Human Trust and Computer Vision in Autonomous Driving

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Zi Zeng

Spring, 2020.

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Lu Feng, Department of Computer Science & Engineering Systems and Environment

Human Trust and Computer Vision in Autonomous Driving

Zi Zeng

Abstract—To investigate the current state of autonomous system development and to gain a better understanding of the field, this report on autonomous driving systems is divided into three main parts, each of which focuses on a different aspect of these systems. The II part outlines a human trust study related to an autonomous system that aims to reveal the real meaning and significance of human trust in autonomous driving. The III and IV parts, lane and object detection, focus on the use of computer vision technologies in autonomous driving by exploring existing projects and testing hands-on implementations using images and locally recorded videos. By exploring these different approaches, it is possible to provide a deeper and more comprehensive understanding of current research in autonomous driving systems.

I. Introduction

Autonomous driving appears to be the future of transportation, and countless research studies and industry projects have been conducted in this field in the hope of delivering safer and more reliable transportation to the public in the near future. For example, all major car manufacturers plan to deliver self-driving cars commercially by 2025 [1]. The commercialization of autonomous cars will provide multiple benefits to consumers. In 2014, Americans were caught in traffic for approximately 6.9 billion hours; with autonomous cars, drivers will have more free time because they don't have to drive [2]. Additionally, the economic benefits should not be neglected. According to a NHTSA study, vehicle-related accidents were responsible for $871 billion in economic losses [3]. The use of autonomous vehicles will help resolve this problem, provided the autonomous systems function correctly. Moreover, more than 57 million people in the United States have a disability, and autonomous cars will provide these people with convenient transportation. One study reported that autonomous driving has the potential to create two million job opportunities because people with disabilities will be able to get to their workplaces more easily [4].

Despite all of these benefits, the concept of autonomous driving is still not widely accepted by the public. Studies have shown that people generally have the impression that autonomous cars are not reliable, and this presents a significant challenge for autonomous car designers and manufacturers [5]. Therefore, it is extremely important to increase human trust in autonomous cars. However, it is impossible to achieve that goal without improving the algorithms used in these cars in areas such as route selection and object detection algorithm, which involve the fields of machine learning and computer vision. For example, neural networks, supervised learning, and reinforcement learning are already contributing to the development of more advanced autonomous vehicle control systems [6].

II. Human Trust Experiment with a Driving Simulator

This section documents part of a human trust study in relation to the autonomous car system as conducted by our research group. Driving scenarios were created using the PreScan software and human subjects were recruited to engage with the simulated scenarios in a driving simulator. The change of human trust level in Likert scale and the subjects' other responses during the experiment were recorded for future analysis [7]. Trust level have been used to evaluate human trust in autonomous systems. For example, in a study on trust, the researchers recorded and analyzed the participants' trust level prior to and after driving in manual and autonomous mode, respectively [8].

To understand the importance of trust in autonomous driving better, the definition of human trust in relation to the autonomous driving system needs to be clarified. Researchers do not all define human trust in the autonomous system in the same way, and so, to avoid ambiguity, our research group defined trust as participants delegating their responsibility when driving to the autonomous system, as this expresses their willingness to accept the risks and uncertainties of autonomous driving activities [7].

All driving scenarios used in the experiment were created on a driving simulation platform called PreScan and then run on a driving simulator. Researchers can do a number of things with PreScan: for example, they can choose the level of automation that best fits their research goals, they can add multiple cameras and various sensors to be included in the automation, and they can place different obstacles on the routes. PreScan also has the capability to execute the pre-designed scenarios using its GUI-based preprocessor [9]. With its powerful functionality, PreScan is widely used in both industry and academic research for the design and study of autonomous driving. Researchers can test their driving scenarios within a virtual system, and they can measure the efficiency of their driving algorithms, such as an intersection collision avoidance algorithm, with ease [10]. The driving simulator used in this study has three screens, a steering wheel, and throttle and brake pedals. Participants use these as they would in a real vehicle. In addition, the steering

wheel has two buttons that allow participants to indicate their increased, or decreased, level of trust in the system while it is operating. Pressing both buttons simultaneously will switch them between autonomous and manual driving modes. Data on trust level change and mode switching will be recorded throughout the experiment [7]. Depending on the experimental goals and setups, a simpler desktop kit, such as the Logitech Racing Wheel with pedals, may also be used.

During the experiment, each participant was required to drive in three scenarios, all of which required them to travel from one point to another along routes that were pre-planned by the autonomous system. These PreScan scenarios were developed by Ph.D. students in the group, and each scenario included different types of incident, with the order of the incidents varying. One scenario included four incidents that followed in the order: pedestrian, empty incident, obstacle, truck. A second scenario had four incidents in the order: truck, empty incident, obstacle, pedestrian. The third scenario presented its four incidents in the order: pedestrian, truck, pedestrian, obstacle. A baseline trial was provided which acted as training so the participants could familiarize themselves with the driving simulator before the actual experiment. Each trial started in the autonomous mode setting, and all the incidents, except the empty ones, were designed to be detectable by sensors attached to the autonomous system. Drivers were given two options when they heard an alarm from the autonomous system indicating that the sensor had detected an incident a certain distance in front of them. The distance was pre-determined when the scenarios were first designed in PreScan, and it allowed a period of time for the driver to react to the situation. If they did not trust the autonomous system's ability to avoid the incident, they were free to change from autonomous mode to manual mode and therefore to take control of the steering, brake, and throttle. Alternatively, they could choose not to switch modes and to allow the autonomous system to handle the incident [7].

Throughout the experiment, the drivers adjusted their level of trust in the autonomous system by pressing the buttons on the simulator's steering wheel. After each trial, the participants were asked to fill out a survey, which asked questions such as their satisfaction rate with the performance of the autonomous system and their reasons for taking over from the autonomous system when faced with an incident [7]. The autonomous system relies on various forms of advanced sensors and built in algorithms to adjust its control of throttle, brake, and steering and also to choose routes for the driver. The participants' responses and the recorded data will provide answers to research questions such as how autonomous system designers can help new autonomous drivers to gain more trust in the system. One potential solution is to implement algorithms that choose better routes at the beginning of the autonomous experience to allow drivers to build a foundation of trust. A definition of "better" in this context could refer to roads that were straighter and had a lesser chance of encountering incidents.

This initial part of the research report focuses on the details involved in assisting simulation experiments to study trust and safety in autonomous driving. In addition, the group designed another driving trial using PreScan. This trial involved combinations of a series of driving scenarios and was intended for use by members of the group for collecting and analyzing data such as the time taken for a driver to switch from autonomous driving mode to manual mode when encountering an incident. The scenarios for this trial included incidents such as barrels, falling trees, and trucks in the road. However, unlike the human trust experiment mentioned previously, a second autonomous car was added, which maintained the same speed as that operated by the participant and travelled in front at a predetermined distance. This was done to avoid the drivers becoming aware of the incidents ahead of the alarm going off. MATLAB was used to record additional outputs such as lane types and distances between the autonomous cars. The lanes were provided through four output ports that allowed further calculations to determine whether the driver had driven off the road.

III. Lane Detection for the Autonomous Car

One of the greatest challenges for autonomous system designers is to ensure that autonomous vehicles do not drive out of their lanes. As one can imagine, autonomous car users will have high expectations that the vehicle will be able to identify lanes. An autonomous system therefore requires a lane detection algorithm that is able to recognize solid or broken lines and different colors, mostly white and yellow. The algorithm also needs to perform this task quickly and efficiently because lane detection is needed in real-time.

In computer vision, Canny Edge Detection and Hough Transformation are often used to detect lines. To examine the use of these algorithms for lane detection in real time, an autonomous vehicle project from Udacity, and coded by Galen Ballew, was studied [11, 12]. This project constructed a simple lane detection system using OpenCV and was written in Python using Jupyter notebook. OpenCV is a computer vision library for image processing that is often used for real-time computer vision tasks. Jupyter notebook allows developers to visualize their computing results on the same page as their codes. To test this project, the input video needed to be broken down into a sequence of frames. Each frame of a video is in fact an image in RGB format, which is so named for the three color channels, red, green and blue. The first step was to covert each frame to a grayscale image that had only one channel. This type of color space transformation is often seen in computer vision and is accomplished by using the functions provided in the OpenCV library. To describe it simply, each index in the greyscale channel is the average of three indices at the same location in the RGB image. Sometimes, the color information from the three channels has little impact on the final results, so the greyscale images are used to save computing time as they require only one third of the time that RGB images take [13]. This offers a great advantage in real time processing. As mentioned previously, lanes are often demarcated in white or yellow, and it is therefore essential that the algorithm can differentiate between these two colors and the other colors. Fortunately, each color has its own pixel range, which can be

used to identify it. To prepare for this, the greyscale images must first be converted into images in the HSV color model. Unlike the RGB color model, HSV contains hue, saturation, and value information, which make it far easier to describe a color within a range [14]. Before the Canny Edge Detection is introduced, Gaussian blur needs to be applied to the images. As its name suggests, Gaussian blur involves blurring an image, resulting in a modified image with reduced noise and detail [15]. Image noise normally refers to brightness and color information that does not belong to the subject being filmed but rather results from random electronic input [16]. Image noise can affect the results of edge detection greatly, so it is very important to remove it before processing [17]. After the Gaussian blur, the Canny Edge Detection algorithm can be applied to the image. Canny Edge Detection computes both the gradient and direction of the edges identified in the image, and it also combines edges that are very close to each other into a single edge by using non-maximum suppression [18]. Just as it is important to differentiate white and yellow from the other colors, it is also necessary to differentiate between the region in front of the car and the other areas in the image. This region is called the area of interest. This is done by defining a polygon shape within the image [11, 12]. Hough Line Transform is then applied to the line drawing codes to determine whether the detected edges are, in fact, lines. After the lanes have been detected and drawn into the image, the modified images, or frames, can be combined to form a new video which has the same content as the original but now has the lanes in front of the car highlighted in real-time. For demonstration purposes, three images processed by the algorithm described are presented in Figure 1, 2 and 3.

As with the other methods, this lane detection method is not perfect. For example, the region of interest is fixed, which means that it does not perform well in situations where the vehicle is turning. Researchers have been working on developing better methods for lane detection. Some people have taken advantages of feature detection by using ROI selection methods, which increase lane detection accuracy [19], while others have found a way of converting the images from a front view to a top view to get more precise lines [20]. Some advanced lane detection algorithms take camera calibration, curvature of lanes, and vehicle position all into account [21].

Fig. 1. Lane Detection Result 1 [22]

Fig. 2: Lane Detection Result 2 [23]

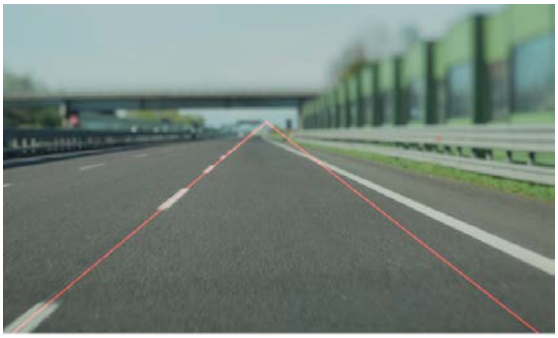

Fig. 3. Lane Detection Result 3 [24]

IV. Object Detection Through Static Image and Video

As with lane detection, object detection is a very important aspect of an autonomous system. To make a safe lane switch, an autonomous car needs to have the ability to detect vehicles nearby and also the speed of those vehicles. The car also needs to understand its environment well before it can make its operating decisions. For example, it needs to identify traffic lights and road signs.

In this section of the report, both static and real-time object detection methods are explored. Road objects such as vehicles and traffic lights are some of the most important objects to detect in autonomous driving. However, it is also important to detect other kinds of objects while driving, in complex driving circumstances in particular, such as driving in urban environments. Therefore, computer vision methods that focus on generalized object detection were selected for investigation first, and detection method for a specific object was presented next.

For object detection in static images, the approach used is to fine-tune and retrain a pre-trained ResNet, using the COCO-dataset. COCO is a large-scale object detection dataset that contains 80 object categories. In this dataset, some categories are relevant to autonomous driving, such as traffic lights, trucks, buses, motorcycles, bicycles, cars, and persons [25]. The COCO dataset has 330K images, and training the ResNet with all these images would take a long time on a personal laptop. Therefore, to increase the training speed and reduce network overheads, only 20,000 images were used for the training and 1,000 images for the validation. Those 20,000 training images and 1,000 validation images were randomly selected from the COCO dataset. Image transformations were done using the

Torchivision package, which contains a number of dataset and transformation operations that are constantly used in computer vision [26]. All the images were resized to 256 x 256 then cropped and centered. Image normalization was applied to make the computation more efficient. To train and validate a neural network, it is essential to have a dataset, which is a group of labeled images. As mentioned previously, Torchvision contains a list of datasets that include two types of COCO dataset: Captions and Detection [26]. In the approach taken, it was necessary to map each image to an eighty-dimensional vector. Each entry in the vector represented one object type. Value 1 indicated that the object-type was present, and 0 that it was not. The Torchvision COCO Detection dataset was modified for this purpose. Two datasets were created, one for training and the other for validation. To batch and iterate the datasets, two data loaders were created using a batch size of 128. A batch size of 128 means that the datasets are divided into groups where each group contains 128 images. After pre-processing the data, the next step was to build the model. Again, taking advantage of the Torchvision package, a pre-trained model named ResNet-50 was loaded. ResNet is a convolutional neural network that performs deep residual learning for image recognition [27]. To retrain this model on a new small set of data without having to do everything from scratch or worry about the negative effect of overfitting, the loaded ResNet model was finetuned by replacing the last layer with a new linear classification. The weights of the network were also modified using the SGD training. The object classifier was defined as the modified ResNet, the loss function was defined as a combination of the Sigmoid layer and the Binary Cross Entropy loss [28]. The network was then trained ten times and validated each time it was trained. Each training iteration processes all the data in a batch at a time. The model weights were updated throughout the training and validating process. The final training accuracy of the network was 82.9%, and the validation accuracy was 83.41%. Figure 4 to 9 show some test results: the top ten classes predicted are shown with their predicted score next to them.

Fig. 4. Object Detection Result 1 [25]

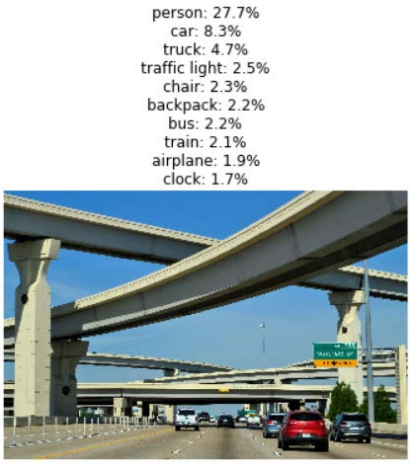

Fig. 5. Object Detection Result 2 [29]

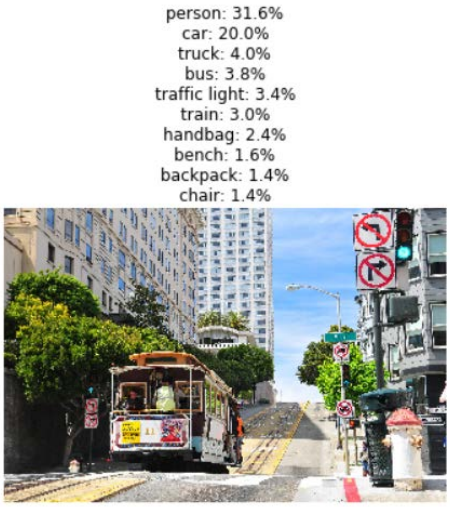

Fig. 6. Object Detection Result 3 [30]

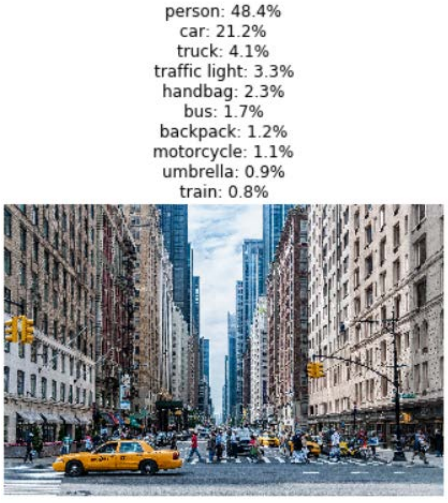

Fig. 7. Object Detection Result 4 [31]

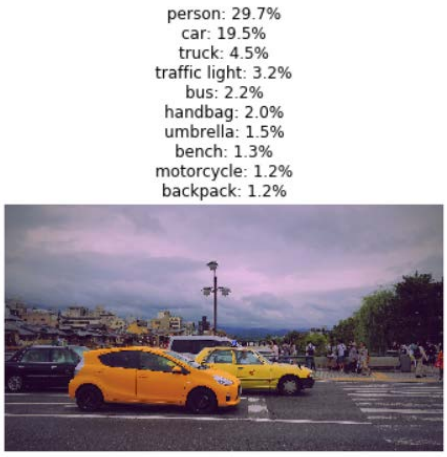

Fig. 8. Object Detection Result 5 [32]

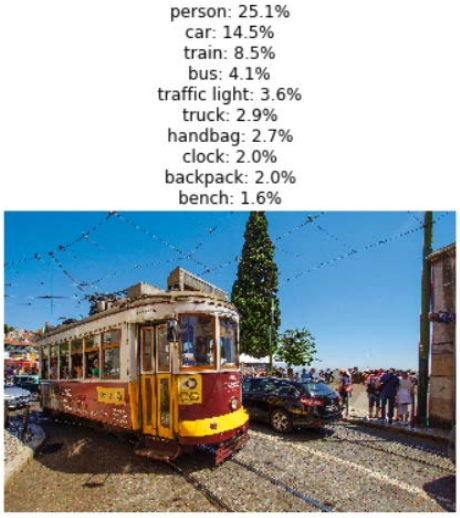

Fig. 9. Object Detection Result 6 [33]

Unlike object detection in static images, real-time object tracking is more useful in an autonomous system. To learn more about real-time object detection for the autonomous car, YOLO was explored. This is an object detection system that processes images in three main steps, which include resize image, run image through convolutional network, and threshold detection result by confidence scores, which is the accurate score of the object detected [34]. The name YOLO stands for You Only Look Once, which implies that the system is capable of predicting multiple objects simultaneously from one input image. Full images are used in training the YOLO system and, as a result, YOLO is better at catching the general features of an object than other comparable approaches [34]. Using the code from Art Poltavsky's real-time object detection project, which used YOLO, I was able to do object detection on video streams taken locally [35]. Screenshots were taken and are shown in Figure 10 to 16. From the results, it is possible to see that YOLO did a better job than the static object detection algorithm described previously because it produced less false detection, which is a significant advantage also mentioned in Redmon's paper. It was also able to predict objects accurately even when the objects had not been fully captured by the camera. Nonetheless, YOLO is not perfect, and it can still give wrong detection results in some situations.

Fig. 10. Detect Moving Vehicles (YOLOv3)

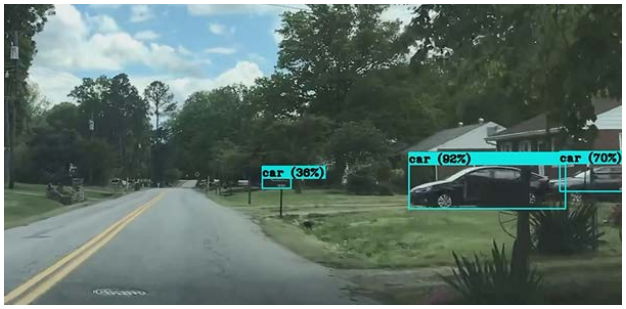

Fig. 11. Detect Parked Vehicles (YOLOv3)

Fig. 12. Stop Sign - True Positive and False Positive (YOLOv3)

Fig. 13. Bench vs. Bridge - False Positive and True Negative (YOLOv3)

Fig. 14. Train vs. House - False Positive and True Negative (YOLOv3)

Fig. 15. Truck – False Positive and True Positive (YOLOv3)

Fig. 16. Detect Objects that are Not Fully Captured (YOLOv3)

The version of YOLO from Poltavsky's project was YOLOv3, which trained on the COCO dataset [36]. As shown in the images of the results presented above, Poltavsky's YOLOv3 has the ability to detect multiple classes of objects, such as stop signs, traffic lights, and cars. While this is beneficial, sometimes a YOLO detector that focuses on a specific object may be desired because it would more accurately detect that object. Anton Muenlemann created a project on training YOLOv3 for any object that a user wanted [37]. Based on his implementation, a YOLOv3 was retrained into a customized version that only detects traffic lights in this study. First, the LISA Traffic Light Dataset was downloaded from the Kaggle website. This dataset contains frames of driving sequences in JPG format as well as annotation information [38, 39, 40]. However, these annotation data were not used, instead, customized annotation data were created manually. A hundred images were selected from the day-sequence part of the dataset. Although those images were manually chosen in a somewhat random manner, as much as possible, I tried to select images that included traffic lights from different roads (instead of choosing the same traffic lights from different angles and distances). VoTT, a tool for labeling images and creating annotations, was used to label those images. All images were labeled manually in VoTT. Four of the labeled images that were used for training are shown in Figure 17 below. Ninety images were used for training, and 10 images were used for validation. The training still took advantage of YOLOv3's pretrained dark-net weights, so no need to train the YOLOv3 from scratch; the final training loss of this modified YOLOv3 was 23.0596%, and the validation loss was 28.4340%. To test this model, I recorded local traffic video streams, and the results are shown below. Figures 18–21 each contain three pictures. The top and middle images show the results from the modified YOLOv3 that only detected traffic lights and the unmodified YOLOv3, respectively. The bottom image is the original frame before any detection process was initiated. As shown by the results, the modified YOLOv3 is more accurate than the unmodified YOLOv3, sometimes even detecting traffic lights that the unmodified YOLOv3 could not detect; however, the modified YOLOv3 did not perform as well as the unmodified YOLOv3 in detecting traffic lights that were very far away. If more images were used to train the YOLOv3 tool, more accurate results could likely have obtained.

Fig. 17. Labeled Traffic Lights [38, 39, 40]

Fig. 18. Traffic Light Detection Result 1 (Customized YOLOv3)

Fig. 19. Traffic Light Detection Result 2 (Customized YOLOv3)

Fig. 20. Traffic Light Detection Result 3 (Customized YOLOv3)

Fig. 21. Traffic Light Detection Result 4 (Customized YOLOv3)

V. Conclusion and Future Proposals

This report explored three important areas that industry and researchers have been working on. These included a human trust study on using the autonomous system, and the use of neural networks and other computer vision technologies for lane and object detection in autonomous driving. The human trust study described in detail the human trust experiment with human subjects using PreScan software to develop driving trials that combined varies driving scenarios. This study will help autonomous system researchers to develop better algorithms to potentially maximize passengers' trust in the autonomous system, and therefore, improve the likelihood of their becoming involved with autonomous driving and determine their future driving experience. The use of computer vision technologies was also explored in the hope of providing a better understanding of current research directions in the development of the autonomous system. Using the Canny Edge Detection and Hough Transformation algorithms together can provide a basic lane detection model, but the results yielded were not up to expectation because they were limited by many factors such as the shape of the roads, the surrounding environment, and the lighting conditions. Other advanced lane detection technologies were also briefly discussed, but all of them have their own limitations.

Using modified neural networks and YOLO for object detection exceeded the study's expectations, although even those results were not perfect. The neural networks constructed

by retraining RestNet-50 can only be applied to static images within the current set up, and they may give false positives when detecting objects presented in an input image. YOLO functions better in real-time detection than the neural network detection model, but it is less than perfect and it may detect the wrong type of objects. As found with the modified YOLOv3, it is beneficial to use a specific set of data to train the network in object detection, and one can anticipate an increase in accuracy given a larger training set. Moreover, the traffic light detector presented in this study is only suitable for daytime detection; if nighttime traffic light data is used in training, it will certainly improve the proposed model's ability to detect traffic lights in more scenarios.

Future studies are needed to explore other capabilities of autonomous systems. With six levels of automation, including no automation, driver assistance, partial automation, conditional automation, high automation, and full automation, more work is needed to improve current autonomous systems [2]. Technologies, such as text mining and action recognition, may be helpful because autonomous systems could benefit from the ability to retrieve useful information from road objects, such as road signs or police hand gestures. It would also be beneficial to add voice commands and other recognizable features to an autonomous system; this may help provide a customized driving experience.

VI. Acknowledgment

The author gratefully appreciates the human trust on autonomous driving research and training experience as a member of Dr. Lu Feng's research group at the University of Virginia, and would like to express special thanks to Dr. Lu Feng, Ph.D. students Shili Sheng, Erfan Pakdamanian and undergraduate students Margaret Cheng and Haoxiao Zhang for their support in this journey.

VII. References

[1] A. Takacs, I. Rudas, D. Bosl, and T. Haidegger, "Highly Automated Vehicles and Self-Driving Cars [Industry Tutorial]," IEEE Robotics & Automation Magazine, vol. 25, no. 4, pp. 106–112, 2018.

[2] "Automated Vehicles for Safety," National Highway Traffic Safety Administration , 13-Apr-2020. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[3] J. Lowy, "Traffic accidents in the U.S. cost $871 billion a year, federal study finds," 29-May-2014. [Online]. Available: https://www.pbs.org/newshour/nation/motor-vehicle-crashes-u-s-cost-871-billion-year-federal-study-finds.

[4] Ruderman, "Self-Driving Cars: The Impact on People with Disabilities," 2019. [Online]. Available: https://rudermanfoundation.org/white_papers/self-driving-cars-the-impact-on-people-with-disabilities/.

[5] S. Sadvandi and D. Halkias, "Challenges of Human Factors Engineering in the Coming Transition to Autonomous Vehicle Technologies: A Multiple Case Study. ISM Journal of International Business, 3(1), 3–8.," 2019. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=139003520&site=eds-live.

[6] S. Kuutti, S. Fallah, R. Bowden, and P. Barber, "Deep Learning for Autonomous Vehicle Control: Algorithms, State-of-the-Art, and Future Prospects," Synthesis Lectures on Advances in Automotive Technology, vol. 3, no. 4, pp. 1–80, 2019.

[7] S. Sheng, E. Pakdamanian, K. Han, B. Kim, P. Tiwari, I. Kim, and L. Feng, "A Case Study of Trust on Autonomous Driving*," 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019.

[8] C. Strauch, K. Mühl, K. Patro, C. Grabmaier, S. Reithinger, M. Baumann, and A. Huckauf, "Real autonomous driving from a passenger's perspective: Two experimental investigations using gaze behaviour and trust ratings in field and simulator," Transportation Research Part F: Traffic Psychology and Behaviour, vol. 66, pp. 15–28, 2019.

[9] TASS International, "PreScan Overview," 27-Mar-2020. [Online]. Available: https://tass.plm.automation.siemens.com/prescan-overview.

[10] M. Baek, H. Lee, H. Choi, and K. Ko, "Poster: Development of ICA algorithm for V2X communications by using PreScan," 2015 IEEE Vehicular Networking Conference (VNC), 2015.

[11] G. Ballew, "OpenCV For Lane Detection in Self Driving Cars," 17-Jul-2017. [Online]. Available: https://medium.com/@galen.ballew/opencv-lanedetection-419361364fc0.

[12] G. Ballew, "Finding Lane Lines on the Road," 2016. [Online]. Available: https://github.com/galenballew/SDC-Lane-and-Vehicle-Detection-Tracking/blob/master/Part%20I%20-%20Simple%20Lane%20Detection/P1.ipynb.

[13] K. Harikrishnan, "Image Processing tips for Computer Vision and Deep Learning tasks," Medium, 26-Apr-2017. [Online]. Available: https://medium.com/@kharikri/image-processing-tips-for-computer-vision-and-deep-learning-tasks-e5247ec94f3.

[14] K. Chinnathambi, "A Little About Color: RBG vs. HSV," kirupa.com, 26-Jul-2008. [Online]. Available: https://www.kirupa.com/design/little_about_color_hsv_rgb.htm.

[15] "Gaussian blur," Wikipedia, 16-Apr-2020. [Online]. Available: https://en.wikipedia.org/wiki/Gaussian_blur.

[16] "Image noise," Wikipedia, 28-Apr-2020. [Online]. Available: https://en.wikipedia.org/wiki/Image_noise.

[17] S. Sahir, "Canny Edge Detection Step by Step in Python - Computer Vision," Medium, 27-Jan-2019. [Online]. Available: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

[18] "Canny edge detector," Wikipedia, 14-Apr-2020. [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector.

[19] M. Li, Y. Li, and M. Jiang, "Lane Detection Based on Connection of Various Feature Extraction Methods," Advances in Multimedia, vol. 2018, pp. 1–13, 2018.

[20] B. Dorj and D. J. Lee, "A Precise Lane Detection Algorithm Based on Top View Image Transformation and Least-Square Approaches," Journal of Sensors, vol. 2016, pp. 1–13, 2016.

[21] A. Gunzi, "Advanced Lane Line Project," Medium, 21-Aug-2018. [Online]. Available: https://chatbotslife.com/advanced-lane-line-project-7635ddca1960.

[22] T. Wolter, Auto Highway Road Tunnel Speed. 2015. [Online]. Available: https://pixabay.com/photos/auto-highway-road-tunnel-speed-962083/

[23] M. Cuadros, Brown Mountains Near Road. 2020. [Online]. Available: https://www.pexels.com/photo/brown-mountains-near-road-2373495/

[24] H. Heorhiichuk, Gray Asphalt Road Under Blue Sky. 2020. [Online]. Available: https://www.pexels.com/photo/gray-asphalt-road-under-blue-sky-1021683/

[25] "Common Objects in Context," COCO. [Online]. Available: http://cocodataset.org/.

[26] "torchvision¶," torchvision - PyTorch master documentation. [Online]. Available: https://pytorch.org/docs/stable/torchvision/index.html.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[28] "torch.nn¶," torch.nn - PyTorch master documentation. [Online]. Available: https://pytorch.org/docs/stable/nn.html.

[29] ArtisticOperations, Highway Road Off Ramp Cars Travel. 2018. [Online]. Available: https://pixabay.com/photos/highway-road-off-ramp-cars-travel-3836653/

[30] Facncycravel, City Transportation San Francisco Urban Street. 2015. [Online]. Available: https://pixabay.com/photos/city-transportation-san-francisco-823604/

[31] Pexels, Architecture New York City Manhattan. 2010. [Online]. Available: https://pixabay.com/photos/architecture-new-york-city-manhattan-1853552/

[32] Free-Photos, Cars Traffic Road Transportation. 2020. [Online]. Available: https://pixabay.com/photos/cars-traffic-road-transportation-690932/

[33] Franky1st, Lisbon Tram Blue Portugal. 2019. [Online]. Available: https://pixabay.com/photos/lisbon-tram-blue-portugal-4401276/
[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[35] A. Poltavsky, “Real-time object detection using YOLO upon Google Colab in 5 minutes,” Medium, 08-Sep-2019. [Online]. Available: https://medium.com/@artinte7/real-time-object-detection-using-yolo-upon-google-colab-in-5-minutes-fd65a4903df5.
[36] J. Redmon, “YOLO: Real-Time Object Detection,” Apr-2020. [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed: 03-May-2020].
[37] A. Muehlemann, “How to train your own YOLOv3 detector from scratch,” 18-Nov-2019. [Online]. Available: https://blog.insightdatascience.com/how-to-train-your-own-yolov3-detector-from-scratch-224d10e55de2. [Accessed: 03-May-2020].
[38] M. Born and Jensen, “LISA Traffic Light Dataset,” 28-Feb-2018. [Online]. Available: https://www.kaggle.com/mbornoe/lisa-traffic-light-dataset. [Accessed: 03-May-2020].
[39] M. B. Jensen, M. P. Philipsen, T. B. Moeslund, and M. M. Trivedi, “Trivedi MM. Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives. I E E E Transactions on Intelligent Transportation Systems.,” Feb. 2016.
[40] M. P. Philipsen, M. B. Jensen, A. Mogelmose, T. B. Moeslund, and M. M. Trivedi, “Traffic Light Detection: A Learning Algorithm and Evaluations on Challenging Dataset,” 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Sep. 2015.