Automated Detection of Vehicles for the Visually Impaired to Cross Roads

Sangeet Satpathy
Henry M. Gunn Senior High School
780 Arastradero Road, Palo Alto, CA 94306
sangeet.satpathy@gmail.com
ss36498@pausd.us

Abstract

Due to the surge in birth rates during the 1950s and 60s, the United States has started to host an aging population. With a rise in the elderly population, technological assistance for the visually impaired has become especially important to pursue. Individuals with visual impairment currently often have to rely on other people to cross roads safely. An app that could scan the road for moving vehicles and use aural cues to alert the user would be beneficial to these individuals. This paper looks at an approach using the YOLOv5 Object Detection Model and the DeepSORT object tracking algorithm to try and register moving vehicles. The paper reflects on the benefits and limitations of such an approach to this app.

1 Introduction

As humans age, their vision usually deteriorates over the years. According to the National Institutes of Health, it is projected that the legally blind population will severely increase over the next few decades [1]. Naturally, the impairment of vision makes it hard for an individual to complete their activities independently. Among these activities is navigating and crossing streets. Currently, legally blind individuals use several tactics to help them cross the streets, including listening to vehicular sounds, using an assistive cane to find the curb of the sidewalk, and making patterns out of pedestrian sounds.

1.1 Gap in Research

With the rising power of artificial intelligence, it is no surprise that developers have begun creating apps that help the visually impaired to cross roads safely. However, the apps currently in the marketplace mainly address urban roads. For example, OKO, an app released in 2023, helps legally blind individuals at pedestrian crosswalks [2]. When a legally blind person arrives at a crosswalk, they can use the app to detect the crosswalk signal; the app will alert the user when to cross using aural cues. These apps are certainly helpful, as urban areas have a lot of cars and a lot of distracting background noise. However, crossing residential roads can prove to be challenging due to their infrequent traffic patterns — and without any apps helping them in this, they have to sacrifice their independence and rely on others.

An app that allows pedestrians to quickly scan both sides of a residential road for moving vehicles, and notify them of when to cross the road, would be a suitable filling for this gap. Since this app needs to be accessible on smartphones, the detection of moving vehicles cannot be done using distance sensors; instead, this app must do this detection by using simple phone features, such as the phone's camera.

Preprint. Under review.

2 Methodologies

For this project, I made a program that attempts to detect moving cars from various real-life video clips. This program was made to investigate the feasibility of using such technology for aiding the visually impaired to cross residential roads. The idea is to model an app that would just need the user to turn the phone left, turn it right, and turn it left again to check for moving cars, similar to how most people use their eyes to check for cars before crossing a road. Testing the program in various situations provides an understanding of the benefits and shortcomings of such an approach. This allows future developers of such technologies to take into account the shortcomings and adjust for it.

In order to detect moving vehicles, a custom object detection model was trained using the fifth iteration on the YOLO [3] object detection algorithm, YOLOv5 [4]. In order to differentiate moving vehicles from stationary vehicles, I added a feature for object tracking, in which the computer keeps track of individual cars as separate "objects". This allows the computer to assign IDs to each vehicle, and, as a result, detect movement for each vehicle. For this, I used DeepSORT, an object tracking algorithm that uses a variety of mathematical models to keep track of cars [5]. I implemented this into my code by using the open-source Python DeepSORT module [6].

Figure 1: The car with an ID of 57 is an example of a successful detection of a car that is obscured. However, there are several other cars in the background that are not detected.

Once I completed the program, I conducted systematic testing at various locations, including parking lots and residential roads. Specifically, to represent a visually impaired person trying to cross a road, I closed my eyes and turned my phone to look left, look right, and look left again. Several iterations of testing allowed me to analyze the failures of the program.

The link to all the code I used for this project can be found in the link below.

`https://github.com/sangeetsatpathy/AutomatedDetectionOfVehicles_ForVisuallyImpaired/tree/main`

3 Analysis

This paper will divide the analysis into two parts: the performance of the object detection algorithm (YOLOv5) and the performance of the object tracking algorithm (DeepSORT).

3.1 Object Detection

A closer inspection of the videos reveals that YOLOv5 does a fairly good job detecting present vehicles. However, while it does detect some obscured vehicles, it is unable to detect them all (See Figure 1).

The failed detections are a problem in specific situations — for example, a situation in which a car is not fully visible but is turning onto the main road (See Figure 2).

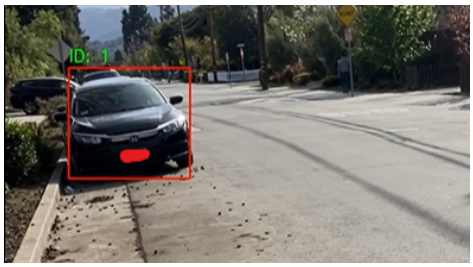

Figure 2: The black car behind the car with ID 1 is not detected, even though it is turning right onto the road and poses a threat to the user.

3.2 Object Tracking

Examining the performance of the object tracking, moving cars do not appear to be moving when they are at further distances from the camera (see Figure 3). This is problematic because by the time the user walks on the road, they run the risk of being hit by the car.

The center of the bounding box appears to be stationary due to the camera angle. This is something we experience with our eyes; our eyes can usually sense that a vehicle is becoming larger relative to the surroundings, and is thus moving. However, while the object detection model is quite accurate in detecting the relative location of the bounding box, it is often inaccurate when it comes to detecting bounding box widths (see Figure 3).

Note that this issue is less problematic when we are observing cars coming from the right. This is because cars to the right of the app's user will be on the opposite side of the road, so the camera can see the length of the car well (as opposed to only the front of the car). As a result, the vehicles will not appear to be stationary.

(a) Frame A (b) Frame B

Figure 3: The pictures above show two detection frames within a few seconds of each other. The bounding box of Vehicle 16 does not move significantly; In addition, in Frame A, the bounding box on Vehicle 16 is much larger than it should be - meaning that we cannot rely on the width of the bounding box to determine if the vehicle is moving.

3.3 Potential Solutions

The challenge mentioned in the last section could be overcome if there were some way to measure the distance of the car precisely using just a phone's features.

One possible solution would be to use the camera's focal length to gauge the distance of a vehicle. To find the focal length of the camera, we can set up an object of a known width at a known distance away. Using the width of the bounding box of the ensuing detection, we can set up a pair of similar triangles as shown in Figure 4 below. Using the focal length, we could ideally determine the distance of any object that we know the size of.

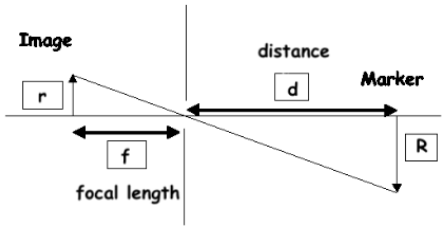

Figure 4: The setup to determine a camera's focal length. See references for image source [7]

However, the problem is, with vehicles, their widths and heights vary in size, meaning that unless we know the specific width of that vehicle, this method cannot be used to determine the distance. Perhaps if the object detection could differentiate between vehicle models, this method could be used. Nevertheless, more research is needed to see if this is viable.

Another possible way to tackle these problems would be to not use object tracking at all — instead; we can have some way to identify whether cars are on the road or parked on the side by their position on the road. This could be done by having the app recognize the "boundaries" of the road; if a car is on the very edge of a road, it is probably parked — and if it is in the middle of the road, then it is probably driving around and is a threat to somebody trying to cross the road.

However, such an approach would not work in situations as shown by Figure 5 (below). More research would need to be done on the possibilities of using such an approach.

4 Conclusion

To summarize the analysis presented in the previous section, this study has found three main limitations of using such technology to make a "look left, look right" mobile application.

Firstly, the camera finds it difficult to detect obscured cars (as noted in Figure 3). Secondly, the object detection model does not accurately detect the bounding box width (and is very volatile to change). Third, cars in the distance do not appear to be moving much, which can pose a risk if they are moving fast — which is a major limitation of this kind of technology.

In order to get more accurate object detection, more research will be needed. In this paper, I have specifically looked at the effectiveness of YOLOv5 using the Udacity Self-Driving Car Image Dataset [8]. For future research, comparing different object detection algorithms trying different vehicle datasets is necessary.

In order to bypass the last limitation, more research will be needed as well. Perhaps some other method of accurately determining the distance of a car would remedy this limitation. If this project were to be continued, the proposed ideas in Section 3.3 would be implemented and tested for viability.

Figure 5: Both cars in this scenario are moving; but they would be regarded as being on the side of the road by such a program.

Other alternative approaches would have to be developed as well. This paper serves as the groundwork for further research to be done in developing an app that serves the visually impaired to help them cross roads safely and efficiently.

Acknowledgements

This research paper was written as a part of the Advanced Authentic Research (AAR) class at Henry M. Gunn Senior High School, taught by Ms. Rachael Kaci. Ms. Kaci took me through the steps required to write this research paper, including analysis of existing research, data collection, examination of my findings, and more. Her support was essential in writing this research paper.

This project was mentored by Mr. Emmanuel Mayssat, a cloud engineer at JPMorgan Chase & Co. He leads the AI/ML group within the Silicon Valley office. His work involves leveraging cloud platforms to implement Machine Learning models, focusing on areas such as Natural Language Processing, Computer Vision, and Predictive Analytics. Mr. Mayssat has offered his technical expertise at every step of this project, and this paper would neither be complete nor thorough without his invaluable guidance.

Additionally, I would also like to thank my parents and sister for their emotional support and encouragement.

References

[1] "Visual Impairment, Blindness Cases in U.S. Expected to Double by 2050." *National Institutes of Health*, U.S. Department of Health and Human Services, 19 May 2016, www.nih.gov/news-events/news-releases/visual-impairment-blindness-cases-us-expected-double-2050.

[2] Alexiou, Gus. "Oko App Leverages Ai to Help Blind Pedestrians Recognize Traffic Signals." *Forbes*, Forbes Magazine, 5 Oct. 2023, www.forbes.com/sites/gusalexiou/2023/08/10/oko-app-deploys-ai-to-make-crossing-the-street-safer-for-blind-pedestrians/.

[3] Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." *arXiv.Org*, 9 May 2016, arxiv.org/abs/1506.02640.

[4] Ultralytics, Glenn Jocher. "Ultralytics/Yolov5: Yolov5 in PyTorch > ONNX > CoreML > TFLite." *GitHub*, github.com/ultralytics/yolov5. Accessed 26 June 2024.

[5] Wojke, Nicolai, et al. "Simple Online and Realtime Tracking with a Deep Association Metric." *arXiv.Org*, 21 Mar. 2017, arxiv.org/abs/1703.07402.

[6] Wojke, Nicolai. "Nwojke/Deep_sort: Simple Online Realtime Tracking with a Deep Association Metric." *GitHub*, github.com/nwojke/deep_sort. Accessed 26 June 2024.

[7] Emara, Taha. "Real-Time Distance Measurement Using Single Image." *Real-Time Distance Measurement Using Single Image*, emaraic.com/blog/distance-measurement. Accessed 26 June 2024.

[8] Zhang, Edward. "Udacity Self Driving Car Dataset." *Kaggle*, 2 July 2022, www.kaggle.com/datasets/sshikamaru/udacity-self-driving-car-dataset.