
Final Project Write Up

COMP4102A

Automotive Safety Suite

Authors:

CONNER BRADLEY – 101073585
CHRISTIAN BELAIR – 101078744
ADAM PAYZANT – 101082175

Carleton University

February 5, 2020

Abstract

Contents

1	Introduction	2
2	Background	2
2.1	Gaze Detection and Tracking	2
2.2	Pedestrian Detection and Tracking	2
2.3	Road Sign Tracking	2
3	Approach	2
3.1	Gaze Tracking	2
3.2	Pedestrian Tracking	2
3.3	Road Sign Tracking	3
4	Results	3
4.1	Gaze Tracking	3
4.2	Pedestrian Tracking	3
4.3	Road Sign Tracking	4
5	List of Work	5
5.1	Gaze Tracking	5
5.2	Pedestrian Tracking	5
5.3	Road Sign Tracking	5
5.4	GUI Integration	5
5.5	Shared Component Integration	5
6	GitHub Page	5

1 Introduction

The Automotive Safety Suite (ASS) is a collection of safety features which try to solve distracted driving problems using computer vision. The ASS has three main components: Gaze Tracking, Pedestrian Tracking, and Road Sign Tracking. Gaze tracking determines where eyes exist in the driver's camera view and derives gaze in order to determine where it is the driver is looking. Pedestrian Tracking involves trying to identify pedestrians in the application's road view. This component is meant to identify where within the view pedestrians are if they exist. The last component is Road Sign Tracking where using the same road view as Pedestrian Tracking it detects and identifies where within the video frame road signs are if they exist. The main goal is to have a suite of features that work together to avoid distracted driving by notifying the driver whenever their gaze is not within an acceptable range of oncoming road signs or pedestrians.

2 Background

2.1 Gaze Detection and Tracking

Gaze detection and tracking has many positive implications for driver safety, namely to detect if a driver is distracted or drowsy. The US department of transportation found that in years 2011-2015 an overall 2.5% of fatalities were caused by drowsy driving [4], and for distracted driving in 2018 it was found that 8% of fatalities were distraction-affected [5]. Currently there are a wide variety of well-established and novel techniques for gaze tracking as found by a survey by Chennamma and Xiaohui [2].

2.2 Pedestrian Detection and Tracking

Pedestrian detection and tracking is an important aspect in driver safety. Drivers must be aware when one or more pedestrians are moving around their vehicle as to ensure their safety. Stimpson, Wilson, and Muelleman research into pedestrian fatalities shows that from 2005 to 2010 the fatality rate of 116.1 per 10 billion vehicle miles driven had increased to 168.6 in 2010 [7]. OpenCV includes an implementation of the Histograms of Oriented Gradients (HOG) which can identify a person within an image or video. This creates a basis for identifying pedestrians. The implementation details in terms of human detection are found in this paper[3]

2.3 Road Sign Tracking

Road signs are vital to road safety. Stop sign violations accounted for approximately 70% of all crashes.[6] In addition, in some areas, speed limits are lowered in unexpected areas pose great risk for a driver, even when they're striving to follow the speed limit. While much research into this area is focused on using machine learning, some computer vision exclusive implementation have been studied.[1]

3 Approach

3.1 Gaze Tracking

3.2 Pedestrian Tracking

Pedestrian tracking is achieved through a single function which takes in a *cv::Mat* object that represents the current frame of the video and returns a custom data structure called *PedTrackingResult*. The function initializes an instance of OpenCV's *HOGDescriptor* object. With the help of OpenCV, this object can be passed a human detection method which can be passed to the *HOGDescriptor*. Once this method is passed to the *HOGDescriptor*, a clone of the original frame is produced. From here, the process of identifying humans in the frame is done by the work of the *HOGDescriptor* using its *detectMultiScale()* function after converting the input frame from four channels to one channel as needed by the function. This function produces a vector of *cv::Rect* objects representing the bounding boxes for human-shaped objects in the frame and the weights associated with these bounding boxes. From here each *cv::Rect* object in the frame is drawn onto the frame with its associated weight. The resulting frame and *cv::Rect* vector is returned to the image pipeline in order to display the frame and to calculate driver attentiveness.

With the returned vector we first calculate the centre of the width for each *cv::Rect* object. We then compare this center pixel against which third of the main video width the bounding box's centre lies in. We then compare the current eye state to this pedestrian object state and see if the

driver is looking at the same third. This process is applied to each bounding box until the end of the list has been reached. Since there are three thirds of the window, if there are an even number of objects detected in each window then we can say that the minimum rate for attentiveness is a third of the total detected pedestrians. This means that if a driver pays attention to third with the most number of pedestrians then the minimum threshold needed for the classification to be passable for driving is $\frac{1}{3}$.

3.3 Road Sign Tracking

The road sign tracking is achieved through a single function which takes in a *cv::Mat* that must be the type “CV_8U4C”, and returns a *SignTrackingResult* containing a *cv::Mat* with the rectangles drawn on and a *std::vector<cv::Rect>*.

The road tracking is performed by getting the reds, yellows, and whites within the image. This is done by converting the image to HSV and masking off its specific value ranges. Then, a canny edge detector is performed on each mask, with a dilation performed on the result to make the edges easier to detect. On the dilated result, *cv::findContours()* is performed to find all continuous borders in the image. With the contours and their hierarchy find, the approximate polygonal curves in the image are found, where the edges are then counted. If the edges count match a sign’s count in its respective color, the shape is approved and is included in the return.

The results of the sign tracking could be implemented in the same manner as the pedestrian, but the number of false positives made it a poor addition to detecting alertness.

4 Results

4.1 Gaze Tracking

For our gaze tracking component, we investigated a few different approaches to gaze tracking as documented in the survey of gaze tracking performed by



Figure 1: Gaze tracking, straight. In the top left-Figure 2: Gaze tracking, right. In this example hand corner of the window is the current eyethe left bounding box mistook the dark region tracking state of hair as a pupil

4.2 Pedestrian Tracking

The original testing results when testing with a laptop camera were a little lackluster. For the most part, the use of the HOG descriptor using this wecam resulted in inaccurate readings of human objects within the frame. Mostly it would detect human-like objects in the negative space between objects. An example of this would be when someone was standing reasonably far away from the camera and the HOG descriptor would identify human-shaped objects in the armpits of the subject standing in front of the camera or in the surrounding spaces. Another example is when the subject was standing in front of the camera with their hand turned sideways. This issue arose because the quality of the camera used produced a significant amount of noise in each frame resulting in changes in gradients.



(a) Detected Armpit Error

(b) Detected Hand Error

Figure 3: Errors produced using a noisy webcam

Later, testing started to use video files to be more in line with how we wanted to achieve this project. The video used was of a motorcyclist who had a camera attached to their helmet and was riding around while driving by careless pedestrians. When viewing the results of the video file, the accuracy of the HOG descriptor was much more in line with what we had hoped. Most if not all of the pedestrians were identified when within an acceptable range. Whenever the pedestrian tracking component produced a false negative, the cases that generated such errors were those where either a pedestrian was too far from or too close to the camera, and whenever the pedestrian was partially occluded. Both of these conditions are not out of the ordinary since being too close to the camera results in some occlusion since the pedestrian will be partially in frame. When too far away, the gradients in the HOG descriptor become harder to determine if they are human-shaped objects which is normal in nature. Partial occlusion of a pedestrian in the frame is also normal as parts of the gradient data is lost due to occlusion.

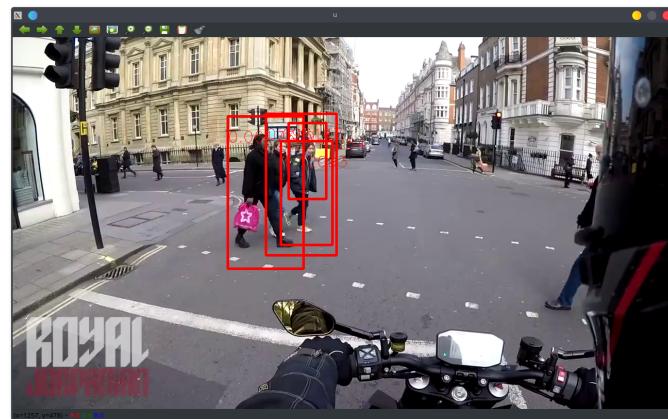
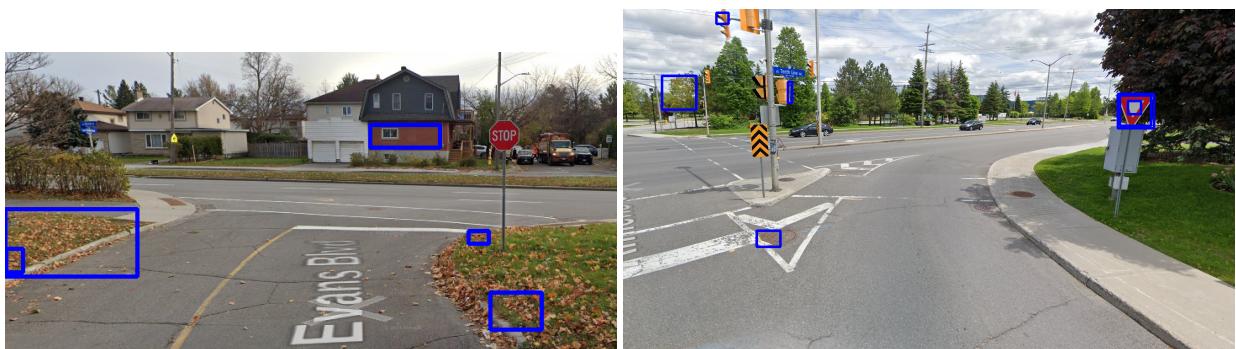


Figure 4: Correct detection in video containing pedestrians

4.3 Road Sign Tracking

The contour edge counting method implemented is very prone to false positives. This is because, for example, any red shape with eight edges would be seen as a stop sign. Additionally, this method is highly dependent on optimal conditions. Were environment or camera inaccuracies to significant offset the color of signs, they would fail to be detected as it's looking for specific color ranges.



(a) The method used is prone to failing to identify signs
 (b) The edge counting method is prone to false positives

The sign tracking was initially going to be handled by training an SVM and utilizing a HOG.

This attempt ultimately failed due to the limitations on hand as well as limitations of machine learning. The data sets obtained did not have enough variety for most signs, meaning it would be a poor detector of individual signs. Additionally, the variety of road signs prevented the model from being trained as a single class.

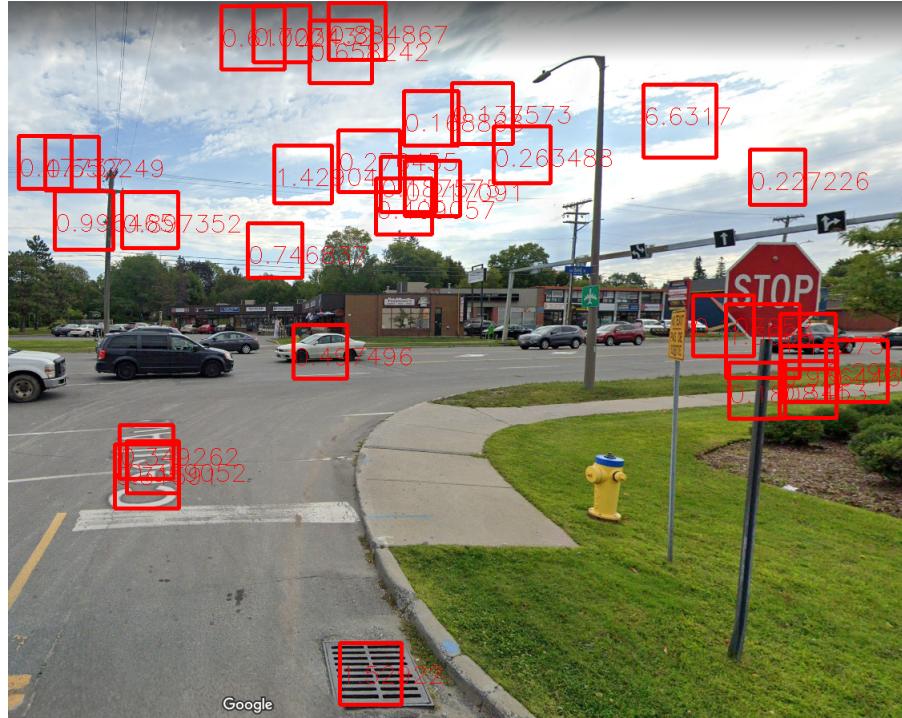


Figure 6: The single-class SVM model was highly inaccurate

As a result of the limitations of SVM and the datasets, the model could not achieve >30% accuracy and ultimately had to be dropped in favor of edge counting.

5 List of Work

5.1 Gaze Tracking

- Conner Bradley

5.2 Pedestrian Tracking

- Christian Belair

5.3 Road Sign Tracking

- Adam Payzant

5.4 GUI Integration

- Conner Bradley
- Christian Belair
- Adam Payzant

5.5 Shared Component Integration

- Conner Bradley
- Christian Belair
- Adam Payzant

6 GitHub Page

<https://github.com/ChristianBelair/COMP4102-Project>

References

- [1] Tarik "Ayaou, Mourad Boussaid, Karim Afdel, and Abdellah Amghar". Improving road signs detection performance by combining the features of hough transform and texture. *arXiv preprint arXiv:2010.06453*, 2020.
- [2] HR Chennamma and Xiaohui Yuan. A survey on eye-gaze tracking techniques. *arXiv preprint arXiv:1312.6410*, 2013.
- [3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. 1:886–893 vol. 1, 2005.
- [4] US Department of Transportation. Crash stats: Drowsy driving. *Traffic Safety Facts*, DOT HS 812 446, 2015.
- [5] US Department of Transportation. Research note: Distracted driving. *Traffic Safety Facts*, DOT HS 812 926, 2018.
- [6] Richard A. Retting, Helen B. Weinstein, and Mark G. Solomon. Analysis of motor-vehicle crashes at stop signs in four us cities. *Journal of Safety Research*, 34(5), 2003.
- [7] Jim P. Stimpson, Fernando A. Wilson, and Robert L. Muelleman. Fatalities of pedestrians, bicycle riders, and motorists due to distracted driving motor vehicle crashes in the u.s., 2005–2010. *Public Health Reports*, 128(6):436–442, 2013. PMID: 24179255.