if it were to be analyzed manually. Not only that, but using computer vision, measure qualities of a road network that would not be measurable using manual analysis such as measuring the trajectories and mapping exact paths of pedestrians. Such measurements are used in [13], however they are applied to vehicles instead of pedestrians. These measurements can then be used to more accurately create and train models which simulate pedestrian behaviors.

1.3 Thesis Outline

This paper presents a tool that uses open source libraries to analyze pedestrian behaviors. My tool uses a camera to gather video data on an area of road in which pedestrians and vehicles will interact, such as a crosswalk. It then uses computer vision to count the pedestrians and the vehicles in the area at a given time. Each count will then be attached to a time stamp of when those counts were taken. The data can then be analyzed for trends in the number of vehicles and pedestrians relative to the time they occurred.

Data for the tool is collected using a monocular camera placed so that it has a full view of the area in question. The camera will then take a video stream of the area. The stream must be frequent enough so that no pedestrians or vehicles can pass through the area undetected, but the stream should be slow enough that the computer vision algorithms can still operate without lagging behind the stream. The computer vision algorithm then counts the pedestrians and vehicles in the image. After the computer vision model has counted the number of vehicles the data will then be prepared for analysis. After the tool has counted the number of vehicles and pedestrians, the numbers will then be paired with a time stamp of when the image was taken. The resulting tuple will then be stored in an csv file so that it can be used for analysis later.

After the tool has run for what ever time has been deemed necessary, the data will then be analyzed for significant trends in traffic in the area. The data is stored in an SQL database so that it can be easily retrieved for analysis. The data will be able to be displayed with a graph created using the python package MatPlotLib. The histogram will display time of day on the X axis and traffic of pedestrians and traffic of vehicles on the Y axis.

The goal of this tool is that the data will show patterns of the traffic. That way

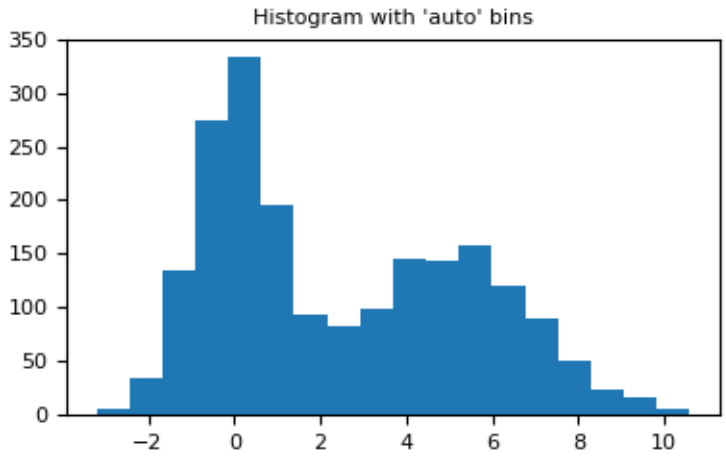

Figure 1.4: An example of the histogram that can be created using the Numpy Python library

those patterns can be observed and utilized to make roadways safer and more efficient. If the data shows that there is a large amount of vehicle traffic at a certain time, then it can be assumed that pedestrian traffic will have a larger impact simply because it will be effecting more drivers. It can also be assumed that if there is a large amount of vehicle traffic that the area will be less safe for pedestrian traffic. This effect is amplified if there is low pedestrian traffic. That is because if there is low pedestrian traffic drivers will be less attentive to pedestrians[7]. It can also be assumed that high pedestrian traffic will cause more delay for vehicles, since more pedestrians mean that drivers will have to give way to more pedestrians. High pedestrian traffic and high vehicle traffic will be a non-optimal situation for all parties since delays will be at a maximum for both parties.

Using the data created by my tool, I hope to make transportation networks safer and more efficient for all parties that use them.

1.4 Background

1.4.1 Computer Vision

There are 4 different computer vision algorithms that I considered for use in the tool.

1. OpenCV Pedestrian Detector
2. OpenCV Blob Detector
3. The Fastest Pedestrian Detector in the West
4. TensorFlow Object Detection

1.4.1.1 OpenCV Pedestrian Detector

OpenCV comes with a pre-trained pedestrian detection model that uses histograms of Oriented Gradients (HOG) and a support vector machine. The method was first suggested by Dalal and Triggs [8]. The OpenCV model was trained using the CalTech Pedestrian Dataset which utilizes 10 hours of 640x480 30Hz video taken from a vehicle driving through an average urban environment containing approximately 2300 distinct pedestrians [11].

The method proposed by Dalal and Triggs is a 6 step method (see steps shown below). The basis behind said method is that the appearance of an object can be characterized by edge directions and distribution local intensity gradients (which will be explained later). This is implemented by dividing an image into small groups of adjacent pixels called cells. Then, for each cell in the image, compute a histogram describing the orientation of the gradient or edge direction in said cell. In order to reduce interference from lighting or shadows, the results from each cell are

normalized over slightly larger regions known as blocks. The normalized result are called the Histogram of Oriented Gradient (HOG) descriptors of the image. Finally, the HOG descriptors are classified using a linear support vector machine (SVM) as belonging to the object in question or not.

1. **Normalize Gamma and Color**

   The first step of OpenCV's pedestrian detector is gamma and color correction. This is where the Gamma, colloquially referred to as brightness, is adjusted in the image. That way there is minimal interference from shadows and differences in illumination in the image. In addition to gamma correction, color correction is performed as well. However, it was found that color correction produced negligible improvements in performance. It is also worth noting that color images performed 1.5% at 10^{-4} false positives per window better than grayscale.

2. **Compute Gradients**

   In this step, the image is processed in order to remove any noise that could interfere with analysis. The method of processing is called Gaussian Smoothing. The more an image is "smoothed", blurrier the image is. This method reduces the effect of static or other noise that could have occurred when the image was created. In addition to smoothing, an image mask is also created. The mask is possible values that a pixel can have. It was experimentally found in [8] that no smoothing and an image mask of [-1,0,1] produced the best performance. For color images, a gradient is computed for each color channel. The channel with the highest norm is then chosen as the pixel's gradient.

3. **Weighted Vote into Spatial and Orientation Cells**

   In this step, each pixel casts a weighted vote for the orientation of the gradient element that pixel contains. The weight for the vote of the pixel was decided

using the image mask in the previous step. The votes are then collected into bins over each cell. The orientation of the bin can either be 0°-180°for unsigned gradients or 0°-360°for signed gradients. The vote is then decided using a function of the gradient's magnitude. A larger magnitude typically represents an important feature.

4. **Contrast Normalize over overlapping Spatial Blocks**

   The gradient orientation is then normalized over block. A block, as stated before, is a group of adjacent cells. This is necessary because gradient strengths over an image can vary greatly. This is due to differences in illumination between different parts of an image. It can also be caused by differences in color. Without normalization, differences in illumination could be seen as an edge and generate a false positive or obscure the object in question. To perform normalization, cells are grouped together in blocks and normalization is performed over all cells in a block. Performance is also increased if blocks overlap during normalization. The final descriptor is the vector of all normalized components of the blocks in the detection window.

5. **Collect Histogram of Oriented Gradients Over Detection Window**

   A detection window is then scanned throughout the entire image at every location and scale. Using the detector window to take samples at different scales creates an image pyramid. That way features can be detected regardless of their size in the original image. The step that the detection window takes is the size so that each block is sampled 4 times.

6. **Linear Support Vector Machine**

   The linear SVM is then run on the resulting image pyramid. A SVM is a machine learning supervised learning model. It is used for classifying datasets.

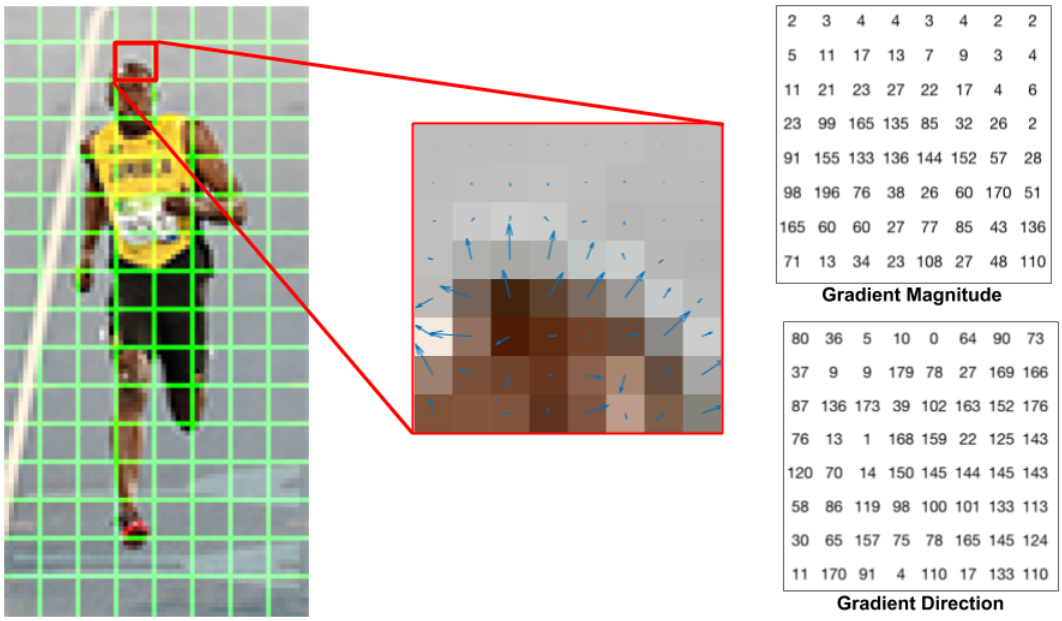

Figure 1.5: An example of a HOG descriptor created by OpenCV

Given a training model it will classify new examples as apart of one class or another. The method used by Dalal and Higgs utilizes a soft linear SVM that is used when data is not linearly separable. In this case, the SVM will classify a block as apart of the given feature or not apart of the feature.

1.4.1.2 OpenCV Blob Detection

OpenCV Blob Detections is a much simpler alternative. The method for blob detection was first proposed by Danker and Rosenfeld [9]. A blob is a group of pixels that share a common property. Properties that the OpenCV bob detector uses is shown in figure 1.4. The blob detector uses 4 steps.

1. **Thresholding**

   Thresholding is taking a method of image segmentation. Thresholding constructs a binary image from a grayscale image. In order to threshold an image, each pixel must either be black or white. Whether a pixel is black or white

is decided based on whether the intensity of the pixel is over some constant I'm going to call the thresholding limit. When using OpenCV blob detection there is a thresholding minimum, a thresholding maximum, and a thresholding step. The base image is thresholded as many times is need to get from the thresholding minimum to the thresholding maximum in increments equal to the thresholding step. That means the algorithm creates multiple thresholded versions of the original image.

2. **Grouping**

   Then, for each binary image created during the thresholding process, white pixels are grouped together. These groups of white pixels are called binary blobs.

3. **Merging**

   Then centers of the binary blobs are computed. If the blobs are less than a certain distance apart, this distance is called the merge distance. The coordinate of the center point is calculated using the method proposed in [15] shown below.

$$x_{center} = \frac{x_{max} + x_{min}}{2} \; and \; y_{center} = \frac{y_{max} + y_{min}}{2}$$

4. **Center and Radius Calculation**

   In this stage the center of the merged blobs are calculated. The centers are then used to calculate the radii of the merged blobs.

The blobs can then be filtered using different criteria. Blobs can be filtered by size. There can also be a minimum area where blobs under the minimum area will be excluded. Blobs can also be sorted by circularity. Circularity is defined as such

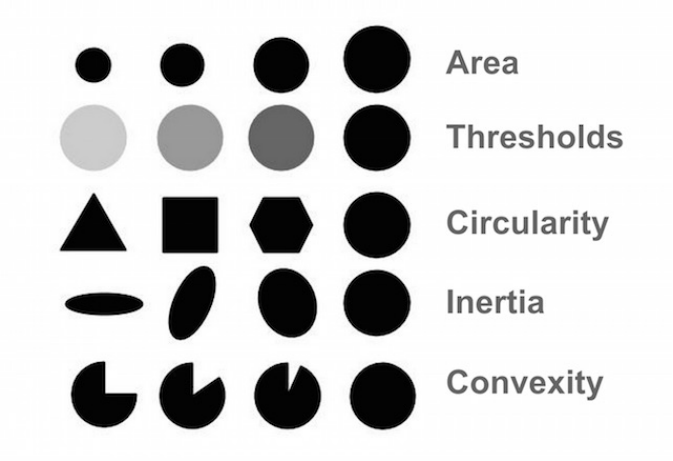

Figure 1.6: Examples of the different criteria used to classify blobs

$$\frac{4\pi Area}{perimeter^2}$$

The next criteria that can be used to sort blobs is convexity. Convexity in this case is defined as the area of the blob divided by the area of its convex hull. The convex hull of a shape is tightest convex shape that completely encloses the shape. The last criteria the OpenCV Blob detector uses to sort is by inertia ratio. Inertia ratio is how elongated a shape is. Inertia ratio is calculated by taking the ratio of the minimum radius and the max radius of a shape.

1.4.1.3 The Fastest Pedestrian Detector in the West

The next pedestrian detection algorithm that I considered is "The Fastest Pedestrian Detector in the West" (FPDW) [10]. The improvements made in FPDW were not improvements made in reducing false positives or false negatives, but made vast improvements in computational time as shown in figure 1.5. Instead of constructing a densely sampled image pyramid FDPW used features from one scale

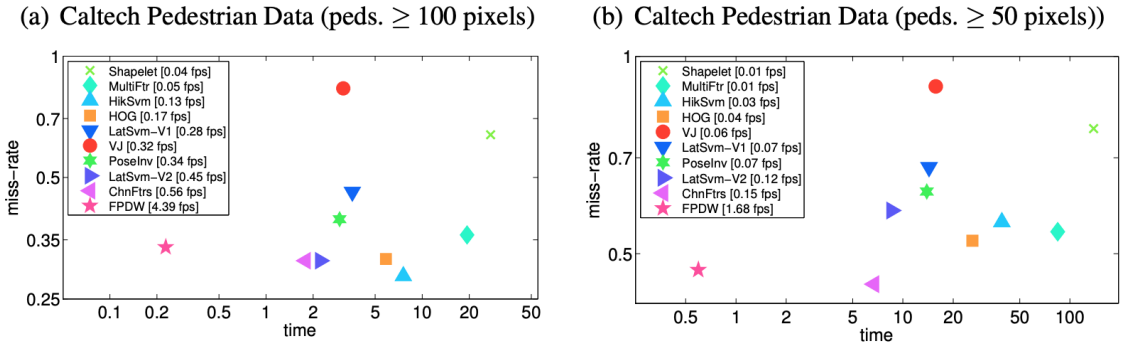

Figure 1.7: The computational speed of FPDW versus other state-of-the-art pedestrian detectors on the caltech pedestrian dataset

to approximate features on near by scales. FPDW uses gradients computed at one scale to approximate gradients at different scales. Since the features in an image do not change if the image is scaled differently, then it follows that approximations at different scales should be similar enough to not reduce the accuracy of the detector.

FPDW has shown experimentally that it is 10-100 times faster than other state-of-the-art algorithms with only a 1%-3% loss in accuracy. Where other detection methods take 10 seconds per image, FPDW runs at slightly over 1 image per second[10].

1.4.1.4 TensorFlow Object Detection

TensorFlow is a large scale machine learning system that operates in heterogeneous environments. It is an open source project that was developed by Google and is still used by many Google services in production. It is easily scalable so that it can be used on distributed clusters, local workstations, or even mobile devices. Deep neural networks such as TensorFlow have achieved breakthrough performance with object detection. TensorFlow ships with an easy to use object detection API that can be used to train you own model [2].

Figure 1.8: TensorFlow object detection trained to identify vehicles

Chapter 2

Related Works

There has been significant research on the topic of traffic analysis. The majority of the focus has been on the analysis of vehicle traffic since vehicle traffic has more of an impact for more people. However, the analysis of pedestrian traffic and how it effects vehicle traffic has been the subject of a great deal of research. A portion of that research has been on the safety of pedestrians while occupying a shared space for vehicles and pedestrians. Although, there has not been a great deal of research on the interaction of vehicles and pedestrians in an unsignaled crosswalk, much of the existing research can be applied to the subject.

2.1 Primary Sources

2.1.1 Quantitative analysis of lane-based pedestrian-vehicle conflict at a non-signalized marked crosswalk

When using an unsignaled crosswalk, conflict between pedestrians and vehicles are inevitable. The study conducted by Almodfer et. al. [3] aims to classify the types of conflict between pedestrians and vehicles. In this study data was collected using camera's placed in buildings bordering the crosswalk, as seen in figure 2.1, and footage was analyzed manually. [3] proposes a method for evaluating the

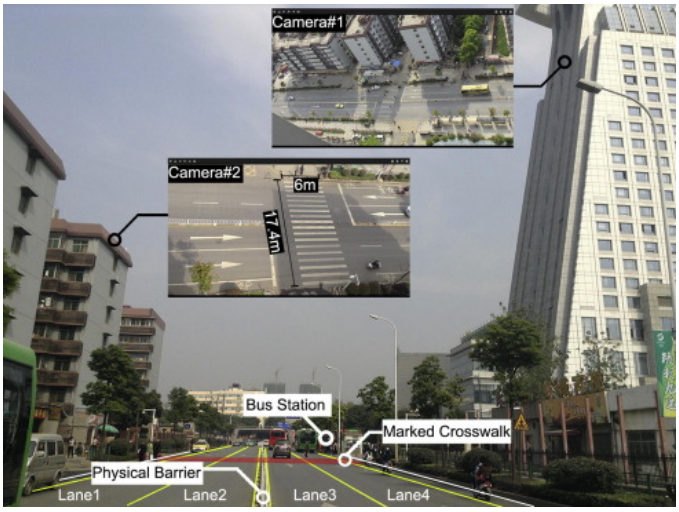

Figure 2.1: An image of the test area used in the study [3]

"post-encroachment time" between pedestrians and vehicles. It also uses the findings on lane based conflict to draw conclusions for pedestrian wait time.

2.1.1.1 Classification of Traffic Conflict

[19] defines traffic conflict as "... an event involving two or more road users, in which the action of one user causes the other user to make an evasive maneuver to avoid a collision". That means that a conflict occurs when an oncoming vehicle must brake abruptly, swerve to avoid a pedestrian, or if the pedestrian must take evasive action, such as speeding up or jumping out of the way. In order to quantify conflict in crosswalks there are different proximity measures that can be used. They are time to collision (TTC) and post-encroachment time (PET). TTC is the expected time until two road users collide and PET is the time between when a road user leaves a potential conflict area and when the other road user enters the conflict area. Road conflicts are then categorized by their severity based on thresholds of their TTC and PET.

This particular study defines a new criteria called lane-based post encroachment time (LPET). LPET is PET where the potential conflict area is the entire lane of

traffic of the same direction. In their study they classify pedestrian vehicle conflict in 3 thresholds of LPET, an LPET greater than 5 seconds is a potential conflict where a pedestrian crossed in the presence of a vehicle, but the vehicles is far enough away, slight conflict when LPET is between 1 and 5 seconds and where a pedestrian crosses the crosswalk and the vehicle is far, and serious conflict when LPET is less than 1 second and the pedestrian had to rush through the crosswalk.

2.1.1.2 Results

The wait time of a pedestrian at a crosswalk is defined as the difference between their arrival time at a crosswalk and their departure time. It was experimentally shown that waiting times under 3 seconds caused about twenty times more serious conflicts than those that waited between three and thirty second. Also those that waited for more than 30 seconds crossed in very risky situations.
Although the purpose of this study is different to mine, there are different aspects of this study that can apply for my own works. For example, my work could be expanded so that waiting times and distances are calculated automatically without the need for a person to manual watch all of the video footage. If that could be achieved, then it would be a tool that could automatically determine the safety of a crosswalk by counting the numbers and severity of conflicts. This could then be a tool for city officials to determine crosswalks that need to be redesigned.

2.1.2 Vision-based pedestrian behavior analysis at intersections

The research that was done by [21] was based on creating a computer vision based pedestrian detection and data collection tool. Their tool also analyzes waiting time, crossing speed, and facility utilization. The focus of their research was to mitigate

the problems caused by the use of optical flow detection method. Optical flow uses movement to approximate features in the frame; however, this method ceases to work if the target is not moving. This means that using optical flow to detect waiting time, the amount of time a pedestrian is stopped at the intersection, undesirable. Optical flow also does not handle the case where there are a group of pedestrians walking close together or if the video is of poor quality. The solution that was proposed by [21] was a combination of motion based pedestrian detection and appearance based pedestrian detection. The best detections from each method are then combined using what is called contextual fusion.

2.1.2.1 Motion Based Pedestrian Detection

Motion based pedestrian detection is done by using adaptive background subtraction since it is more reliable for small moving objects and less computationally expensive than optical flow. They use a Gaussian Mixture Model to create an adaptive background. Each pixel is modeled by a certain number of Gaussian mixtures which handle lighting changes, slow moving objects, and repetitive background motion, such as swaying branches or flags. A pixel is then part of a pedestrian if it does not fit any of the background criteria.

2.1.2.2 Appearance Based Pedestrian Detection

Appearance based pedestrian detection is using the values of the pixels themselves to determine what is and is not a pedestrian. Unlike motion based pedestrian detection, appearance based detection is done on a single image and not a stream of images. This method of detection uses positive and negative image samples to train an object classifier. This classifier is then used to determine if an image contains a pedestrian. This is the type of detection that is used in Tensorflow object detection

and OpenCV's object detector.

2.1.2.3 Contextual Fusion

The use of both Appearance and motion based pedestrian detection allows for the detector to be more accurate by being able to use one method of detection over the other when one is more reliable. Appearance based detections are only performed in mix areas. Mix areas are areas where both detection methods are being run. They are defined by the user. In the case of pedestrian detection, mix areas are around signals and crosswalks as these are the areas that are of the most interest. This reduces the appearance of false positives by allowing the two methods to cross reference with each other.

2.1.2.4 Results

The researchers were able to create a tool that could measure the wait times and average speed of pedestrians in an intersection. The results from their study can be found in figure 2.2. Wait time, in this study, was determined to be when the pedestrians speed fell under a certain threshold. The pedestrians speed during wait time was not factored into the calculation for the pedestrian's average speed. Figure 2.3 is a heat-map that shows the density of moving and waiting pedestrians. The tool was not only able to map the pedestrians' usage of the crosswalk, but also the area around the crosswalk. In figure 2.3(A) you are able to see where the tool was able to detect the path of jaywalkers. In figure 2.3(B) you are able to see a higher density of pedestrians waiting outside of a shop.

The tool that was developed by [21] represents a possibility of what the end goal of my project in terms of pedestrians detection. If I were to expand my research, I would like to implement features that were used in this project such as the wait

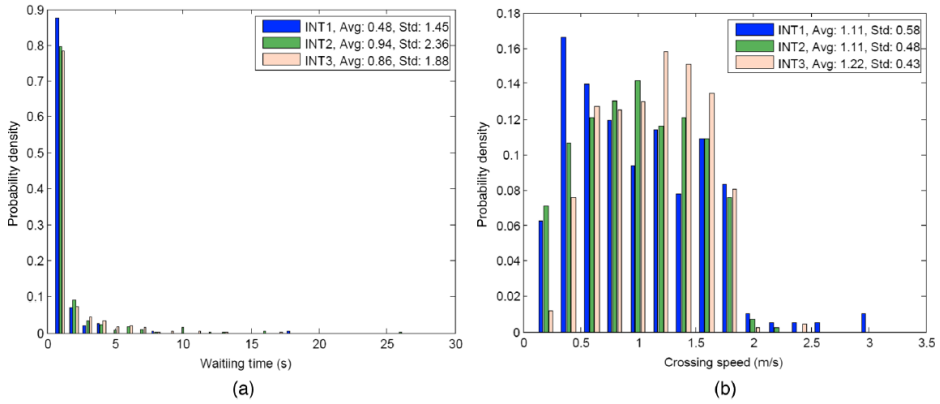

Figure 2.2: This graphic shows the (a) average waiting time and (b) average crossing speed in meters per second for three different intersections [21].

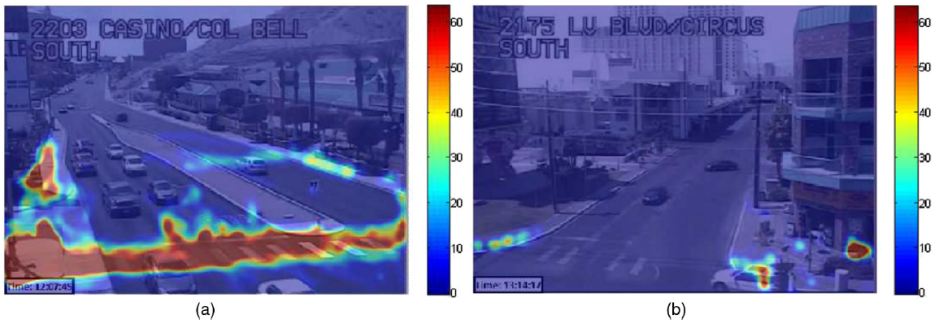

Figure 2.3: This is a heat map showing the density of pedestrian traffic in the camera frame. The areas with darker colors indicate higher pedestrian traffic. the heat maps show (a) the moving pedestrian density and (b) the density of waiting pedestrians in two different intersections [21]

time recorder and speed calculator. I would like to implement those features in addition to the ability to plot the results with respect to time, that way any trends with respect to time can be analyzed. With that information, one would be able to optimize the road layout and regulation to best fit that particular intersection whether that be by changing traffic signal timers to favor pedestrians or vehicles at certain times of the day or restricting access to the road to best fit the intersections needs.

2.1.3 Automated Analysis of Pedestrian Group Behavior in Urban Settings

The focus of [22] was to create a tool that is able to analyze pedestrian group behavior. The motivation behind analyzing group behaviors is to see how pedestrians interact with their surroundings to understand how to create urban environments that are optimized for pedestrian usage. It was found that most pedestrians walk in groups of 2 or more, as found in [4]. It follows that most of the focus on analyzing pedestrian behavior should be focused on analyzing groups. The main goals of their research was to classify the similar behaviors of pedestrians in groups, count pedestrians in groups, and demonstrate the feasibility of such a project.

2.1.3.1 Methods

The first step in their method of pedestrian group tracking was camera calibration. This was done by mapping real world coordinates with image plane coordinates. This allows for accurate speeds and distances to be recovered from the images. The next step of the method is the similarity measure of a group of pedestrians. The motion process of a pedestrian, which is called ambulation, is made up of a series of

steps. The length and frequency of those steps determine the pedestrian's walking speed. Also, a pedestrian's speed is not constant and varies over time. A pedestrians walking speed can be modeled by a composition of cyclic functions that repeat over time, as seen in the graph in figure 2.4. These properties were used to determine the similarities of those in groups. The step after similarity measures is group size calculation. In a group there would be slight differences in similarity measures. Group size is calculated by finding sub groups within a group until the group is broken up into multiple groups of the smallest size, one pedestrian.

2.1.3.2 Results

When tested, the tool was able to measure differences in walking speeds of people in groups versus other groups walking close by or those just walking alone. When applied to solve problems in real world applications, this tool provides a novel approach to measure how pedestrians interact with the walk way. The data this tool provides give unique insight on how pedestrian utilize sidewalks and other structures as well as how those structures effect pedestrian movement. The tool also, does not need to be applied solely to sidewalks. I can be applied to places like airports or train terminals and other high foot traffic areas.

I found that mapping real world coordinates to image coordinates to be a very interesting idea. The ability to measure distance in an image opens more possibilities for this area of study aside from those presented in this paper. That ability would be able to automatically measure pedestrian-vehicle conflict presented in [3]. This is something I would like to incorporate into my project in the future to add more functionality.

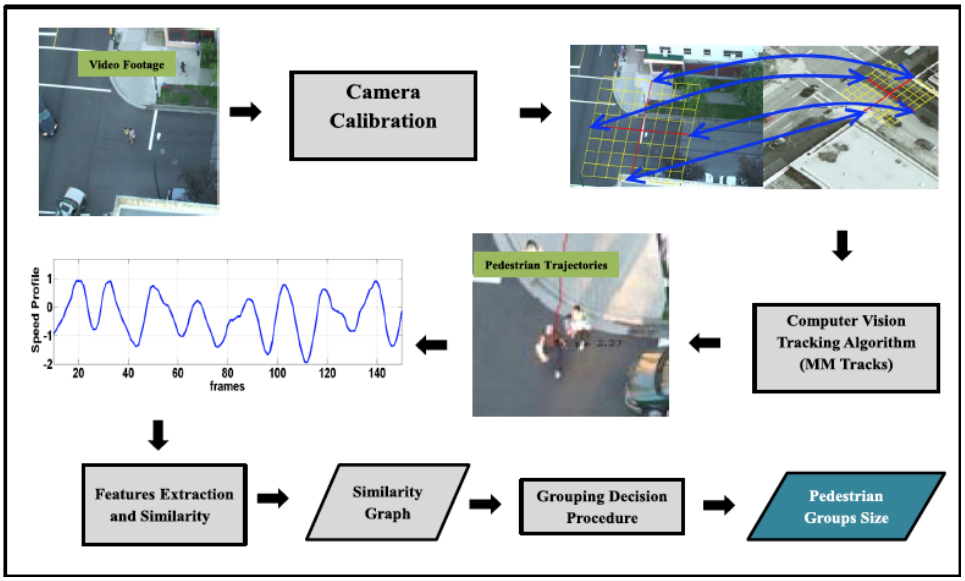

Figure 2.4: A flow chart showing the architecture of the tool given in [22]

Chapter 3

Method of Approach

3.1 Experiment Outline

3.1.1 Hardware

The hardware necessary for the tool is a camera capable taking a photo stream at about 5 frames per second. The other hardware component is a processor that can run the computer vision algorithm.

The hardware that was used for the tool was a Logitech webcam and a laptop running Ubuntu 16.04. The laptop was able to run object detection at a rate of about 2 frames per second, which is fast enough to not miss nay vehicles from the chosen vantage point.

3.1.2 Experiment Environment

The tool was tested on a crosswalk located on North Main Street in Meadville, Pennsylvania on the campus of Allegheny College near Alden hall. The crosswalk is well lit by numerous LED street lights. That way even during night, the tool will still be detect traffic in the crosswalk. The crosswalk also has significant traffic from both pedestrians and vehicles. There are a wealth of pedestrians using the crosswalk since it is near one of the main buildings on Allegheny College's campus. This

Figure 3.1: The webcam that was used in data collection

ensures that there is significant data for pedestrian traffic. North Main Street is also one of the most heavily traveled roads in Meadville. This will assure that there is significant data for vehicle traffic.

The camera was placed in a window in the near by Alden Hall. The hall is close enough to the crosswalk that it has a view of both sidewalks and the road. It was also placed inside the building so that the tool, for the purposes of prototyping, does not need to be weather resistant.

The detector needs to be able to process images fast enough that no pedestrians or vehicles will be missed and hence not be counted. The photo stream will need to run at a minimum of 2 frames per second so that no pedestrians or vehicles are missed. The desired maximum speed of the photo stream is 5 frames per second. That means that the computer vision algorithm will need to be able to operate 2 times for each image, since both vehicles and pedestrians will need to be detected.

Figure 3.2: A picture the crosswalk that will be used in the experiment

3.1.3 Selection of a Computer Vision Algorithm

In order to select one of the four computer vision algorithms that were outlined in the introduction each will be implemented and trained for pedestrians and vehicles and then they will each be tested in the experiment environment. Which ever performs the best will then be selected for full data collection.

3.1.3.1 OpenCV Pedestrian Detector

OpenCV's pedestrian detector ships with OpenCV pre-trained. Since the detector come already implemented and already trained, this is the simplest of the 4 options. It is not trained for vehicles so the detector would need to be retrained for vehicles. It also able process video at a minimum of 2 frames per second. However, it is worth nothing that OpenCV pedestrian detector has poor accuracy as seen in figure 2.3.

3.1.3.2 OpenCV Blob Detector

OpenCV's blob detection would be very easy to implement. However, it would require a great deal of time of calibrate the algorithm so that it can recognize pedestrian shaped blobs and vehicle shaped blobs. The blob detector sorts blobs by different parameters. Through trial and error I would need to calibrate the algorithm with the best parameters so that it can recognize pedestrians and vehicles for the testing environment. Since the blob detector works by thresholding, if there are 2 pedestrians or vehicles that are overlapping, they could be merged into one blob. Then the merged blob would not fit the necessary parameters and not be recognized.

3.1.3.3 The Fastest Pedestrian Detector in the West

The main drawback about using FPDW is that there is not optimized implementation available for use. If I were to use this algorithm, I would have to write my own implementation. Since I do not understand many of the mathematics used by the author for the approximation of features at different scales, I will not be testing FPDW for this tool.

3.1.3.4 TensorFlow Image Recognition

TensorFlow is the algorithm that is most appealing before testing. It does not come pre-trained for pedestrians or vehicles so the detector would need to be trained. However, there is abundant documentation for the training and usage of TensorFlow. Also TensorFlow has shown to experimentally outperform the OpenCV pedestrian detector by as much as 20% [5].

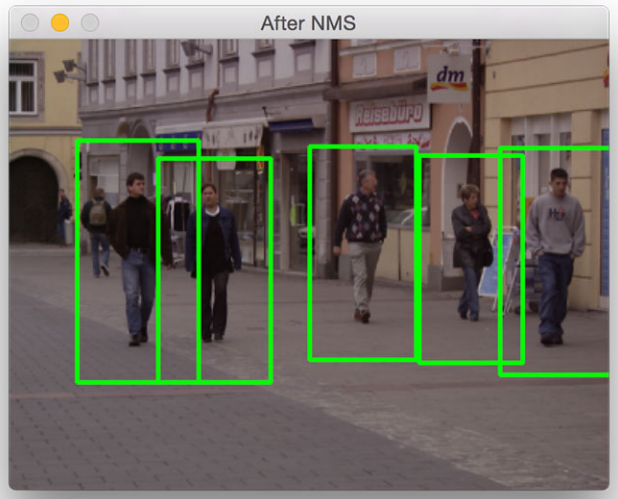

Figure 3.3: An example of the OpenCV Pedestrian detector processing an image. It is worth noting that there are 2 pedestrians in this image that the detector missed.

3.1.3.5 Rational behind the Choice

The algorithm that I chose was the Tensorflow object recognition. OpenCV's pedestrian detector operated in real time well, but it was not as accurate as the Tensorflow algorithm. Blob detection is not well suited for this purpose as the size and shape of both pedestrians and vehicles can very greatly and I would not be able to calibrate the algorithm to achieve the necessary levels of accuracy. Since there are no pre-built Fastest Pedestrian Detector in the West that disqualifies it since it would take too long to implement it myself. Also the resulting implementation would likely not be as high of quality as the Tensorflow or OpenCV. Hence Tensorflow proved to be the best choice.

3.1.4 Data Characteristics and Storage

The data that is produced from the tool is a tuple containing a timestamp, the number of pedestrians in the image and the number of vehicles in the image. The data will then be analyzed for meaningful trends. Ideally, in order to ensure that any trends that are found are significant, data collection will run continuously for three weeks; However, in the case of this experiment data collection was only able to happen for about 23 hours. Despite the shorter than optimal data collection time, the tool collected 124,901 data points

The data that was output from the tool was stored in a csv file. The file contained a column for the date the image was taken, the time the image was taken, the number of seconds since 1970 the image was taken, and the counts for pedestrians and vehicles detected.

3.1.5 Data Analysis

Since the data set is so large, there is no practical way to analyze it by hand. The most reasonable way to organize the data for analysis would be a line graph plotting the traffic counts versus time. This way any repetitions, patterns, or relations can be observed. Also, the average number of pedestrians and vehicles detected per hour can be used to analyze any trends. That way it can be shown that certain times have higher or lower traffic than average. Given enough data, one could also calculate the average traffic counts at a given time (for example, the average number of pedestrians detected from 11:00 AM to 12:00 PM on Mondays). Those calculation could then be used to show pattern that occur at a given time.

Chapter 4

Implementation

In order to build my tool, I used a combination of Tensorflow for object detection, the Python Multiprocessing library, and OpenCV to handle input from the webcam and to output to the screen. The final tool is able to count vehicles and pedestrians in an image and output those counts to a csv file along with the time those counts occurred.

4.1 Important Libraries

4.1.1 Tensorflow Object Detection API

The Tensorflow object detection API is an open source framework built for Tensorflow that allows users to easily build, train, and deploy object detection models. In order to save time a pre-trained model was used from the Tensorflow object detection Model Zoo. The model that was used was trained on the Common Objects in Context (COCO) dataset [16]. The dataset contains labels for 91 common objects that "would be easily recognizable by a four-year-old".

Tensorflow object detection works by loading the object detection model into a Tensorflow 'Graph'. The object detection model, which is provided by Tensorflow, is the framework that utilizes the Tensorflow machine learning library for object

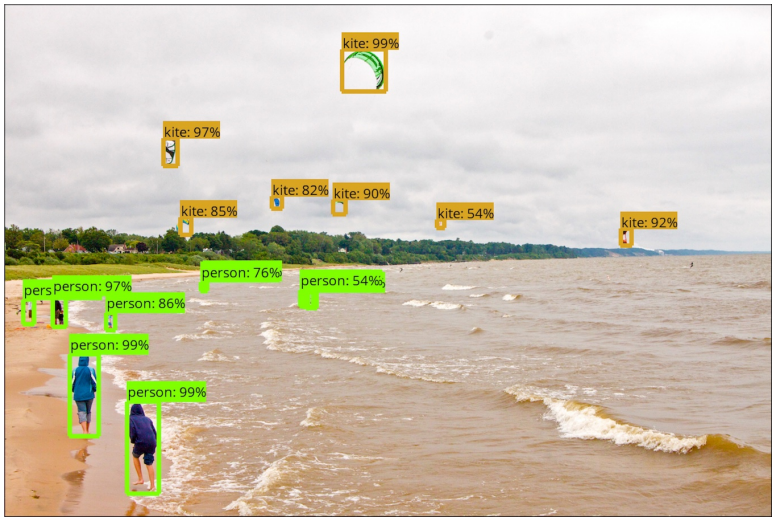

Figure 4.1: An example of running a Tensorflow object detection model trained using the COCO dataset on a single image.

detection. A graph is made up of 'Tensors' which are units of information pertaining to a certain model. From example, tensors associated with the object detection model are object classes (what is being detected in an image), boxes (the region of an image in which an object is detected), and the number of detections; the image itself is also represented at a Tensor. Once the model has been loaded into the graph, the actual object detection can occur. In order for any calculations to be done, the graph must be loaded onto a 'Session'. A Session is an environment in which operations are done on Tensors. In that Session the values for the Tensors are evaluated. Once the detection has occurred, the results can then be drawn on the image and displayed to the user or recorded in other ways. Objects in the image will be given a confidence value, signifying how confident the model is that it has properly recognized what the objects are. Confidence values over 50 will result in an actual detection. For more information about the Tensorflow library and the object detection framework see [1] and [17]

4.1.2 Multiprocessing

The Multiprocessing library is a Python library that supports parallelism and multi-threading. This project uses process pools and queues that are provided by the multiprocessing library. Process pools are a method of parallelizing across the cores of a multi-core processor. In the case of this project, process pools are utilized so that there can be a frame of video being processed simultaneously on each core, thus allowing for closer to real-time object detection. The channel of communication between the processes are Multiprocessing Queues. The queues function as a shared data space between the processes.

In the tool, the process pool is initialized with as many processes as there are cores in the machines processor. Each process has a function called a worker. The worker function is the function that does the actual computation. Each video frame is, read from the camera, pushed onto an input queue, and then an available worker will takes the data from the queue and run object detection. When object detection is finished it will wait for another image in the input queue. This will continue until the processes in the pool are terminated via the 'Pool.close()' command. For more information, see [12].

4.1.3 OpenCV

OpenCV is an open source computer vision library for Python. In this project, it is used to handle input from the webcam and output to the computer screen. OpenCV is able to start a video directly from the webcam and read it into memory frame by frame using the 'VideoCapture()' function. OpenCV also converts the video frame from BRG color format to RGB and then back again so that the resulting image can be output to the screen. OpenCV also draws bounding boxes around the images to show identified images. Finally, it outputs the bounded images to the monitor.

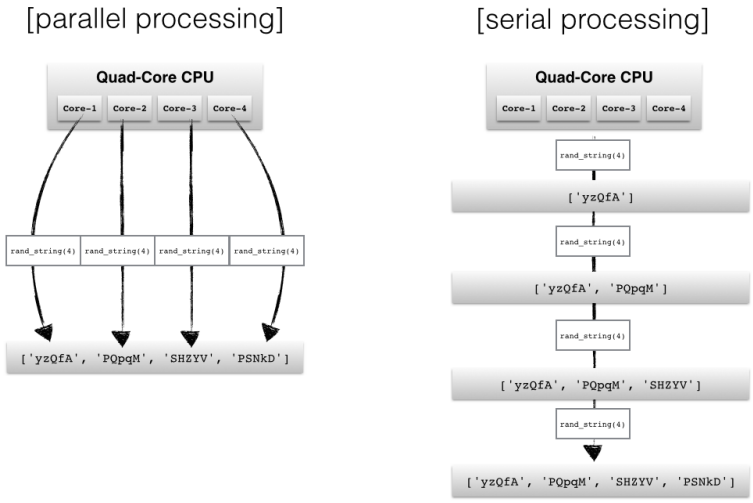

Figure 4.2: A graphic showing how parallel processing can improve the processing time of some task as opposed to serial processing

4.1.4 Tensorflow Object Counting

This is an open source project that I found on Github for the purpose of counting objects in an image that were detected by the Tensorflow object Detection API [17]. The API supports counting all objects detected in an image or counting one specific object; however, it does not support counting only three types of objects detected in an image. I had to make my own edits to the API to enable such functionality. The API also supports speed estimation, but that functionality was not able to be implemented due to the environmental requirements of that function and due to the fact that in order to couple the speed estimation with the basic object counting that I had to edit, would have required the object detection to run twice for each image, which would have compromised the real time analysis. In the future, I would like to be able to merge the features so that they can be used together since speed estimation opens up many new possibilities for this research.

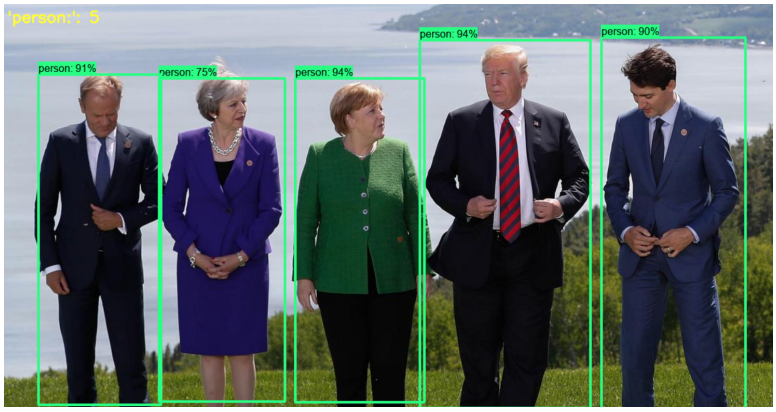

Figure 4.3: An example of the object counting API counting the people detection by Tensorflow object detection in an image. The counter can be see in the top left corner of the image.

4.2 Code Execution

In this section, I will outline the steps the code takes during execution at a high level.

The tool begins by initializing the input and output multiprocessing queues as well as the process pool using 2 as the default number of workers. After that it initializes the video stream from the webcam, the source of which was passed as an argument at execution. Next, a frame is read from the webcam and pushed into the input queue. The frame is then picked up by one of the workers. At which point the worker will initialize a Tensorflow Graph and it will load the COCO object detection model onto the graph. Once the model has been loaded, the graph is used to create a Session. Now, empty Tensors for the image, boxes, confidence scores, object classes, and the number of detections are extracted from the detection graph. At this point object detection is run using the graph and the frame, which at this point has been converted from BGR to RGB color format. The empty tensors are then set to the values that are returned by the detection. Bounding boxes are draw on the frame corresponding to the location of detected objects and the number of

pedestrians and vehicles are written to a csv file along with the time and date the frame was taken. Finally, the frame with the bounding boxes drawn is pushed to the output queue and output to the screen.

Chapter 5

Discussion and Future Work

I was able to collect data for a total of about 12 days. During that time, I collected over 1.5 million data points. Through my data I was able to analyze a significant trends in traffic.

5.1 Summary of Results

5.1.1 Overview of Results

Figure 5.1 shows the complete data collected throughout the duration of the tools 23 hour run time. Although all trends in the data might not be apparent from this presentation, there are some trends that can be observed. Just from simple observation, one can see that the number of vehicles detected outnumbers the number of pedestrians. Another is that traffic in total is much lighter from midnight to 7:00 AM. The total number of pedestrians detected is 140,519 and the total number of vehicles detected is 656,960. Bare in mind that these are not the numbers distinct pedestrians and vehicles, but the number of time a pedestrian or vehicle was detected in an image. However, it does confirm the earlier assumption that there were many more vehicles detected than pedestrians.

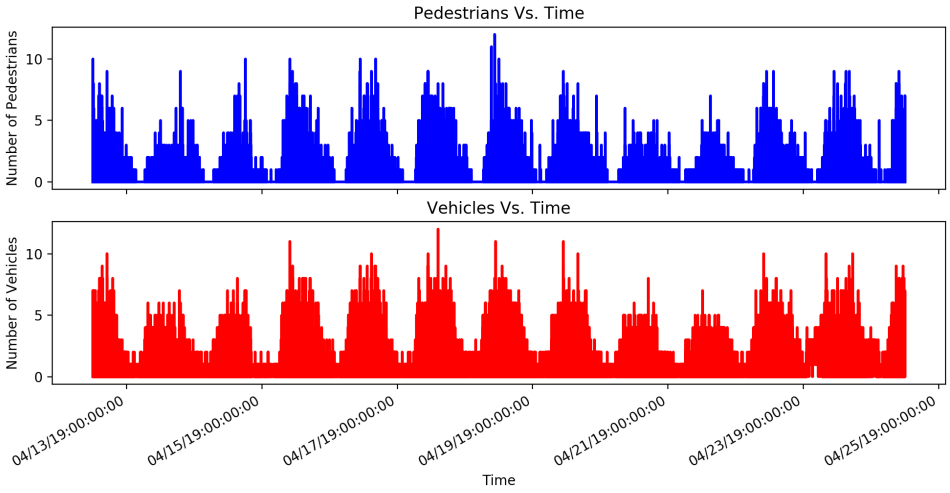

Figure 5.1: The entire data collected during the 12 day run time of the tool

5.1.2 Further Analysis

The average number of pedestrians detected in an hour is 488. The average number of vehicles detected in an hour is 2,281. From midnight to 7:00 during all days there were an average of 9 pedestrians detected per hour and an average of 244 vehicles detected per hour. These numbers confirm that the assumption that the number of pedestrians and vehicles detected during those hours is significantly less than during the day time hours.

Figure 5.2 shows the number of pedestrians and vehicles detected from 7:30 AM to 2:00 PM on Monday April 15 and Monday April 22. As it is plain to see, the data from the two days looks very similar. The number of Pedestrians detected on both days spike at the exact same time. The number of vehicles detected also spike at the same times that pedestrian detections spike. Pedestrian traffic spikes in the ten minutes before 10:00 11:00 and 12:00. This is a repeating trend in the data. This area would be a location where city officials could implement a type of school zone during those times to ensure the safety and ease of crossing of the pedestrians.

Figure 5.3 shows the data from midnight to midnight on April 17. It is plain to see

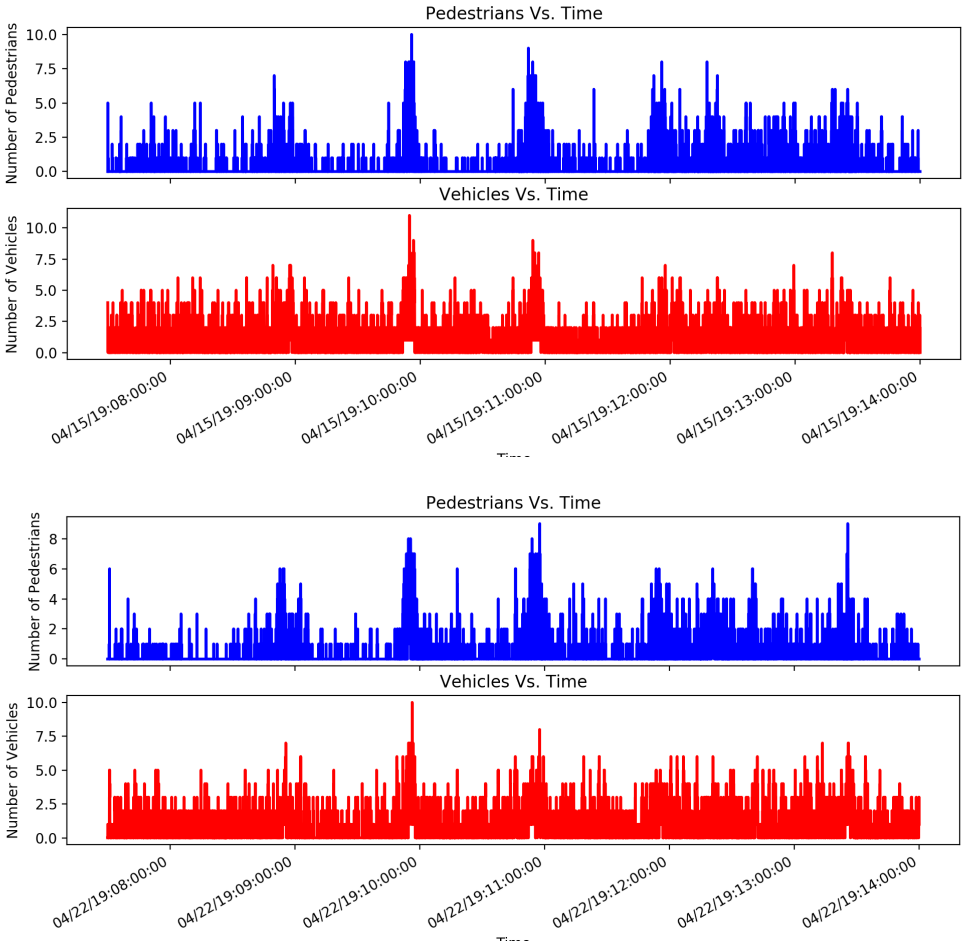

Figure 5.2: These graphs show the pedestrians and vehicles detected from 7:30 AM to 2:00 PM on Monday April 15 and Monday April 22

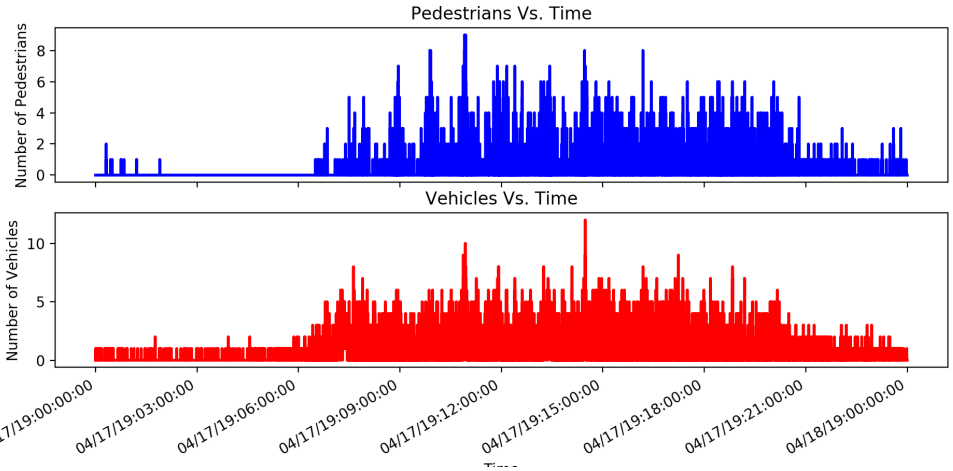

Figure 5.3: This graphic shows the number of pedestrians and vehicles detected from midnight to midnight on Wednesday April 17

that there is not significant traffic until 6:00AM. It is also worth noting that pedestrian traffic spikes before 9:00 10:00, and 11:00 AM just like in the data shown in figure 5.2. Traffic also begins to taper off at 9:00PM.

5.1.3 Discussion of Results

Although the tool ran for a significant period of time, it will take more than 12 days for the data to have meaning in terms of implementation of regulation. If the results obtained by my experimentation could be reproduced and given enough data to show significance, the results could be far reaching. Since the tool is able to detect a reoccurring traffic pattern in an area of road, city officials would be able to better regulate that area of road so that all parties using it could use the road to its full potential if they used the tool. The tool also could effect areas outside of urban planning. The tool could be able to detect delay in vehicle traffic as shown earlier, those delays could be used for traffic delay prediction for routing services like google maps. There are also possible applications outside of roadways. Places that see heavy foot traffic such as air ports, train stations, and retail centers could benefit

from measure the density of foot traffic. That way they could better position advertising, designate time for cleaning and maintenance as to minimize disturbance to patrons, and designate locations for other services.

5.2 Future Work

If I were to continue with my work, I would address the many limitations to my tool. The first is the feature suite. Although a great deal can be learned from the tool in its current state, the technology exists to make the tool much more powerful. Features such as speed estimation and traffic flow are not outside of the realm of possibilities and have been done in other projects [21][22]. This would allow for the analysis of not only the pedestrian and vehicles, but also the pedestrian and their environment. I could even evaluate the safety of different road ways by analyzing the speed of vehicles versus the road's speed limit.

Another major limitation of the tool is the accuracy of its object detection. In the case of my experiment, this issue could have been solved by positioning the camera at a better vantage point, but in the case where there is no better vantage point, the object detection should be sensitive enough to be able to detect its subjects at a reasonable distance. That way an accurate traffic measurement can be obtained. In the future, I would like to spend more time testing different detection models and settings for the tool to see what is the most accurate while still maintaining real time execution.

In the future, I would like to make the tool more user friendly. At the moment, there is no user interface and all data analysis is done manually (in that graphs have to be created and calculation have to be done by the user). The only output of the program is the raw data. If I were to continue work and design the tool so that it can be used for real world applications, it would need to become much more user

friendly.

There are many different types of object detection that are available to me. I would also like to experiment with different methods of object detection. The different papers I read used different methods of object detection to great results. I would like to experiment with different methods. I could also experiment with training my own object detection model with Tensorflow. That way I can create a model that is customized for this unique purpose.

5.3 Conclusion

The tool was a success. It was able to detect patterns in pedestrians and vehicle traffic in an area of road. These trends could have very far reaching effects. If trends can be shown on other roads, then the tool could affect urban planning and allow for the more efficient regulation of shared spaces for pedestrians and vehicles. The tool proved that it can analyze pedestrian and vehicle traffic. If the tool can be applied properly, it will be able to analyze the traffic in areas in pedestrian and vehicles interact and could change the way we look at areas like crosswalks.

Bibliography

[1] Tensorflow api. `https://www.tensorflow.org/api_docs/python/`, May 2018.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] Rolla Almodfer, Shengwu Xiong, Zhixiang Fang, Xiangzhen Kong, and Senwen Zheng. Quantitative analysis of lane-based pedestrian-vehicle conflict at a non-signalized marked crosswalk. *Transportation research part F: traffic psychology and behaviour*, 42:468–478, 2016.

[4] Adrian F Aveni. The not-so-lonely crowd: Friendship groups in collective behavior. *Sociometry*, pages 96–99, 1977.

[5] Abhishika Baruah and AB Kandali. Pedestrian detection based on opencv and tensor flow. 2018.

[6] Cassandra Berry. *Traffic Analysis Tools and Methods : Elements and Consistent Application*. 2014.

[7] Graham Cookson and Bob Pishue. Inrix global traffic scorecard. *INRIX Research, February*, 2017.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[9] Alan J Danker and Azriel Rosenfeld. Blob detection by relaxation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):79–92, 1981.

[10] Piotr Dollár, Serge J Belongie, and Pietro Perona. The fastest pedestrian detector in the west. In *Bmvc*, volume 2, page 7. Citeseer, 2010.

[11] Piotr Dollr, C. Wojeck, B. Schiele, and P. Perona. Caltech pedestrian detection benchmark.

[12] Python Software Foundation. multiprocessing. `https://docs.python.org/2/library/multiprocessing.html#module-multiprocessing.pool`, March 2019.

[13] Stewart Jackson, Luis F Miranda-Moreno, Paul St-Aubin, and Nicolas Saunier. Flexible, mobile video camera system and open source video analysis software for road safety and behavioral analysis. *Transportation research record*, 2365(1):90–98, 2013.

[14] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[15] Bingjie Li, Cunguang Zhang, Bo Li, Hongxu Jiang, and Qizhi Xu. A hardware-efficient parallel architecture for real-time blob analysis based on run-length code. *Journal of Real-Time Image Processing*, 15(3):657–672, 2018.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[17] Mark Daoust Neal Wu, Chris Shallue. Tensorflow object detection api. `https://github.com/tensorflow/models/tree/master/research/object_detection`, 2019.

[18] Ahmet Ozlu.

[19] Mr. Parker Jr and Charles V Zegeer. Traffic conflict techniques for safety and operations: Observers manual. Technical report, United States. Federal Highway Administration, 1989.

[20] Schwartz Retting. Pedestrian traffic fatalities by state. *Washington, DC: Governors Highway Safety Association*, 2018.

[21] Mohammad Shokrolah Shirazi and Brendan Tran Morris. Vision-based pedestrian behavior analysis at intersections. *Journal of Electronic Imaging*, 25(5):051203, 2016.

[22] Mohamed H Zaki and Tarek Sayed. Automated analysis of pedestrian group behavior in urban settings. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1880–1889, 2018.