Damian Hupka, Zachary Zaleski, William Clampett, Nathan Pecoraro
Final Report
ECGR 4090 - Real-Time AI
https://github.com/dhupka/Real-Time-AI

Real-Time Vehicle Detection, Re-identification, and Distance Estimation for Risk Score Generation

I. Introduction

The purpose of this project seeks to accomplish a few key goals in real-time image processing. Initially, the goal of this project was to leverage varying AI models to be the driving force to create a form of risk score generation. This risk score generation was to be based upon the outputs of these real-time models and the risk score generation ultimately used to improve upon worker safety. This project aligned closely with another project that the team was completing at the time, which involved the design of a server to handle the backend communication framework between a set of devices. Furthermore, this server would be where the risk generation would be implemented, but in the current implementation, the server is simply randomly generating these risk scores. The server generated risk scores to be delivered to certain clients within the network. These clients being the: Vuzix Blade Augmented Reality Goggles, Samsung Galaxy Active Smart Watch, and a generic Android tablet to act as a digital twin device. Figure 1 below shows an overview of the end-to-end system that was implemented to aid in risk score generation and transmittal.

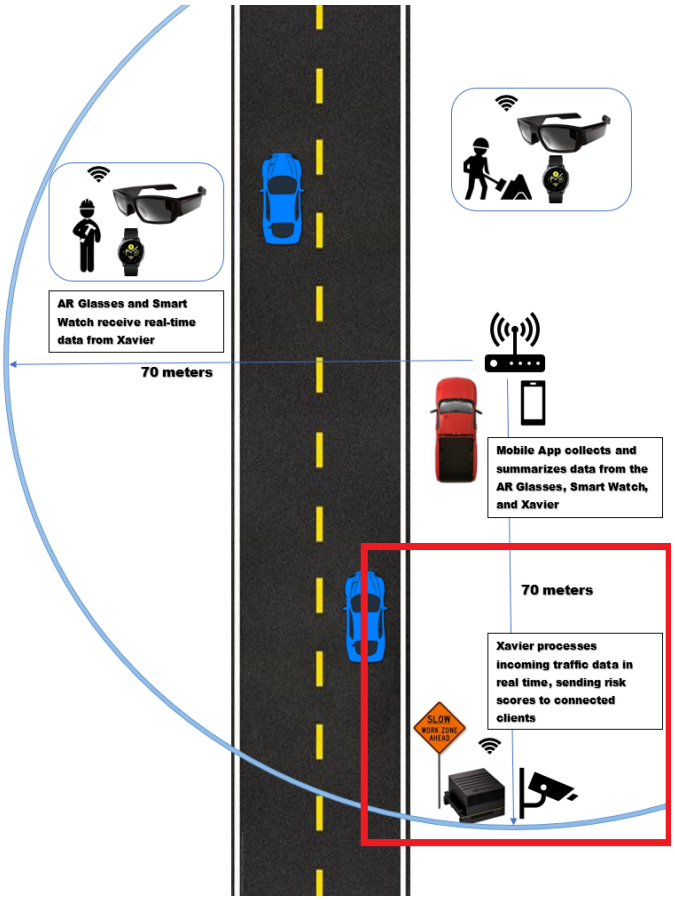

Figure 1: System Schematic

As per the red square in the figure above, for purposes of this paper, this area of real-time traffic data processing and risk score generation will be the primary focus of discussion and analysis. Thus, with the completion of the communication foundation the team took on the task of basing the risk score generation on real-time inference results from traffic data. The inference implementation was to solve three primary goals to aid in risk score generation: vehicle detection (of vehicle categories such as car, truck, bus) using a custom trained YOLO.v4 model, vehicle tracking and re-identification, and distance estimation.

II. Models

In order to implement the required functionality, the team leveraged a few existing models to conduct frame-by-frame inference on traffic data in real-time. Some aspects of the implementation required analysis to be conducted on each frame based on the outputs of the various models that were being used. Namely, two pre-trained models were used to complete the custom vehicle detection and vehicle re-identification characteristics of the system. A very high-speed vehicle detection model would need to be used as the system for risk score generation relies on brisk message transmission to ensure worker safety. Thus, a custom trained YOLO.v4 model was used to complete vehicle detection, this will be discussed in greater detail in section II.I below. Vehicle re-identification and tracking was implemented using the DeepSORT model which uses the bounding box coordinates of the vehicle detections done by YOLO.v4 to assign IDs to each object. The details of the DeepSORT implementation and model specifics will be discussed below in section II.II. Lastly, vehicle distance estimation is based on the output of the bounding box coordinates of the vehicle being tracked. Figure 2 below shows the schematic of the pipeline in which each frame will pass through to reach the final output.

Figure 2: Frame Pipeline

As in the image above, the flow of each frame that occurs within the system is as follows: the input frame of the video is extracted and passed into the custom trained YOLO.v4 model, the bounding box results of YOLO.v4 are passed into the DeepSORT model for tracking and re-identification, the output (bounding box coordinates) of DeepSORT is then used for the distance estimation calculation (discussed in section II.III), and lastly, the results of the YOLO.v4 model and the distance estimation are what is used as the driving force in risk score generation.

II.I. YOLO.v4

The team was provided a custom trained YOLO.v4 model by a graduate student who had completed a similar project previously. This model is capable of simple vehicle detection. A custom trained YOLO.v4 model was chosen opposed to the originally trained YOLO.v4 model which uses the MS COCO dataset that detects between 80 classes. However, as the specifications of the team's requirements only identifications between a subset of vehicles was necessary. Thus, the team utilized a custom YOLO.v4

model which was trained on the BDD100K Dataset. The BDD100K dataset is a “Large-scale Diverse Driving Video Database” which consists of ten classes. However, in training only eight of these classes were utilized as the “light” and “sign” classes were superfluous in this implementation, this left the remaining classes: “bus”, “person”, “bike”, “truck”, “motor”, “car”, “train”, and “rider”. The YOLO.v4 model that was trained on the BDD100K dataset that was provided had an approximate 45 percent validation accuracy. The output of the YOLO.v4 model was one of the central methods for risk score generation and in a primitive implementation the team is using the number of detections that are present in any particular frame to determine the potential risk score associated with the frame. For example, if there are ten detections within the frame the risk score will be five; whereas, with 20 detections in the frame the risk score will move to a seven. Figure 3 below shows the example output of the resulting YOLO.v4 detection

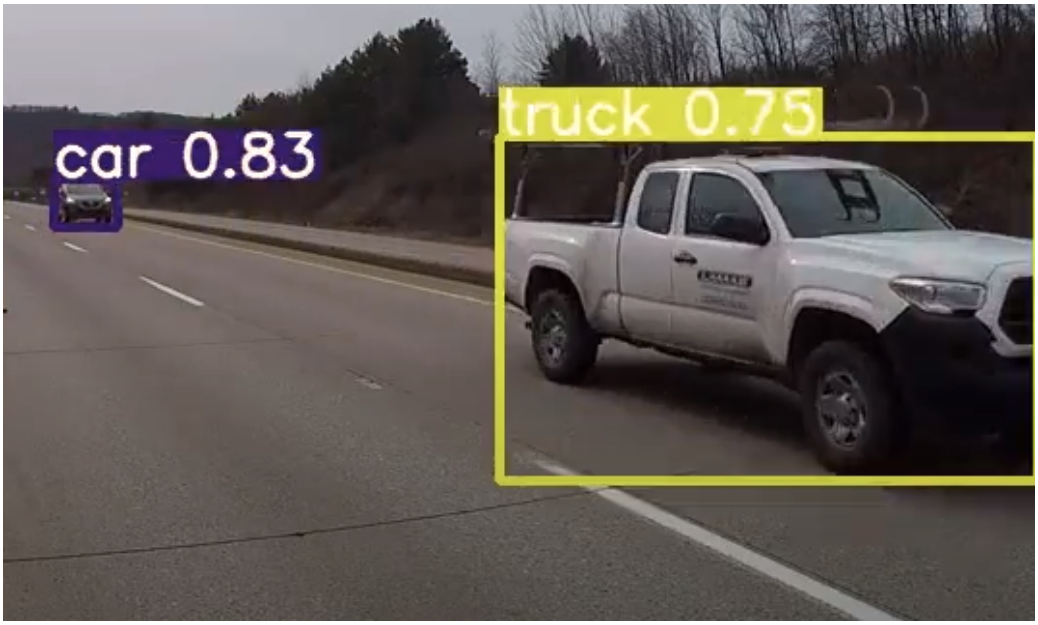

Figure 3: YOLO.v4 Detection

As in the image above, the custom trained YOLO.v4 model correctly detects between the car and the truck, furthermore, the confidence of each detection is displayed with each prediction.

II.II. DeepSORT

DeepSORT is the Simple Online and Realtime Tracking with a Deep Association Metric. Where YOLO.v4 was effective in detecting the vehicles, that alone is not enough for vehicle re-identification. DeepSORT is an improvement upon the original SORT algorithm which used rudimentary data association and state estimation techniques, while DeepSORT implements a deep association metric for object tracking. Furthermore, DeepSORT solves the problem associated with the original sort of occlusion and re-identification due to the nature of the deep association metric. The input to the DeepSORT model relies on the output of the bounding box coordinates of the custom trained YOLO.v4 model, which executes the data (ID) association with the detected object and ultimately tracks & re-identifies vehicles in

the scene. While DeepSORT provides the necessary tracking & re-identification information from the traffic data, in the system's current implementation the inference results are not impacting the risk score generation. Ideally, in the future, the results of DeepSORT would be used to be the foundation of completing trajectory analysis, which would in fact generate risk scores. But, due to time constraints of this functionality of the project this has yet to be implemented and will be discussed later in section IV. Figure 4 below shows the resulting demonstration of passing the YOLO.v4 detection into the DeepSORT model to conduct vehicle tracking and reidentification.

Figure 4: DeepSORT Inference Results

The image above shows the DeepSORT tracking and re-identification model correctly tracking the two vehicles in the frame and are being associated with an identification number.

II.III. Distance Estimation

Currently, to estimate the distance of an object from our camera, we have implemented the following equation:

$$Distance = \frac{Focal\ Length\ (mm) * Real\ Object\ Height\ (mm) * Total\ Image\ Height(pixels)}{Detected\ Object\ Height(pixels) * Sensor\ Height\ (mm)}$$

The output provides an accurate distance to an object so long as the "Focal Length" is changed to accommodate the specific camera used. The unit returned is millimeters. The diagram below demonstrates how the equation above is derived.

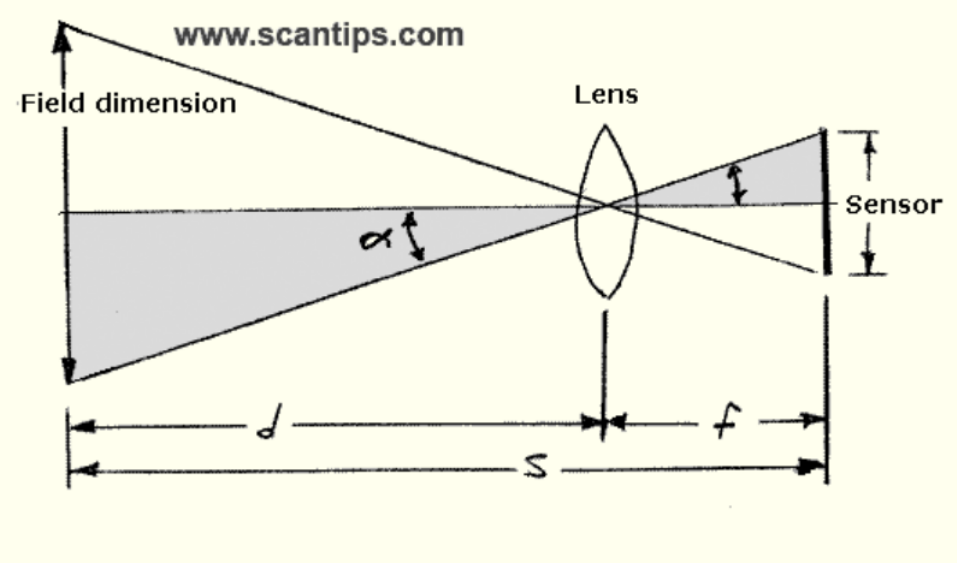

Figure 5: Real Object Height Related to Sensor Height via a Lens

As light passes through a lens, the original image is distorted in a consistent way, relative to the focal length of the lens. We can take advantage of this consistency to calculate the lens' distance from the object observed. Due to the variability of the size of the vehicles detected, we cannot always know the "Real Object Height" factor in the above equation. Since we are only interested in approximate distances, the team decided to choose a constant value for this, averaging the height of an 18 wheeler and a sedan (roughly 2 meters). We use these conditions to calculate a rough estimation of the distance of the object to the lens, or "d" in the above graphic. We use the generated estimation to modify the running risk score. Specifically, the closer an object is to the lens, the higher we make the risk score. The team utilized the previously mentioned distance calculation to show the distance estimation of the detected and tracked vehicles for a demonstration video. These results are shown below in figure 6.

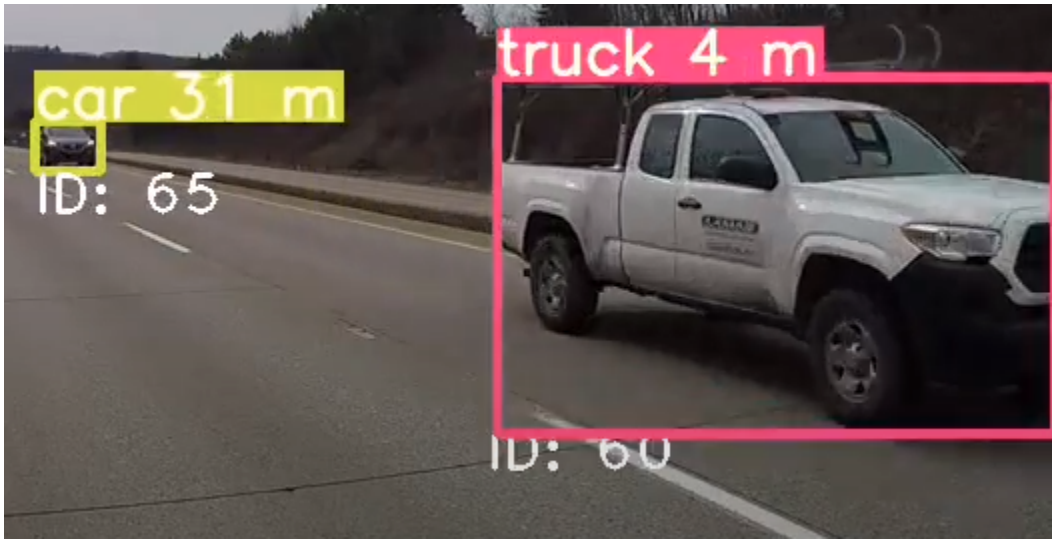

Figure 6: Distance Estimation

Lastly, using information from the YOLO.v4 detection and the calculated distance measurement that the team completed a risk score is generated and displayed to the output window. Similarly, this is the risk score that would ultimately be transmitted to the clients if running the model within the server framework that was discussed prior. The risk score generation follows a few conditional logic statements to modify this number. The risk score will be increased depending on the number of vehicles in the scene and in the case of the distance being less than a set value for any particular frame, the risk score will be similarly increased. Figure 7 below shows an example of the risk score generation in the top left of the output frame.

Figure 7: Risk Score Generation

The image above shows the system operating fully as intended, the risk score is displayed as 2.5 as there is a vehicle within 20 meters of the camera. The following section, IV, will discuss some of the potential modifications that could be made to increase the complexity of the risk score generation for a more robust implementation.

III. Results

The team was able to successfully implement the custom trained YOLO.v4 model which was trained on the BDD100K dataset to identify between eight classes. The validation accuracy of the model was approximately 45% and when testing in a real-time inference setting, this low accuracy is typically adequate as long as vehicles are in the scene long enough it is likely that they will eventually be classified as the correct category. However, in some situations, the downfalls of offering such a low validation accuracy is apparent as occasionally in testing a misidentification of a vehicle will occur and for a prolonged period, sometimes with the correct class never actually being detected. In section IV this will be discussed in greater detail; however, the accuracy of the custom YOLO.v4 model could be improved upon in a number of ways. Ultimately, it is adequate that the YOLO.v4 accuracy is rather low as generally the YOLO.v4 model offers substantially higher performance at a cost of losing accuracy, which in a real-time setting, in a system for improving worker safety, a high-speed model is paramount. Lastly, the dataset in which the YOLO.v4 model was trained on did not offer significant data samples of vehicles at a great distance thus, it is unlikely that at range the vehicle detection will operate as intended (this range is approximately 50m). **The YOLO.v4 model was running at approximately 10FPS.**

The team implemented DeepSORT as a method of completing vehicle tracking and re-identification. The goal of implementing a re-identification algorithm initially was to leverage the results of such a model to include trajectory analysis into the system which would act as another measure of generating risk scores. However, at the current time, trajectory analysis implementation was foregone due to the complex nature of the implementation (requiring comparison of the past and current frame, or other methods such as Optical Flow inferences). Furthermore, DeepSORT essentially doubles the inference time. The DeepSORT model is generally much more robust than that of other models and offers a greater footprint on the system. Also, the model cycles through all previous IDs each frame before assigning specific frames to bounding boxes. The results are passed from YOLO to DeepSORT, so if no objects are detected, DeepSORT is not utilized. Currently, DeepSORT is able to quite accurately track and label vehicles with an appropriate identification number, as well as re-identify the same vehicles when reentering the frame. However, if the accuracy of the YOLO.v4 detection model was to be improved, then the DeepSORT model would similarly experience an increase in tracking & reidentification accuracy. **The DeepSORT model that was based upon the output of the YOLO.v4 model was operating at approximately 5 FPS**. Although the FPS is nearly halved when utilizing another model, in a real-time setting this video processing speed is adequate.

The calculated distance is dependent on the focal length of the camera used. The team used an equation which leverages the Lens Equation. To accommodate the unknown constants of our equations, a rough estimation of real object height was used in order to determine the distance between the object detected and the lens. Because of this estimation, a much larger 18 wheeler will be calculated as closer to the lens than a sedan. This smaller distance will have a stronger effect on the risk score. Therefore, we are inherently biasing trucks as more dangerous. We have found that by calibrating our equation to our camera (focal length of Logitech C270), we can obtain an accurate distance estimation for most cameras. An example of this can be seen in Figure 6 above. Truck ID: 60 can be observed to be roughly 4 meters from the lens.

Utilizing the output of the YOLO.v4 model and the estimated distance of each tracked vehicle the team was able to implement a primitive level of risk score generation, and ultimately reach the goal that was attempted in beginning this project. However, the complexity of the risk score generation could absolutely be improved upon in the future, but for this implementation what is currently used is sufficient.

IV. Future Implementation

Development on this project moving forward should focus heavily on trajectory and speed analysis. The team believes that to be a much more difficult task that would require more time than is currently available. Trajectory analysis would require keeping track of the current and previous frame, and then comparing the two to draw a line. Trajectory analysis could also be implemented using the Lucas Kanade model which leverages Optical Flow methodology, but algorithms such as this will likely further impact the performance of the system. Thus, if the team were to implement trajectory analysis it would likely be

done in the former method. Implementation of such trajectory analysis would prove to be another highly beneficial tool in risk score generation. For example, if it is predicted that a vehicle may be heading in a trajectory to danger a workzone, a very high risk score will be generated. Furthermore, since the team was able to successfully implement the distance estimation, doing a similar method of previous & current frame comparison and using the distance of each detected vehicle, the speed of each vehicle in the scene could be estimated.

The accuracy of the custom trained YOLO.v4 model could be improved in a number of ways. A more substantial section of the dataset could be used in training the model and the model could be modified to be more comprehensive. For example, the model could be trained for a greater number of epochs, and a custom dataset could be developed to further distinguish between categories of vehicles to improve upon specificity of the detection, which will in turn be used to create a more accurate risk score.

Lastly, improving the complexity of the risk score generation logic would greatly improve the efficiency of the system. This complexity does not only include additional conditions for minute changes in the risk score based on vehicle distance or number of detected vehicles, but could also cater to varying scenarios. For example, the risk score can be increased proportionally for every additional truck in the frame. This can also be expanded to accommodate other vehicle combinations that may be considered more dangerous. Also, with implementation of trajectory analysis and speed estimation the risk score generation could be further improved.

V. Conclusion

In conclusion, the team was able to successfully implement a risk score generation system which leveraged varying deep-learning models, mathematical equations, and data manipulation techniques. The overall implementation enabled the team to demonstrate a system with the following capabilities: Vehicle detection (car, truck, bus, etc.), Vehicle Tracking & Re-identification, and Distance Estimation. Although, in the system's current state the vehicle re-identification is not being directly utilized in risk score generation, it is a necessary building block for future implementations which will involve the utmost importance risk score generation method, trajectory analysis. Currently, the system was designed with the goal of maximizing speed as the system will be integrated within a multithreaded server which will be transmitting the generated risk scores to a number of clients. However, including the real-time inference of approximately 5FPS into a server that is constantly sending/receiving messages the communication latency may be detrimentally impacted, but further testing of network/system configurations to ensure maximum speed is necessary.

VI. References

Seita, Daniel. "BDD100K: A Large-Scale Diverse Driving Video Database." The Berkeley Artificial Intelligence Research Blog. Accessed May 11, 2021. https://bair.berkeley.edu/blog/2018/05/30/bdd/.

Paul-Pias. “Paul-Pias/Object-Detection-and-Distance-Measurement.” GitHub. Accessed May 11, 2021. https://github.com/paul-pias/Object-Detection-and-Distance-Measurement.

Wojke, Nicolai, Alex Bewley, and Dietrich Paulus. “Simple Online and Realtime Tracking with a Deep Association Metric,” March 21, 2017. https://arxiv.org/abs/1703.07402.

M. A. Khan, P. Paul, M. Rashid, M. Hossain and M. A. R. Ahad, "An AI-Based Visual Aid With Integrated Reading Assistant for the Completely Blind," in IEEE Transactions on Human-Machine Systems. doi: 10.1109/THMS.2020.3027534

Dey, / Sandipan. “Implementing Lucas-Kanade Optical Flow Algorithm in Python.” sandipanweb, February 28, 2018.