ZEBRA CROSSWALK DETECTION ASSISTED BY NEURAL NETWORKS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master Of Science in Computer Science

by

Jason Banich

June 2016

© 2016
Jason Banich
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Zebra Crosswalk Detection Assisted by
Neural Networks

AUTHOR: Jason Banich

DATE SUBMITTED: June 2016

COMMITTEE CHAIR: John Seng, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Franz Kurfess, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Phillip Nico, Ph.D.
Professor of Computer Science

ABSTRACT

Zebra Crosswalk Detection Assisted by Neural Networks

Jason Banich

It can be difficult to guide yourself across a crosswalk when your visual capabilities are limited, which can be an everyday issue for someone with impaired vision. This paper aims to alleviate that issue for zebra stripe crosswalks by proposing an algorithm that incorporates multiple properties of zebra stripe crosswalks with a neural network to assist in quickly and accurately identifying a crosswalk in video and pictures taken from a smartphone camera.

This method improves the accuracy of zebra crosswalk detection in images. In a large dataset, it correctly identified 76.5% of zebra crosswalks, while reducing the false discovery rate (q-value) from 20% without using neural networks to 2.21% using this neural network method. Only 2.04% of non-crosswalk images as crosswalks using the neural network method.

ACKNOWLEDGMENTS

I would like to thank my parents, Ann and Steve Banich, my girlfriend, Jene Smith, and my thesis advisor, Dr. Seng, for their support.

TABLE OF CONTENTS

	Page
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER	
1 Introduction	1
2 Background	3
2.1 Types of Crosswalks	3
2.2 Neural Networks	3
2.2.1 History	4
2.2.2 Supervised and Unsupervised Learning	5
2.2.3 Neural Network Basics - Feedforward	6
2.2.4 Backpropagation	6
3 Related Works	8
3.1 Standard Lane Following Algorithm	8
3.2 Crosswalk Detection Approach Using Pixel Bipolarity	9
4 Detecting Zebra Crosswalks	11
4.1 Properties of Zebra Crosswalks	11
4.2 Coughlan Approach	11
5 Using Neural Networks to Detect Zebra Crosswalks	15
5.1 Input Image Constraints	15
5.2 Algorithm Overview	16
5.2.1 Stripelet Detection	16
5.2.2 Neural Network Prediction of Crosswalk Stripelets	18
5.2.3 Crosswalk Detection	21
5.2.4 Crosswalk Boundary Line Drawing	21
6 Results	23
6.1 Methodology	23
6.1.1 Environment	23
6.1.2 Dataset	23

6.2	Neural Network Configuration	24
6.3	Crosswalk Detection	27
6.3.1	Effect of Different Video Angles	28
6.3.2	Effect of Time of Day	30
6.3.3	Zebra Crosswalk Versus Hybrid Crosswalk	30
6.3.4	Effects of Crosswalk Traversal on Accuracy	32
6.3.5	Per Crosswalk Analysis of Results	34
6.3.6	Yellow Crosswalks	35
6.3.7	Challenging Images and Analysis	35
6.3.8	Overall Results	37
6.3.9	Crosswalk Boundary Line Drawing Results	37
7	Future Work	41
7.1	Conversion to Phone App	41
7.2	Improving Stripelet Detection	41
7.3	Crosswalks With Shadows	42
7.4	More Training Parameters for Neural Network	42
7.5	Improving Detection of Crosswalk Boundary Lines	43
7.6	Use Neural Network for Comparing Two Stripelets in Crosswalk De- tection	43
7.7	Performance Improvements	43
7.7.1	Multithreading	44
7.7.2	Finding the Smallest Image Size Necessary	44
8	Conclusion	45
	BIBLIOGRAPHY	46
	APPENDICES	
A	Crosswalk Results Tables	49

LIST OF TABLES

Table	Page
6.1	Neural Network results over stripelets in final dataset 25
6.2	Results of crosswalk detection categorized by video angle 29
6.3	Results of crosswalk detection categorized by time of day 30
6.4	True positive percent results of crosswalk detection categorized by type of crosswalk 32
6.5	Results summary by video/crosswalk 34
6.6	Results of crosswalk detection with/without neural networks. Im- provement of 937% in q-value 37
A.1	Crosswalk identification results from neural network training with different hidden layer configurations 49
A.2	Results for each crosswalk by name with neural network enabled . . 50
A.3	Results for each crosswalk by name with neural network disabled . . 51

LIST OF FIGURES

Figure	Page
2.1 A few types of crosswalks	4
2.2 Real world samples of the types of crosswalks	4
2.3 Neural network classification result[19]	5
2.4 Example of a small neural network with weights labeled	6
3.1 Steps of detecting a zebra crosswalk using Bipolarity and Projective Invariant: (a) gray scale image, (b) bipolarity of each segmented region, (c) original image highlighting the largest detected region, (d) extracted feature points shown on region	10
4.1 Matched top and bottom pairs via the Coughlan Approach [6] . . .	12
4.2 Scatter plot of vertical Y location and vertical width of stripelets via Coughlan Approach [6]	13
4.3 Coughlan Results [6]	14
5.1 Input image blurred and converted to grayscale	16
5.2 Sobel derivative of the image - positive values colored red, negative values colored blue	16
5.3 Lines detected using Hough transform on image from figure 5.2 - blue is potential crosswalk top lines, red is potential bottom lines	17
5.4 All of the stripelets detected after running the constraints on all the input lines from figure 5.3	17
5.5 Left to right: (a) Stripelets used to detect crosswalk. (b) Points taken from the edges of the stripelets	22
5.6 The green line is a manually drawn edge line, and the blue line is the estimated line	22
6.1 Final neural network configuration showing input layers, hidden layers, and output layers	26
6.2 Examples of the low angle video that garnered poor results	28
6.3 All five angles which videos were taken of the crosswalks. Left to right: (a) Low Angle (b) Left Side (c) Standard (d) Right Side (e) High Angle	28

6.4	Poorly painted hybrid crosswalk	31
6.5	True positive results grouped by frame of video (frame 1 to frame 20) as crosswalk is traversed	33
6.6	True and false positive results grouped by frame of video and showing number of detected stripelets (frame 1 to frame 20)	33
6.7	Example frames from 'NonC Output Night 13,' the video with the highest amount of false positives	35
6.8	Example frames from 'NonC Output Night 11,' the video with the second highest amount of false positives	36
6.9	Example frames from 'Night hybrid 3,' the crosswalk with the worst true positive rate	36
6.10	Shows the true positive rates per crosswalk with and without the neural network enabled	38
6.11	Shows the false positive rates per video with and without the neural network enabled	39
6.12	Orange and red dots are the right and left points used. Green is the manually drawn boundary. Blue is the estimated boundary. Left to right: (a) Left evaluation metric of 38, right 34. (b) left: 178, right: 135. (c) left: 235, right: 90	39
6.13	Orange and red dots are the right and left points used. Green is the manually drawn boundary. Blue is the estimated boundary. The evaluation metric on the right side is 400, which is considered the high range of acceptable	40

Chapter 1

INTRODUCTION

Crossing the street can be dangerous for visually impaired individuals because orienting oneself is difficult without being able to use a visual marker. Visually impaired people must use non-visual cues such as listening for cars stopped at red lights and trying to walk in front of them. Vision processing could be applied to recognize the crosswalk to assist the user with limited vision. In order to increase the safety of blindly crossing the street, a phone app could allow the user to hold up their phone and get feedback about the location of the crosswalk. As they cross the street, they would receive feedback as to whether they were going off course. Developing a fast and reliable metric for detecting the crosswalk is the first step towards developing a vision processing phone app to aid the visually impaired.

There have been some investigations of zebra crosswalk detection, but none so far have used neural networks [6][12][20]. Neural networks are formed by a number of interconnected nodes that require training. Once trained, they are configured to a specific application to predict outcomes based on the inputs. Neural networks are being used for an increasing number of applications where patterns are difficult to decipher manually. Although the training process can be time consuming, neural networks run quickly once trained, which is a large benefit to any application where speed is a factor.

Figure-ground segmentation has been used to separate the objects from the background scene. This technique can be applied to zebra crosswalks to allow the detection of potential stripelets in the image using geometric parameters [6]. Adding neural networks to this discovery method allows us to use parameters to evaluate whether stripelets are part of a crosswalk, a task which would be difficult manually. This paper

aims to use neural networks to improve the reliability of zebra crosswalk detection methods, by offering a different approach to the problem.

Chapter 2

BACKGROUND

2.1 Types of Crosswalks

There are many types of crosswalks (See figures 2.1 and 2.2 for some examples), the main two are zebra stripe crosswalks (sometimes called 'continental') and standard two line crosswalks. Two line crosswalks are more commonly used due to their simplicity, but zebra crosswalks are often used in more vulnerable intersections due to their benefits in visibility for drivers [14]. Vision processing for the driving lane of a car has been done many times, and there is no substantial difference between that and a two line crosswalk. As such, algorithms that work for these are mostly interchangeable.

2.2 Neural Networks

Neural networks are a type of machine learning that have been used recently for many vision applications such as image classification (See figure 2.3) and pattern recognition [19], because they are useful for datasets with hard-to-detect patterns. Neural networks function using an interconnected network of neurons that work together to process the training data in order to discover a pattern or classification. Neural networks are well suited for many data sets that include lots of information that could be difficult to manually train a computer to categorize. One odd use case for neural networks is that they have been used to help determine which signals from a rat's brain should be used to trigger prosthetic limbs to move by measuring results of brain probes [4].

Figure 2.1: A few types of crosswalks

Figure 2.2: Real world samples of the types of crosswalks

Neural networks are given a large amount of training data with known outputs. The network then runs over the dataset repeatedly, adjusting weighted parameters every iteration to reduce the amount of error in the predictions. Backpropagation of the data is also used in order to assist in training. Once the neural network is trained, the application can feed in new values and receive a prediction of the output.

2.2.1 History

The origins of neural networks originate in 1943, when McCulloh and Pitts published “A logical calculus of the ideas immanent in nervous activity.” They showed that simple neural networks could compute a wide array of functions with the same underlying method. A computer that worked as a neural network was created by

Figure 2.3: Neural network classification result[19]

Frank Rosenblatt and others, with a focus on pattern matching. In 1986, Rumelhard, Hinton and Williams published "Learning Internal Representation by Error Propagation," which popularized the backpropagation method of finding weights, and brought neural networks to the forefront [16], [22].

2.2.2 Supervised and Unsupervised Learning

Machine learning algorithms can be supervised or unsupervised, each with their own uses. Unsupervised algorithms do not have a labeled output, but can attempt to cluster the outputs. An example of this is trying to distinguish human faces from backgrounds, with a goal of clustering features together. Reinforcement learning is a type of unsupervised learning where the environment rewards good behavior, which leads to that good behavior being repeated. One use case for unsupervised learning is games, because there cannot be a known output. Winning, losing, and gaining points are good examples of reinforcement.

Supervised algorithms like backpropagation, require a training set with known

Figure 2.4: Example of a small neural network with weights labeled

outputs. Backpropagation works towards a goal of reducing the error so that the predicted outputs are close to the known outputs. Supervised algorithms are useful when the outputs are rigidly defined[17].

2.2.3 Neural Network Basics - Feedforward

The quintessential example of a neural network is a feedforward neural network. This network allows information to move forward only, from the inputs to the outputs. An example of a basic feedforward neural network is shown in figure 2.4. A series of weights are assigned to every connection, the inputs are fed through the network, multiplied by each respective weight, summed, and then fed to the next layer until the output values are generated. To train a feedforward network, one must either manually adjust the weights, or come up with a mathematical formula to do so. One such method is backpropagation[18].

2.2.4 Backpropagation

Backpropagation is a method of training neural networks by adjusting weights backwards through the network. It starts with weights for every link, and runs through

them. The error on the output is calculated versus the expected output using the squared error function and the sum of the total error among all output nodes. Then, working from the outputs backwards, one works with all of the weights feeding into a prior node one at a time using partial derivatives and the chain rule. The reason the partial derivative is used is because the hidden layers all jointly affect the output nodes' error and must be taken into account. Once all of the weights are adjusted, the error should decrease and then the process is repeated until the error level is reduced to an acceptable level [10].

Chapter 3

RELATED WORKS

3.1 Standard Lane Following Algorithm

As autonomous cars evolve into reality, there is a need for algorithms to assist them in navigating and staying inside the lines. These algorithms detect lines on the pavement, and instructing the car to stay in the center. This is similar to a two line crosswalk, the goal is to keep the user between the two lines. However, with a crosswalk, the goal is to keep them inside the crosswalk, not necessarily centered. One algorithm for lane following is as follows:

1. Convert the image to grayscale
2. Crop to the center portion of the image
3. Identify the edges using an edge detection algorithm, and then draw the edges onto a new image
4. Apply Hough line detection to find shapes in the edges [21]
5. Delete extraneous lines that were detected
6. Apply and draw the discovered lines on the main image

By following these steps, the image is processed and the lanes are returned, which can then be used to direct the car to remain inside the lane [5].

3.2 Crosswalk Detection Approach Using Pixel Bipolarity

Kyoto Institute of Technology's Uddin and Shioyama created a method of detecting zebra crosswalks using their geometric properties and the bipolarity of the images [20]. Their paper focuses on grouping large areas together that have high values of dark and/or light inside of them. Also, they use the projective invariant aka cross product to determine if the areas of light and dark in their image fit the pattern of a crosswalk. They then use the Fischer criterion to extract feature points in the image where the change from white to black (and vice versa) occur.

The steps are as follows:

1. Convert to grayscale and split the image into 16x16 pixel groups.
2. Find regions that are strongly bipolar by merging pixel groups that have a distance ratio within a threshold. The distance ratio is a measure of how similar the regions intensities are to each other in regards to bipolarity. A merged image is shown in figure 3.1 section (b).
3. Once grouped, discard regions that don't exhibit enough bipolarity.
4. Refine the segmentation by allowing larger groups to absorb smaller groups inside of them.
5. Check where the region is in the image. If it is far to the left or right, classify it as such. If the bottom of the region is too high up in the image, discard it.
6. Use a Fourier transform to check if the crosswalk angle is too extreme. If it is, throw it out.
7. Use Fischer criteria to find the feature points where the intensity changes drastically, determine that there are at least seven (which means there are four or

Figure 3.1: Steps of detecting a zebra crosswalk using Bipolarity and Projective Invariant: (a) gray scale image, (b) bipolarity of each segmented region, (c) original image highlighting the largest detected region, (d) extracted feature points shown on region

more crosswalk stripes in the image), and then check the projective invariant criteria to see if it passes or is thrown out[20].

8. Repeat for each region until a match is found or all regions are exhausted.

This method had decent results, with a high true positive rate (74 / 79), and zero false positives out of 34 other images. One downside of their method is that the image is required to have a minimum of four crosswalk stripes in the image to detect it, which means the majority of the crosswalk must be in the frame. Theoretically this method will work with a moderate amount of the crosswalk covered up (by other pedestrians or cars) as long as there are enough feature points in a vertical line in at least one area to detect the crosswalk.

DETECTING ZEBRA CROSSWALKS

4.1 Properties of Zebra Crosswalks

Zebra crosswalks have many properties that could be used to help identify them, one example would be the lengths of the crosswalk's edges. The edges are well defined in most images of crosswalks, so they can be collected rather easily, and have a distinct shape. Other qualities include the fact that painted stripes are typically homogeneous in the center in terms of pixel intensity. Zebra crosswalk stripes are also spaced a consistent distance apart. The horizontal lines in the crosswalk are all exactly parallel, and the side edges of each side of a zebra crosswalk should be parallel as well and point towards the vanishing point. Some of these properties will be recognized and used in later sections to uniquely identify zebra stripe crosswalks.

4.2 Coughlan Approach

In Coughlan and Shen's 2006 paper [6] about identifying zebra crosswalks, they propose a method for recognizing zebra stripe crosswalks using figure-ground segmentation. Their method starts by converting the image to grayscale, shrinking it, and then blurring it slightly. Then, the derivative of image intensity is taken in the Y direction to find the pixels that look like top and bottom edges. The derivative assigns top lines a negative value, and bottom lines a positive value. These edges are then greedily grouping together into larger line segments. The edges are then matched up with other edges that might fit together to be part of the same crosswalk stripe. They define a candidate stripe fragment feature as a 'stripelet.' A stripelet is defined as the combination of two line segments with the following properties:

Figure 4.1: Matched top and bottom pairs via the Coughlan Approach [6]

1. The upper and lower segments have polarities consistent with a crosswalk stripe (upper is above lower).
2. The two segments are roughly parallel.
3. The segments have sufficient overlap in the X coordinate range.
4. The vertical width of the stripelet must be between 2 and 70 pixels (determined value from their test images).

After matching up pairs of lines into stripelets, many stripelets are found, most of which are not part of the crosswalk (see figure 4.1).

After extraction, unary and binary cues are used to determine if the stripelets are figure or ground. Their unary cue uses the fact that stripes lower in the image would be wider if they are part of a crosswalk. Coughlan and Shen found a relationship between the vertical width and vertical Y coordinate, as shown in the scatter plot in figure 4.2, a line drawn with a negative slope along a few points with low width and Y value follows the lines that represent stripelets from the crosswalk. They then use

Figure 4.2: Scatter plot of vertical Y location and vertical width of stripelets via Coughlan Approach [6]

a line fitting procedure to try and find an envelope line that fits that model. Once that line is found, they can use it to evaluate other stripelets. They look for longer stripelets that are close to the envelope in order to classify them as potentially a crosswalk stripelet. Additionally, a binary cue is used when the cross ratio test is applied to pairs of stripelets to check that all four lines are approximately parallel. Finally, the cross ratio is also used to verify that adjacent stripelets are within an acceptable distance from each other.

They ran this algorithm multiple times on each image with different envelope lines, and used the result that had the highest belief value of being correct. Their results are shown in figure 4.3. These results were generated in few seconds per image.

In their next paper in 2008 [12], they implemented this algorithm in Symbian C++ on a Nokia N95 cell phone and obtained a true positive rate of 72% and a false positive rate of 0.5% on detected lines being predicted correctly as part of a crosswalk or not over a test set of 30 crosswalk images and 60 non-crosswalk images. Running on their 332 MHz ARM 11 processor running on 320x240 resolution pictures, they

Figure 4.3: Coughlan Results [6]

were able to obtain a processing speed of three frames per second with optimizations, compared to their processing speed on a PC with unoptimized code, and they believed the code could be further optimized as well.

USING NEURAL NETWORKS TO DETECT ZEBRA CROSSWALKS

5.1 Input Image Constraints

Some image constraints were chosen for this scope of this paper because they would have been difficult to incorporate into this algorithm and were not necessary for a proof of concept implementation.

- All images are taken horizontally.

As detailed in future work, odd image angles can be very easily accounted for using the sensors of a phone, and the frame being horizontal makes image processing easier. Being able to assume that all true stripelets are horizontal removes them from the sample early on for being too vertical, or having a slope that is too far from the horizon.

- All images contain crosswalks unaffected by shadows.

Shadows in a crosswalk create extremely strong lines that can interfere with the detection algorithm, as such they will be left for future work.

- All crosswalks are white.

Yellow and white crosswalks have different pixel intensities, which may cause issues with the neural network training parameters, so in order to negate that issue, yellow crosswalks were excluded.

Figure 5.1: Input image blurred and converted to grayscale

Figure 5.2: Sobel derivative of the image - positive values colored red, negative values colored blue

5.2 Algorithm Overview

5.2.1 Stripelet Detection

Following the method in the figure-ground paper [6], the frame is blurred slightly, and then converted to grayscale (Shown in figure 5.1).

The Sobel derivative is taken in the Y direction, leaving just the changes in the Y direction. The resulting pixels are either positive or negative, which signifies whether they are going from dark to light, or light to dark, giving us pixels that can be either the bottom or the top of the crosswalk. These are potential edges of crosswalks in the image (See figure 5.2).

A standard method of Hough transform for line detection is then used to discover

Figure 5.3: Lines detected using Hough transform on image from figure 5.2 - blue is potential crosswalk top lines, red is potential bottom lines

Figure 5.4: All of the stripelets detected after running the constraints on all the input lines from figure 5.3

lines (figure 5.3).

The generated lines are either defined as potential crosswalk top edges, or bottom edges. These potential edges are matched up against other edges (top vs bottom edges) to determine if they fit certain criteria to be considered stripelets. A stripelet is defined as an area bounded by a top and bottom line that follows certain constraints. The constraints used in Coughlan’s figure-ground crosswalk paper [6] are the same as used here: the top line is above the bottom line in the image, the two lines’ slopes are very similar, the vertical width of the generated stripelet is between a defined cutoff value, and the two lines have sufficient X overlap. If all of these are true, then it is returned as a stripelet (See figure 5.4).

5.2.2 Neural Network Prediction of Crosswalk Stripelets

At this point, the algorithm has generated a list of stripelets that are potentially part of a crosswalk. The inputs used for the neural network are a few parameters extracted from the image. The parameters are chosen because they possess characteristics that are believed may contribute towards helping identify a stripelet. A few sample parameters used are the stripelet's vertical/horizontal width, variance of the pixel intensity, and the lengths of the top and bottom lines. These numbers are calculated and then fed into the trained neural network, resulting in a very fast prediction as to whether or not that stripelet is part of a crosswalk.

Eleven different characteristics of the stripelets were chosen for use as the neural network parameters. The parameters are as follows:

1. Bottom line length

- Description: The measure of the length of the bottom of the stripelet.
- Reasoning: Crosswalk stripelets are to have certain lengths, so this measure could help identify the validity of the stripelet.

2. Top line length

- Description: The measure of the length of the top of the stripelet.
- Reasoning: This value is similar to the bottom line length, and should pair well with the bottom length to give more information about the stripelet, as the two values are correlated.

3. Difference between top line length and bottom line length

- Description: The difference in length between the top and the bottom of the stripelet.

- Reasoning: Feeding in the difference between these measures explicitly could assist the neural network in using the difference. The bottom ought to be wider and the difference between the two should typically be a reasonably small number.

4. Vertical stripelet width

- Description: The width of the stripelet vertically between the top and bottom lines.
- Reasoning: The vertical width of the stripelet is a useful factor because stripelets should have a certain vertical width that correlates with their top and bottom lengths, which would help define them as being part of a crosswalk.

5. Horizontal stripelet width

- Description: The width of the stripelet horizontally; the average length of the top and bottom lines.
- Reasoning: Similar to the vertical stripelet width, this parameter should correlate with the other parameters.

6. Vertical stripelet location

- Description: The vertical stripelet location (Y value of the stripelet center).
- Reasoning: The vertical stripelet location should help to correlate with other parameters to give a better prediction. For example, one might expect that a crosswalk stripelet higher in the image would be smaller, which would correlate this value with vertical and horizontal stripelet width.

7. Variance of the stripelet pixel values

- Description: Pixel intensity variance within the stripelet.

- Reasoning: The pixel intensity variance should be lower for well painted stripelets, and high for stripelets that contain many different pixel values. This would be indicative of a non-crosswalk stripelet.
8. Standard deviation of the stripelet pixel values
- Description: Square root of the variance.
 - Reasoning: This measure may produce the same insights as the variance, and may allow a similar but simpler metric for the neural network to use.
9. Standard deviation of the pixel intensity of the area surrounding the stripelet
- Description: An area surrounding the stripelet is defined and the standard deviation of the pixel intensity in the area is found.
 - Reasoning: The area around the stripelet should have a low standard deviation because it would be regular concrete without any lines for a crosswalk stripelet.
10. Standard deviation of the area surrounding the stripelet divided by standard deviation of the stripelet
- Description: The standard deviation of the area surrounding the stripelet divided by standard deviation of the stripelet.
 - Reasoning: This ratio may help to give the neural network information about the relationship between the two measures.
11. Average stripelet pixel intensity divided by average pixel intensity of the area surrounding the stripelet
- Description: Average pixel intensity of the stripelet divided by the average pixel intensity of the area surrounding the stripelet.

- Reasoning: The stripelet pixel intensity should be much greater inside the stripelet than the area surrounding it, so this value could assist in finding which ratios are indicative of crosswalk stripelets or not.

5.2.3 Crosswalk Detection

The stripelets that are predicted by the neural network to be part of a crosswalk are then checked one by one against the other stripelets through a manual feature comparison process. The compared features of the two stripelets are their average slopes, the horizontal pixel overlap, the vertical width of the stripelets, and the Y range overlap. The average slopes should be similar, the horizontal pixels should have a large amount of overlap, the vertical width of the stripelet which is lower in the image should be wider, as well as the horizontal width, and the Y ranges should have minimal overlap, if any. If two of the predicted crosswalk stripelets satisfy all of these constraints, then the image is declared to contain a zebra crosswalk. If all of the constraints fail to pass, then the image is not labeled a crosswalk.

5.2.4 Crosswalk Boundary Line Drawing

In order to attempt to draw the boundary lines for the crosswalk, the edges from the stripelets that were positively identified as being part of a crosswalk are used. Three points are taken from the outer edges of the stripelets to assist in predicting the boundary as shown in figure 5.5.

RANSAC line fitting is applied to these points in order to fit a line that is the approximation of the edge line, shown in figure 5.6.

Figure 5.5: Left to right: (a) Stripelets used to detect crosswalk. (b) Points taken from the edges of the stripelets

Figure 5.6: The green line is a manually drawn edge line, and the blue line is the estimated line

6.1 Methodology

6.1.1 Environment

This project was done entirely in OpenCV 3.0 using their C++ libraries. The computer used for this project was running Windows 8.1 on a Intel i5-2500K CPU with a maximum clock speed of 4.1 GHz. The OpenCV libraries are widely used and support a large amount of languages and platforms, including Apple iOS, Android, OpenCL, CUDA, as well as Mac, Linux and PC, which means that the code is easily ported to most any platform [2]. The input images were all read from saved video files on disk, and fed one by one into this project's algorithm.

6.1.2 Dataset

The data used for the results were a selection of videos from multiple white crosswalks and streets without crosswalks taken at and around street intersections. The videos were taken at multiple times of day ranging from early morning to early night. For each crosswalk, five videos were taken going the same direction across the crosswalk. The videos were a high angle, a low angle, walking on the left side, on the right side, and a walking in the center. Videos were given categories: time of day, angle of video, type of crosswalk, color, crosswalk title, etc. Crosswalks were marked to be used only for training the neural network, and other crosswalks marked only to be used for evaluating results. The training set contained three hybrid crosswalks, six zebra crosswalks, and ten videos that did not include crosswalks. The result testing

set contained three hybrid crosswalks, nine zebra crosswalks, and eighteen videos that did not include crosswalks. All data in the results section is generated from videos that were never used in the training process of the neural network. Over 14,000 stripelets were marked by hand as part of a crosswalk or not part of a crosswalk for use when training the neural network, and over 18,000 manually marked stripelets were used for the output. Each crosswalk video only used 20 frames, whereas videos without crosswalk may have used more. Hybrid crosswalks were included as well because they were working with the algorithm as well.

6.2 Neural Network Configuration

For this project, neural networks are used for the prediction of stripelets as either being part of a crosswalk or not. Neural networks require a set number of hidden layers with a set number of neurons, so some experimentation was done to determine these numbers. As shown in figure 6.1, for this project, the final result was one hidden layer of 14 neurons. A supervised feedforward network trained using backpropagation was used for this project. This neural network configuration is one of the simpler configurations, and remains very effective. An unsupervised neural network configuration would not be as optimal for this same exact data because the outputs are known, allowing a supervised method to train more efficiently. A larger training set would likely be required for an unsupervised network.

There is no real consensus on exactly how many hidden layers and neurons should be used in the perfect neural network configuration. The different rules of thumb have been that no more than one hidden layer is typically needed, and the number of neurons should be between the number of inputs and the number of outputs, and layers must have more than one neuron[9]. With those being the guidelines, one hidden layer with a number of neurons mostly ranging from the number of inputs to

Table 6.1: Neural Network results over stripelets in final dataset

Number of Stripelets	True Pos Rate	False Pos Rate	Q-Val
19461	77.61%	3.57%	0.1663

the number of outputs was chosen with a range from 2 to 22 neurons in the hidden layer. Subsets of these training parameters were tried as well, with none having better results than using all eleven parameters.

The neural network training of the crosswalk stripelets used many different input parameters to predict the output. The metric used for evaluating the parameters was finding the true positive and false negative rates, as well as the false discovery rate (also known as q-value). The true positive rate is calculated by dividing the number of crosswalk stripelets that were predicted correctly as stripelets over the total number of crosswalk stripelets. The false positive rate is calculated by dividing the number of non-crosswalk stripelets predicted as stripelets out of the total number of non-crosswalk stripelets. The q-value is found by dividing the number of false positives by the number of stripelets that were predicted to be part of a crosswalk. Table 6.1 shows the neural network prediction rates over the entire result set. Only 16.6% of the positively predicted stripelets were incorrect.

As shown in appendix table A.1, some configurations were better than others, but none by much. The configuration with the 14 neuron hidden layer had the lowest false positive rate of 3.12%, so it was chosen as the configuration to be used for the rest of this project. This fits with the original assumption that the number of neurons should be between the number of inputs and outputs as shown in figure 6.1.

Figure 6.1: Final neural network configuration showing input layers, hidden layers, and output layers

6.3 Crosswalk Detection

Once the neural network has finished culling stripelets, the remaining stripelets are then fed through the crosswalk detection algorithm. The results are again categorized by type of crosswalk, time of day, etc.

Figure 6.2: Examples of the low angle video that garnered poor results

Figure 6.3: All five angles which videos were taken of the crosswalks. Left to right: (a) Low Angle (b) Left Side (c) Standard (d) Right Side (e) High Angle

6.3.1 Effect of Different Video Angles

There were five different angles used for the videos of the crosswalks, with 340 frames of video at each angle across many crosswalks. Examples of each angle are shown in figure 6.3. The low angle had worse correct prediction rates than the others, as shown in table 6.2. The reason for this is that the cell phone camera was angled too low to capture enough stripelets for a positive identification. A few examples are given in figure 6.2. The poor results from the bad video angle show that in an actual use case, that angle must be disallowed. The user could be trained to hold the cell phone camera at the proper angle, and if they did not, the sensors would allow the application to notify the user to adjust the camera angle. Because of this, the videos taken at that downward angle will be removed from the overall dataset.

Table 6.2: Results of crosswalk detection categorized by video angle

Camera Angle	True Positive Rate	False Positive Rate
Low Angle	45.67%	0.00%
Left Side	77.00%	1.00%
Standard	77.00%	0.33%
Right Side	76.67%	1.33%
High Angle	73.33%	0.00%

Table 6.3: Results of crosswalk detection categorized by time of day

Time of Day	True Positive Rate	False Positive Rate
Morning	79.09%	2.91%
Midday	75.00%	2.05%
Night	75.44%	1.90%

6.3.2 Effect of Time of Day

Each crosswalk was categorized by time of day, morning, midday and night, to see if that had any effect on the prediction rates. The results are shown in table 6.3. There were 300 crosswalk frames for each different time of day. The results for each time of day were rather close together, with a slight increase for morning. This might have to do with the lighting causing the image to be more easy to categorize during the morning hours. This data shows that different lighting caused by different time of day did not seem to be a strong factor in the results.

6.3.3 Zebra Crosswalk Versus Hybrid Crosswalk

The true positive percent results of zebra stripe and hybrid crosswalks detection are shown in table 6.4. There were nine zebra stripe crosswalks in the dataset, and six hybrids. The least accurately predicted crosswalk was in the hybrid dataset, leading to a much higher standard deviation. The zebra crosswalks were predicted more consistently, leading to a lower standard deviation and higher true positive percents. The training data was biased towards zebra crosswalks, with three hybrids trained versus nine zebra crosswalks. As shown in the table 6.4, the zebra crosswalk median true positive rate only dropped about 3%, with the introduction of the neural network,

Figure 6.4: Poorly painted hybrid crosswalk

while the hybrid median dropped almost 24% after neural network application. The hybrid crosswalks observed in the area these videos were taken appeared to more frequently have paint irregularities than the zebra crosswalks, as shown in Figure 6.4, potentially due to different maintenance practices by the municipality, which could also explain the discrepancy. One of the neural network parameters regarding the pixel intensity of the area surrounding the stripelet would also be different for hybrid versus zebra, because hybrids would have a higher value due to the white lines on the sides being part of their surroundings. It is likely that adding more hybrid crosswalks to the training dataset would improve their results.

Table 6.4: True positive percent results of crosswalk detection categorized by type of crosswalk

Type	Neural Network Enabled	Min	Avg	Median	Max	STD
Hybrid	YES	0.00%	57.92%	62.50%	85.00%	30.68%
Hybrid	NO	15.00%	76.46%	86.25%	98.75%	31.18%
Zebra	YES	52.50%	88.06%	95.00%	98.75%	15.30%
Zebra	NO	62.50%	92.22%	97.50%	100.00%	11.85%

6.3.4 Effects of Crosswalk Traversal on Accuracy

As the user walks across the crosswalk, the view of the crosswalk changes and the algorithm needs to continue guiding them. For each crosswalk, 20 frames equidistant throughout the video were used. The expected result of this is that there would be a decrease in true positive rate as the user progresses because there will be fewer stripelets in the image to work with. The data shown in figure 6.5 shows exactly that. The neural network made the rate go down as one would expect, but the drop seemed to increase as the end of the crosswalk was reached. This may be because the training data may be more biased to earlier frames, since the early frames are guaranteed to have more stripelets. In figure 6.6, we can also see that the number of stripelets in each image as the video went on became fewer as the user crossed the crosswalk, but the predictions of the stripelets stayed mostly consistent throughout.

Figure 6.5: True positive results grouped by frame of video (frame 1 to frame 20) as crosswalk is traversed

Figure 6.6: True and false positive results grouped by frame of video and showing number of detected stripelets (frame 1 to frame 20)

Table 6.5: Results summary by video/crosswalk. Crosswalks have true and false positives, while non-crosswalk videos only have false positives

	Cwalk	Minimum	Average	Median	Maximum	STD
True Pos Rates	YES	0.00%	76.00%	82.50%	98.75%	26.52%
False Pos Rates	YES	0.00%	0.59%	0.00%	3.75%	1.26%
False Pos Rates	NO	0.00%	1.67%	0.00%	12.00%	3.45%

6.3.5 Per Crosswalk Analysis of Results

For this section, the crosswalk videos were grouped by their respective crosswalks, and the non-crosswalk videos were kept as separate groups. The results of this are shown in the appendix tables A.2 and A.3, and the summary results are shown in table 6.5. Crosswalks have both true and false positive rates depending on whether they were predicted as being a crosswalk by correct reasoning or not. Videos not containing a crosswalk only have false positives because they cannot contain a crosswalk. There were a couple outliers in each category (crosswalk vs not) that are included in the dataset, and will be investigated in a later section.

Overall, the average true positive rate of the crosswalk videos was $76\% \pm 13.42\%$ with a standard deviation of 26.52%. The average false positive rate of crosswalk images was $0.59\% \pm .64\%$ with a standard deviation of 1.26%. The average false positive rate of the non-crosswalk videos was $1.67\% \pm 1.75\%$ with a standard deviation of 3.45%. The true positive predictions of crosswalks had a rather high standard deviation, which was heavily affected by the one outlier crosswalk, and the false positives were more tightly grouped with a lower standard deviation.

Figure 6.7: Example frames from 'NonC Output Night 13,' the video with the highest amount of false positives

6.3.6 Yellow Crosswalks

In order to determine how well yellow crosswalks worked with white crosswalk training data, two yellow crosswalks were tested, one being a standard zebra and the other a hybrid. The hybrid scored 24% true positive rate, and the zebra scored 66%, both out of 80 frames. Those results are promising enough to conclude that yellow crosswalks will likely work well with this approach, and would probably improve with yellow crosswalks being included in the training data, due to the differing intensity values between the two colors affecting neural network parameters.

6.3.7 Challenging Images and Analysis

The video with the highest amount of false positives (12%) was 'NonC Output Night 13,' which was a street being crossed not at an intersection, and with bike lanes going across both sides. The bike lanes and sidewalk were incorrectly matched together, which ended up passing the neural network and the validation as a crosswalk. A couple examples are shown in figure 6.7, one can see the identified stripelets. More neural network parameters may be able to help remove these from the stripelet set, perhaps something relating to the vertical pixel intensity.

Another video with a high amount of false positives (6.67%) was 'NonC Output Night 11,' which is a of a parking lot with many horizontal lines painted throughout.

Figure 6.8: Example frames from 'NonC Output Night 11,' the video with the second highest amount of false positives

Figure 6.9: Example frames from 'Night hybrid 3,' the crosswalk with the worst true positive rate

A few pictures are shown in figure 6.8. These examples aren't typically what one would expect to see if they were trying to cross the street, but they are lines that look similar to a zebra crosswalk. They could potentially be filtered out by using some sort of expected width vs Y axis level in the image.

One crosswalk specifically had the worst results by far, 'Night hybrid 3,' shown in figure 6.9. None of the eighty frames were correctly identified in this crosswalk, whereas the next lowest crosswalk had 53% of frames correctly identified. The neural network failed to correctly identify enough stripelets in the majority of the images to pass the crosswalk validation. This crosswalk had very odd pavement coloring, which may have contributed to why it failed the neural network. Potentially this issue could be solved by adding more training data and including more odd crosswalks such as this one.

Table 6.6: Results of crosswalk detection with/without neural networks. Improvement of 937% in q-value

Neural Network Enabled	True Positive Rate	False Positive Rate	q-value
NO	91.00%	24.88%	0.200
YES	76.51%	2.04%	0.021

6.3.8 Overall Results

Over all of the videos, including 1200 crosswalk frames and 970 non-crosswalk frames, and using the neural network to cull stripelets before detection, a q-value of .021 is obtained for crosswalk predictions. When not using the neural network to cull the stripelets, the q-value is .200 (see table 6.6). Using the neural network garners an improvement of 933% of this value. In simpler terms, using neural networks, out of 1000 positively predicted images, only 21 would be false positives versus 200 without using the neural predictions. Figure 6.10 shows that for every video, the neural network does drop the positive identifications, but figure 6.11 shows that false positive rates dropped drastically as well for every single case.

6.3.9 Crosswalk Boundary Line Drawing Results

A subset of 180 images were selected to test the crosswalk boundary drawing and had their boundary lines manually drawn. A metric was needed in order to compare the correct line with the estimated line. The metric used involved taking multiple points that were equidistant along the estimated line and finding the normal distance to the manually drawn line. These distances were then summed up to give a general gauge of how close lines were together. Exactly identical lines would have a difference value of zero, while more distinct lines would have greater values. Some example images with

Figure 6.10: Shows the true positive rates per crosswalk with and without the neural network enabled

their evaluation result values are shown in figure 6.12. Overall, the results were rather promising, with the vast majority of lines from images with a recognized crosswalk being close to correct with a median measure of 82 and only 16% of the lines being over the threshold of 400. As shown in figure 6.13, 400 was a generous measure, because even at 400, the lines are still close enough. If the stripelets are predicted properly, and a crosswalk is recognized, the edges of the stripelets are almost guaranteed to be good metrics of the crosswalk edges.

Figure 6.11: Shows the false positive rates per video with and without the neural network enabled

Figure 6.12: Orange and red dots are the right and left points used. Green is the manually drawn boundary. Blue is the estimated boundary. Left to right: (a) Left evaluation metric of 38, right 34. (b) left: 178, right: 135. (c) left: 235, right: 90

Figure 6.13: Orange and red dots are the right and left points used. Green is the manually drawn boundary. Blue is the estimated boundary. The evaluation metric on the right side is 400, which is considered the high range of acceptable

Chapter 7

FUTURE WORK

7.1 Conversion to Phone App

The project was written entirely in the C++ libraries provided by OpenCV, which can be ported easily to a mobile app without many changes. Once the algorithm has access to phone sensors, there are opportunities to implement improvements. The current algorithm assumes that the photos are taken parallel to the horizon, but with access to a phone's sensors, the image frames could be adjusted so that they are guaranteed to be in the correct horizontal orientation. The accelerometer and gyroscope sensors could also be used to help track movement as the user crosses a crosswalk. Using sensors, anything above the horizon line could be discarded [11]. Other mobile technologies such as Project Tango's 3-D mapping hardware [1] could be used as well to assist with spatial awareness and other factors such as detecting obstructions that must be avoided. Once the program is converted to a mobile app, there are many interesting ways that it could progress.

7.2 Improving Stripelet Detection

The stripelets that are detected and fed into the neural network could have been more accurate. Different pixel grouping methods to group the detected edges into lines might result in improved stripelets, which would lead to a more accurate input for the neural network and improve detection.

7.3 Crosswalks With Shadows

As this project did not cover crosswalks with shadows, future work could be done to adapt the algorithm so it can work properly with crosswalks that have shadows inside of them. The current algorithm has difficulties with shadows; one issue is that the Hough line transform might have issues because shadows form strong lines, and another being that some of the neural network training parameters might be drastically different with shadows inside of the stripelets.

One option might be shadow removal algorithms that would remove the shadows so that the main algorithm doesn't have to account for them at all. This could be a good solution because it would require modifying the current algorithm, but might not be the fastest. Shadow removal algorithms are quite common, so there would be many different options [7]. Another potential solution would be to modify the current algorithm to handle the shadows. This method would likely involve modifying the Hough lines grouping, as well as using different or modified training parameters for the neural network, and require additional training data including crosswalks with different types of shadows at various times of day.

7.4 More Training Parameters for Neural Network

The portion of the algorithm that uses neural networks to predict whether or not stripelets are part of the crosswalk would benefit from more investigation into new metrics that would aid the neural network. Investigating new parameters is not a costly procedure in terms of time, and it is not difficult to discover whether a training parameter is working well or not after a single attempt, so it would be worth the investment.

7.5 Improving Detection of Crosswalk Boundary Lines

Ideally, the algorithm could be improved in detection of the edges of the crosswalk in order to assist the user in navigating across the intersection. One large improvement would be to trend the lines and values along multiple sequential frames to give a more accurate measurement. More experimentation could lead to better methods of doing this as well.

7.6 Use Neural Network for Comparing Two Stripelets in Crosswalk Detection

Currently, the algorithm is using a few different metrics in order to predict whether two stripelets could be a part of the same crosswalk or not. A neural network could be applied to this decision instead, in order to potentially garner a better result. The same parameters that are being manually used currently would be a good starting point for neural network parameters, as well as the other parameters currently being used by the neural network for stripelet prediction.

7.7 Performance Improvements

The algorithm contains some areas where the performance could be improved. In this proof of concept, speed isn't as important, so the algorithm is not as optimized as it could be. If it is ported to a phone app, the optimizations would become necessary for a real world usage scenario.

7.7.1 Multithreading

One specific improvement would be to take advantage of multiple CPU cores in order to process multiple frames at the same time. This would improve the throughput and allow greater accuracy due being able to quickly to average the results of frames that are immediately next to each other. The current implementation running on a decent PC can run at around 1.6 frames per second, and would scale linearly with the number of cores, so if one core is currently 1.6 frames per second, then two cores would be 3.2 frames per second, thus multithreading would give a large performance boost for this application. Multithreading would also be extremely beneficial for the crosswalk side edge detection, because running multiple sequential frames at the same time would allow a more accurate prediction.

7.7.2 Finding the Smallest Image Size Necessary

Another improvement would be to try to find the smallest image size necessary for the algorithm to detect the position of the crosswalk. The smaller the image is, the faster the processing time will be because there are fewer pixels for every operation. This could be discovered by running the algorithm on smaller and smaller crosswalks and finding out when the predictions start degrading to an unacceptable level. The performance would drastically improve, because as the image dimensions are cut in half, the number of pixels are divided by four, which scales almost directly with the runtime because many operations are performed on each pixel, and those operation's runtimes would divided by four.

Chapter 8

CONCLUSION

This project incorporated neural networks into a process of recognizing zebra stripe crosswalks in images. This process may be effective in aiding a visually impaired user across a crosswalk.

Neural networks are used in many image processing platforms, but they haven't yet been applied to recognizing zebra crosswalks in this way.

Using the same dataset, a substantial improvement in false discovery rate (937%) was seen when comparing the same code using neural networks versus without them. The false discovery rate is important because the higher it is, the more confidence can be given to a positive prediction. There was an increase in runtime due to the calculations required for the neural network inputs, but optimizations could reduce that impact. The dataset used was large and diverse. The results presented in this paper show that using neural networks is a promising tool in zebra crosswalk detection, and is well worth further investigation.

BIBLIOGRAPHY

- [1] Google's project tango. <https://www.google.com/atap/project-tango/>. (Accessed on 02/08/2016).
- [2] Opencv code @ONLINE. <http://opencv.org/platforms.html>. (Accessed on 03/03/2016).
- [3] ATI NormalMapper. <http://ati.amd.com/developer/tools.html>.
- [4] G. Capi, T. Takahashi, K. Urushiyama, and S. Kawahara. Robots controlled by neural networks trained based on brain signals. In *Machine Learning for Signal Processing, 2009. MLSP 2009. IEEE International Workshop*, pages 1–6, September 2009.
- [5] M. Conti. Vision system - lane detection assignment. <http://www.marco83.com/work/84/84/>, June 2009.
- [6] J. M. Coughlan and H. Shen. H.: A fast algorithm for finding crosswalks using figure-ground segmentation. In *In: Proc. 2nd Workshop on Applications of Computer Vision, in conjunction with ECCV*, page 2, 2006.
- [7] C. Fredembach and G. Finlayson. Simple shadow removal. In *18th International Conference on Pattern Recognition*, pages 832–835, 2006.
- [8] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.
- [9] J. Heaton. *Introduction to Neural Networks for Java, 2nd Edition*. Heaton Research, Inc., 2nd edition, 2008.

- [10] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference*, pages 593–605 vol.1, 1989.
- [11] V. Ivanchenko, J. Coughlan, and H. Shen. Crosswatch: A camera phone system for orienting visually impaired pedestrians at traffic intersections. In *Proceedings of the 11th international conference on Computers Helping People with Special Needs, ICCHP '08*, pages 1122–1128, Berlin, Heidelberg, 2008. Springer-Verlag.
- [12] V. Ivanchenko, J. Coughlan, and H. Shen. Detecting and locating crosswalks using a camera phone. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference*, pages 1–8, June 2008.
- [13] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, New York, NY, USA, 2002. ACM Press.
- [14] A. McGrane and M. Mitman. An overview and recommendations of high-visibility crosswalk marking styles. Technical report, Federal Highway Administration, August 2013.
- [15] nVidia Melody. http://developer.nvidia.com/object/melody_home.html.
- [16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.
- [17] R. Sathya. Comparison of supervised and unsupervised learning algorithms for pattern classification. In *International Journal of Advanced Research in Artificial Intelligence*, pages 34–38 Vol. 2, No. 2. 2013.

- [18] D. Svozil, V. Kvasnicka, and J. Pospichal. Introduction to multi-layer feed-forward neural networks, June 1997.
- [19] C. Szegedy. Building a deeper understanding of images. <http://googleresearch.blogspot.com/2014/09/building-deeper-understanding-of-images.html>, September 2014.
- [20] M. S. Uddin and T. Shioyama. Bipolarity and projective invariant-based zebra-crossing detection for the visually impaired. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, pages 22–22, June 2005.
- [21] M. van Ginkel, C. L. Hendriks, and L. van Vliet. A short introduction to the radon and hough transforms and how they relate to each other, 2004.
- [22] N. Yadav. *An introduction to neural network methods for differential equations*. Springer, 2015.

APPENDICES

APPENDIX A

CROSSWALK RESULTS TABLES

Table A.1: Crosswalk identification results from neural network training with different hidden layer configurations

Neurons in Hidden Layer	True Pos Rate	False Pos Rate	q-value
2	75.83%	5.45%	0.238
3	76.82%	5.94%	0.251
4	76.73%	5.71%	0.244
5	77.23%	4.78%	0.212
6	76.82%	4.33%	0.196
7	78.16%	4.40%	0.196
8	77.42%	4.63%	0.206
9	78.30%	4.16%	0.187
10	77.97%	3.76%	0.173
11	79.47%	4.05%	0.181
12	78.41%	3.79%	0.173
13	79.97%	3.50%	0.159
14	78.73%	3.12%	0.147
15	77.61%	3.38%	0.159
16	79.53%	4.50%	0.197

17	78.82%	4.56%	0.200
18	78.82%	3.59%	0.165
19	80.08%	4.69%	0.203
20	79.78%	3.91%	0.175
21	79.50%	4.09%	0.182
22	79.47%	4.05%	0.181

Table A.2: Results for each crosswalk by name with neural network enabled

Name	Frames	True Pos Rate	False Pos Rate
Afternoon hybrid 1	80	58.75%	3.75%
Afternoon hybrid 3	80	81.25%	0.00%
Afternoon zebra 1	80	96.25%	0.00%
Afternoon zebra 3	80	52.50%	1.25%
Afternoon zebra 5	80	82.50%	0.00%
Morning hybrid 1	80	66.25%	0.00%
Morning hybrid 3	80	56.25%	3.75%
Morning zebra 1	80	98.75%	0.00%
Morning zebra 3	80	93.75%	0.00%
Morning zebra 5	80	77.50%	0.00%
Night hybrid 1	80	85.00%	0.00%
Night hybrid 3	80	0.00%	1.25%
Night zebra 1	80	97.50%	0.00%

Night zebra 3	80	98.75%	0.00%
Night zebra 5	80	95.00%	0.00%
NonC Output Afternoon 4	50	-	0.00%
NonC Output Afternoon 5	50	-	0.00%
NonC Output Afternoon 6	50	-	3.33%
NonC Output Afternoon 7 FC-30	30	-	0.00%
NonC Output Afternoon 8 FC-30	30	-	0.00%
NonC Output Afternoon 9 FC-30	30	-	0.00%
NonC Output Morning 2	50	-	0.00%
NonC Output Morning 3	50	-	6.67%
NonC Output Night 10 FC-200	200	-	3.00%
NonC Output Night 11 FC-30	30	-	12.00%
NonC Output Night 12 FC-100	100	-	0.00%
NonC Output Night 13 FC-50	50	-	0.00%
NonC Output Night 5	50	-	0.00%
NonC Output Night 6	50	-	0.00%
NonC Output Night 7	50	-	0.00%
NonC Output Night 8	50	-	0.00%
NonC Output Night 9	50	-	0.00%

Table A.3: Results for each crosswalk by name with neural network disabled

Name	Frames	True Pos Rate	False Pos Rate
------	--------	---------------	----------------

Afternoon hybrid 1	80	85.00%	7.50%
Afternoon hybrid 3	80	96.25%	0.00%
Afternoon zebra 1	80	97.50%	1.25%
Afternoon zebra 3	80	62.50%	32.50%
Afternoon zebra 5	80	97.50%	2.50%
Morning hybrid 1	80	98.75%	0.00%
Morning hybrid 3	80	76.25%	10.00%
Morning zebra 1	80	98.75%	1.25%
Morning zebra 3	80	93.75%	0.00%
Morning zebra 5	80	86.25%	0.00%
Night hybrid 1	80	87.50%	3.75%
Night hybrid 3	80	15.00%	22.50%
Night zebra 1	80	97.50%	0.00%
Night zebra 3	80	100.00%	0.00%
Night zebra 5	80	96.25%	2.50%
NonC Output Afternoon 4	50	-	8.00%
NonC Output Afternoon 5	50	-	2.00%
NonC Output Afternoon 6	50	-	14.00%
NonC Output Afternoon 7 FC-30	30	-	0.00%
NonC Output Afternoon 8 FC-30	30	-	16.67%
NonC Output Afternoon 9 FC-30	30	-	0.00%
NonC Output Morning 2	50	-	6.00%

NonC Output Morning 3	50	-	22.00%
NonC Output Night 10 FC-200	200	-	19.00%
NonC Output Night 11 FC-30	30	-	46.67%
NonC Output Night 12 FC-100	100	-	45.00%
NonC Output Night 13 FC-50	50	-	68.00%
NonC Output Night 5	50	-	4.00%
NonC Output Night 6	50	-	12.00%
NonC Output Night 7	50	-	2.00%
NonC Output Night 8	50	-	2.00%
NonC Output Night 9	50	-	38.00%