COMP9517 Project: Vehicle Distance & Velocity Estimation, and Lane Detection

Evan Lee (z5169780)
Faculty of Computer Science and Engineering
University of New South Wales
Sydney, Australia
z5169780@ad.unsw.edu.au

Theodore Koveos (z5258231)
Faculty of Computer Science and Engineering
University of New South Wales
Sydney, Australia
theodore.koveos@student.unsw.edu.au

Rohan Maloney (z5122476)
Faculty of Computer Science and Engineering
University of New South Wales
Sydney, Australia
r.maloney@student.unsw.edu.au

Ian Ng (z5256179)
Faculty of Computer Science and Engineering
University of New South Wales
Sydney, Australia
z5256179@ad.unsw.edu.au

Andrew Timkov (z5169762)
Faculty of Computer Science and Engineering
University of New South Wales
Sydney, Australia
a.timkov@student.unsw.edu.au

I. Introduction

An autonomous vehicle or 'self-driving' car is one that is capable of sensing its surrounding environment and moving safely with little or no human input. As technology has progressed in the past decade, the introduction of advanced control systems, artificial intelligence and computer vision technologies has significantly advanced the development of these systems with semi-automated vehicles currently dominating the automobile market. [16]

The prime concern of these autonomous systems is their ability to ensure the safety of individuals as they become more prevalent on the roads. The technology implemented in these vehicles utilise hardware such as cameras, radar and lidar to capture the environment, which are then used in conjunction with complex algorithms, machine learning systems, and powerful processors to calculate and sense their surroundings. Part of this process is vehicle and lane detection; the process of intaking data from hardware to recognise and distinguish other automobiles on the road as well as the lanes of the road which the car is driving on. A complete understanding of these factors is critical for the system to protect both the individuals inside and outside of the vehicle. Specifically, vehicle detection and the subsequent motion detection is essential to calculate any motion planning being considered by the autonomous vehicle. Furthermore, the motion of the car must keep within the guidelines provided by the road, and as such lane detection has an equal importance. Any failure, fault, or oversight in this process has the potential to lead to injury and death.

The TUSimple datasets provide two challenges revolving around object detection in autonomous vehicles: velocity estimation and lane detection. The datasets include examples of various driving scenarios (specifically during the daytime) captured from a camera at the front of the car positioned facing the driver's direction. Each scenario is a set of frames extracted from a one second recording of the example. This is accompanied by a set of ground truths which provide the real-world results of the velocity estimation and lane detection. The datasets for both challenges included training and testing data which allowed for the development of a machine learning system.

As such when tasked to develop such a system to complete these challenges, our team specifically delved into machine learning, canny edge detection, hough feature extraction and geometry-based estimation. Throughout our research and development, these methods were chosen as they produced the best results when compared to the ground truths provided in the TUSimple dataset.

II. Literature Review

The advancements made in machine learning and neural networks have dominated the recent developments in artificial intelligence. Regarding autonomous vehicles, these solutions have been applied within the topics of vehicle and lane detection in order to produce more accurate results. Similarly, hardware advancements have allowed for more accurate detection, utilising sensors such as LiDAR to map 3D environments instead of working on a 2D image generated by a camera.

When researching the topic of object detection, the current method being proposed in academia is utilising neural networks. In this project, our implementation utilised a Convolutional Neural Network (CNN). This is a type of machine learning model that consists of multiple layers of windows that slide over the previous layer multiplying the corresponding elements and summing at each position of the window to compute the values of the next layer. The parameters of these windows are learned by the network through training on known data, tweaking each parameter proportional to its partial derivative with some calculated loss function (how wrong the network was). CNNs perform incredibly well on image analysis tasks as these convolutional layers learn to perform effective feature extraction on the images for the task at hand, instead of constant feature extraction functions such as SIFT [8] or SURF [9]. The Region-Based Convolutional Neural Network model (R-CNN) [7] is an accurate model that accurately tackles the task of not just classifying an image but detecting multiple objects and identifying where they are (bounding boxes).

A simple technique for estimation of distance and velocity is the geometric approach of triangle similarity [1]. This approach involves using the pinhole camera model to map a triangle of real-world dimensions (in m) in the 3D space, to a triangle of virtual dimension (in px) in the 2D image plane. The obvious flaws that have been identified with this geometric approach have been the accounting for pitch (angle) of the camera and the requirement to provide an average real-world value such as the width of a vehicle. It is worth noting that pitch is inconsistent due to the changing angles and inclines/declines of the road.

Based on our test-data, using CNN combined with geometry for distance estimation in our program produced acceptable results, however current academia differs from this methodology for autonomous vehicles. “An Overview of Lidar Imaging Systems for Autonomous Vehicles” by Royo and Ballesta-Garcia (2019) provides an in-depth overview and explaining that “Lidar imaging systems are a novel type of sensor enabling complete 3D perception of the environment, and not just the conventional 2D projection obtained from a camera.” Utilising a pulsed laser, the system is able to determine the distance of an object by recording the time it takes for an object to reflect the laser and reach the receptor again. “Since the speed of light is a given constant while we stay within the same optical medium, the distance to the object is directly proportional to the traveled time. The measured time is obviously representative of twice the distance to the object, as light travels to the target forth and back, and, therefore, must be halved to give the actual range value to the target”. As such the results of LiDAR are extremely precise when calculating distance between objects, and hence why Royo and Ballesta-Garcia state “A combination of radar, video cameras, and lidar combined with deep learning procedures is the most likely solution for a vast majority of cases. Lidar, thus, will be at the core of this revolution.” [17]

Hough transformations detect and extract straight lines from an image [18]. However, when compared to a CNN used for lane detection such as LaneNet, using Hough transforms to detect different lanes along with a neural net that to detect the edges of a lane may show weak robustness due to the complication of curved lanes [19].

III. Method

Our methodology for solving the problem at hand can be generally broken up between a subset of methods for estimating distance and velocity estimation (as they are closely related), and a subset of methods for lane detection.

A. *Vehicle Detection/Segmentation*

1) *Mask R-CNN*

To detect vehicles and produce minimal bounding boxes we used an industry leading deep learning model known as Mask R-CNN [6]. Mask R-CNN was developed by Facebook AI Research (FAIR) in 2017 as an enhancement to the series of models stemming from the original R-CNN model developed in 2014 [7].

Mask R-CNN has several stages, first a Feature Pyramid Network (FPN) [10] (essentially a series of convolutional layers where each layer is connected to the output) computes a feature map for the entire image at different scales. In the next stage, the lightweight Region Proposal Network produces a set of Regions of Interest that may contain objects, these are computing relative to ‘anchors’ which are a constant set of boxes at different scales.

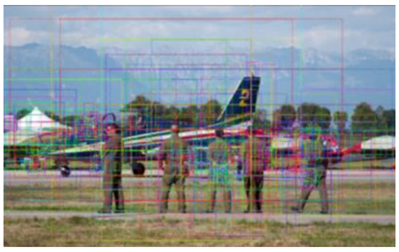


Fig 1.1: Regions of Interest [2]

These regions are then used to extract the corresponding features outputted from the FPN. The final stage then uses these features in yet another network which outputs a category, a restricted bounding box, and a binary mask simultaneously. Each stage of this huge model is trained end-to end and has incredible results.

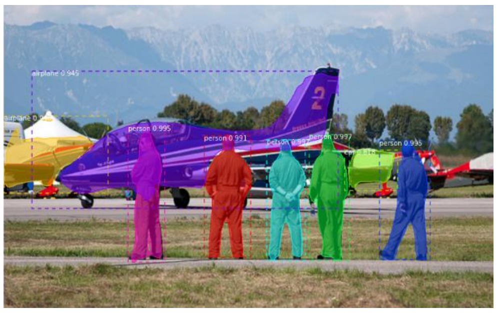


Fig 1.2: Final Output [2]

The classification branch of the final stage in Mask R-CNN is done with a fully connected neural network as this has good performance on modern GPUs and is can backpropagate into the feature extraction layers for training. It is worth noting that an SVM classifier potentially could have been used with similar accuracy, and an SVM classifier was used in the original R-CNN model.

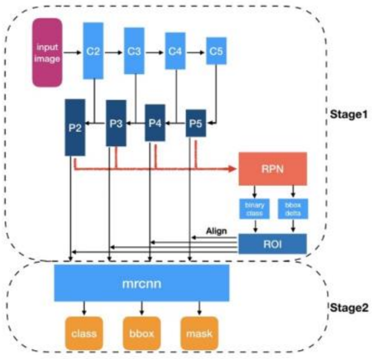


Fig 1.3: Mask R-CNN overview [11]

In our project, we found that that computing a new bounding box using the outputted mask gave better results than using the bounding box outputted by the regressor itself.

We chose to use Matterport’s Mask R-CNN implementation [2] as it was the most used Python implementation. On its own the MRCNN module wouldn’t work, mostly due to incompatibility with Tensorflow versions and usage of deprecated functions. We modified a few lines of code to get it to work on Tensorflow 1.x without any errors, although there are some warnings.

B. *Distance Estimation*

To estimate distance, we opted to used geometry-based techniques by mapping real-world points in the 3D space to points in the 2D image place with the pinhole camera model [4] as shown in Fig 2.1. This was the logical choice since we were provided with a camera intrinsic matrix for the camera used to take the photos. The camera intrinsic are the internal parameters for the camera and include the horizontal/vertical focal lengths (fx and fy) and the coordinates of the principal point (cx and cy). The principal point is the point at which a line perpendicular to the image plane can pass through. Rather than correct coordinates for the principal points, we were given modified coordinates that account for the pitch of the camera's orientation and the distance it is offset from the center of the car.

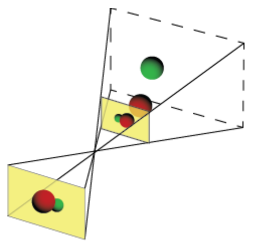
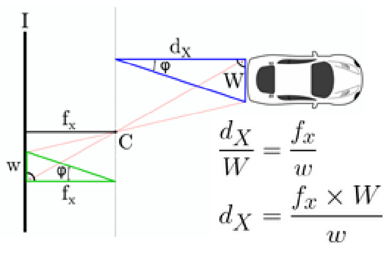


Fig 2.1: Pinhole camera model

Through triangle similarity, real world measurements can be mapped to pixel measurements.


I
d_X
φ
W
f_x
C
w
φ
f_x
$\frac{d_X}{W} = \frac{f_x}{w}$
$d_X = \frac{f_x \times W}{w}$
Fig 2.2: Longitudinal distance calculation

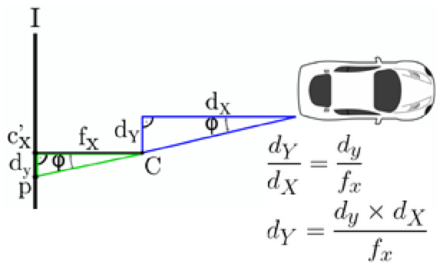

I
c'_X
d_y
p
φ
f_X
d_Y
d_X
φ
C
$\frac{d_Y}{d_X} = \frac{d_y}{f_x}$
$d_Y = \frac{d_y \times d_X}{f_x}$
Fig 2.3: Lateral distance calculation

Fig 2.2 and 2.3 (taken from [1]) show the relationship between these measurements where the blue triangles are composed of real-world dimensions (m) where W is an estimation of 1.8m for the width of a car, and the green triangles are composed of virtual dimensions (px) where w is the width of the bounding box. In Fig 2.2, **W** is the width of the car (which we estimated to be 1.8m on average), **dX** is the longitudinal distance from the camera to the car, **w** is the width of the bounding box as determined by the MRCNN (in px) and **fx** is the focal length of the camera. In Fig 2.3, **dY** is the lateral distance from the camera to the car and **dy** is the distance from the principal coordinate to the bounding box of the vehicle. We then used similar triangles to derive formulae which are used to calculate the values **dX** and **dY,** the relative displacement between the vehicle and the camera along the X and Y planes, respectively.

C. Velocity Estimation

To calculate velocity, we used the velocity formula $v(t) = \frac{\Delta d}{\Delta t}$ where d is displacement from the camera and t is time. Given that each the video clips were filmed at 20 frames per second, the time between each of the frames is 0.05 seconds. Using this information, we can evaluate velocity over several frames and get an accurate value on the assumption that distance estimation is accurate.

D. Lane Detection

The lane detection algorithm was built using only image transformation techniques such as Canny edge detection and Hough transformation. Since no machine learning was used for this component of the project, the lane detection algorithm required no training, and therefore was able to run very fast on a large dataset of images.

First, the given image was preprocessed using a grayscale and Gaussian blur filter to remove noise as shown in Fig 3.1. The preprocessed image was then run through a Canny edge detector to extract the edges of objects [5] in the image, as shown in Fig 3.2. Since the edges of trees, and cars would also be detected during this step, a trapezium shaped mask was applied to the bottom of the Canny image. This is because we assumed lanes of the road are more likely to reside in this region, given the angle of the camera from the dataset images as shown in Fig 3.3. This will also remove the number of false positive lanes detected, as cars and trees in the upper regions of the image will be cropped out.

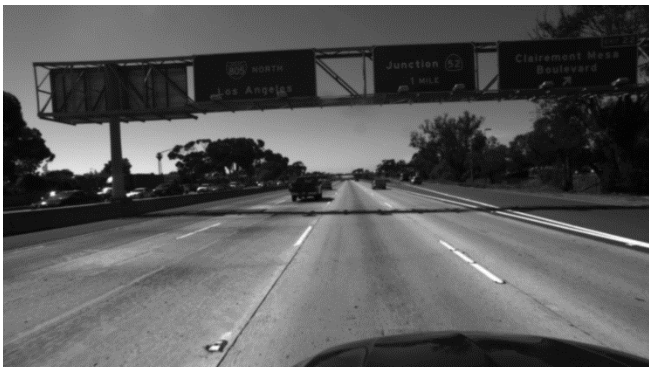


Fig 3.1: pre-processed image

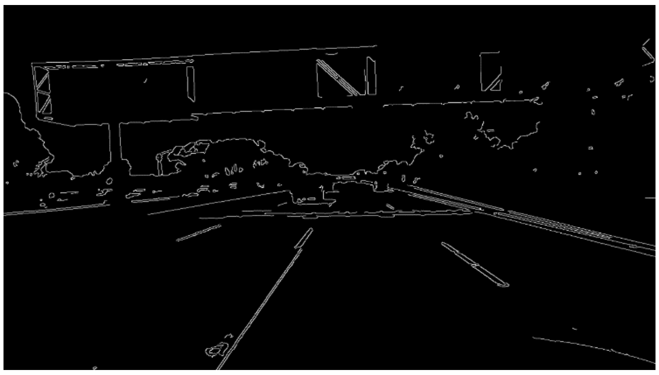


Fig 3.2: Canny image

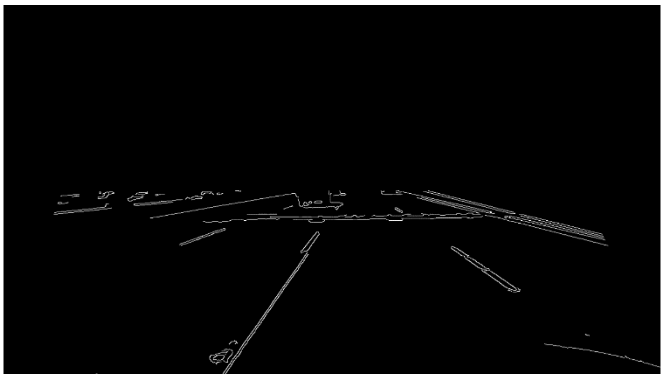


Fig 3.3: Canny image with region of interest applied.

A Hough transform was then used to identify the straight lines from the new masked image, shown in Fig 3.4. Since Hough Transforms are able to detect straight lines in an image [6], and the lane dividers have an overall rectangular shape, the Hough transformation will be able to extract the straight edges of the lane divider from the Canny image. The gradients of the lines extracted from the Hough transform are then used to extrapolate them to the horizon of the road and to the base of the image as shown in Fig 3.5.

From these, only the lines with an endpoint that sit approximately within the centre of the image are taken, and the rest are discarded, as shown in Fig 3.6. We defined an endpoint to be approximately in the "centre" of the image if it had a y-coordinate of 360 and an x-coordinate within the range of 600 – 750. Because of the angle of the camera, straight lines on the road that would otherwise be considered parallel in real life, will converge to the image's centre. Thus, by taking a subset of lines, with an endpoint located approximately in the centre of the image, true positive lanes on the road are more likely to be detected. However, even with the subset of extrapolated lines, the goal of mapping one line to a lane is still not complete. In fact, if the subset of extrapolated lines were to be drawn on top of the image, the ratio of extrapolated lines to expected lanes will be around 10:1, thus over counting the number of lanes that are in the frame as shown in Fig 3.6.

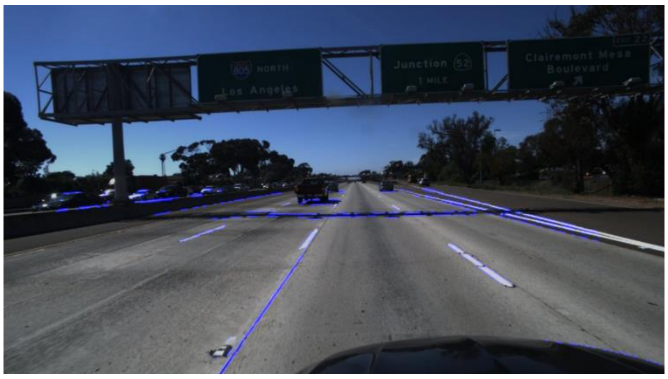


Fig 3.4: Lines generated through Hough transform of Fig 3.3.

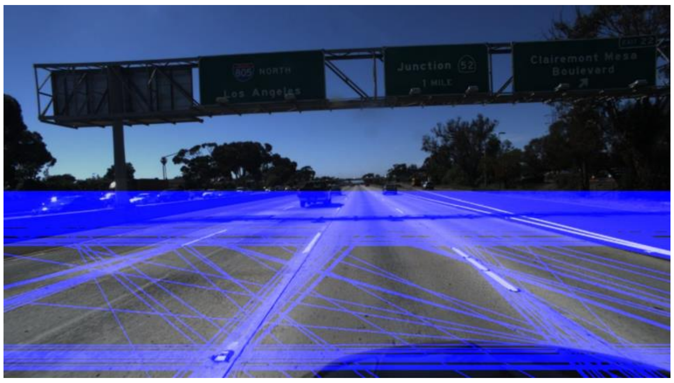


Fig: 3.5 lines from Fig 3.4 extrapolated.

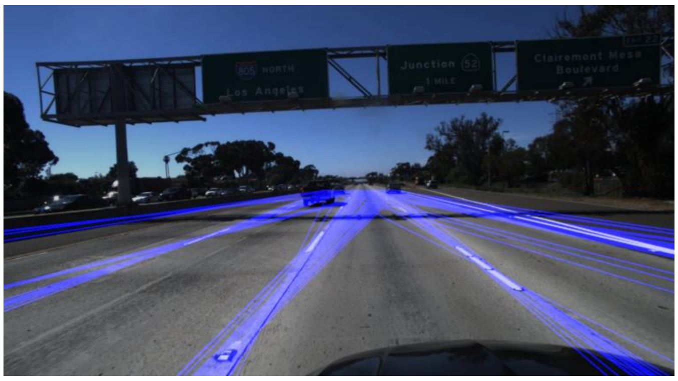


Fig 3.6: Drawing the lines from Fig 3.5 which have an endpoint located at approximately the centre of the image.

To solve this over counting issue, a sliding window is used along the base of the image and the vertical edges. For each set of lines that originate from the axis on which the sliding window is currently on, we sort the lines by y or x value (depending on the axis on which the sliding window is operating on) and take the line that is in the median position. The remaining lines in the window are then discarded. This will reduce the number of lines that overlap on top of one lane like in Fig 3.6, however due to the fixed size of the sliding window, there is still a possible case where two lines that were selected from different windows still double count a lane. However, the number of over counted lines from the previous step has been greatly decreased.

Finally, from the remaining extrapolated lines, a range of y pixel values are chosen, that would most likely contain the origin point of an outer lane. (The values chosen were $450 < y < 600$). We chose these values because any line that had a starting point lower than 450 (closer to the top of the image) were more likely to be the divider of the road rather than a lane. For example, the left most set of lines in Fig 3.6 are lines overlaying the divider of the road. For any of the lines that originated from either the left, or right side of the image, with a starting y coordinate within this range, the median positioned line was chosen to represent the outer left and right lanes.

Similarly, for any remaining lines from the sliding window step that had an origin point with a y coordinate of 720 (lines that originated from the base of the image), we took the median positioned line from the left half of the base to represent the inner left lane, and the median positioned line on the right half of the base to represent the inner right lane. This produces the final output image, with the outer-left, inner left, inner right, and outer right lanes detected and illustrated on top of the image, as shown in Fig 3.7.

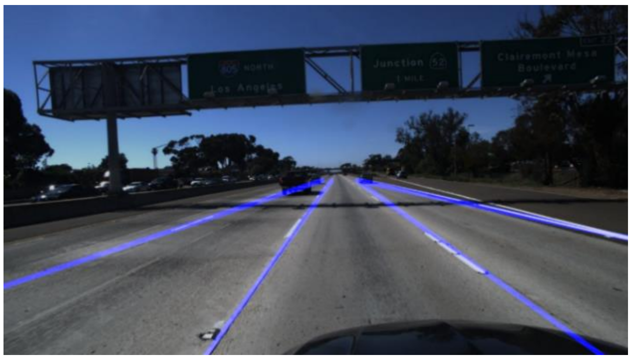


Fig: 3.7 Final image produced from lane detection algorithm.

IV. Experimental Setup

Experimenting for distance and velocity estimation was performed on Google Colab with 12.7GB of RAM and access to an Nvidia K80 GPU with 12GB of VRAM.

A. *Experimenting with Vehicle Segmentation*

Since distance estimation is reliant on the width of the bounding box around detected cars, we tested vehicle detection with an SVM classifier and a Mask R-CNN model. The method which achieved better accuracy in detection would be used for distance and velocity estimation. This is a crucial experiment because of the heavy reliance of distance and velocity estimation on bounding box accuracy.

B. *Experimenting with Velocity Estimation*

The variable we could experiment with for velocity estimation was the timespan (that is, the number of frames) for which the change in relative distance would be measured. We tested measuring velocities over 2, 5, 10 and 40 frame lengths.

C. *Experimenting with Lane Detection*

1) *Otsu's Method*

We initially considered the techniques we knew and the options we had in detecting lanes. Observing the input, we noticed that lanes tended to be white or a significantly lighter colour than the road. This observation led us to consider Otsu's method to possibly find the threshold value with which to separate lanes from irrelevant data.

As Otsu's did not produce ideal results, we reflected upon possible issues. The issues that we deemed most likely included; too much noise and extra information, insufficient or unclear line markings, and vehicles overlaying lines. We trialled a few techniques to help overcome these issues. As shown in Fig 4.1, we applied Otsu's method to an image from the velocity estimation dataset and used applied the region of interest mask. We can see that in this case, the road and lane dividers will be treated as the same after thresholding, and therefore will unable to be distinguished or found during Canny edge detection.

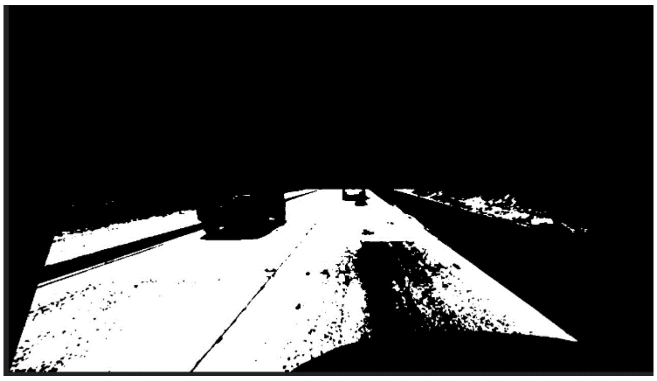


Fig 4.1: Region of Interest applied to Otsu thresholding of an image from the Velocity Estimation Dataset.

2) *Thresholding*

The first technique we tried was to define our own threshold point as we may be able to make better considerations than Otsu's. We attempted this by assessing the image with our eye, then using guess and check to approximate an appropriate value to split lanes from irrelevant data.

This produced varied results with some being very good and others not picking up much, this was due to the many inconsistencies across images which could not be controlled. The main issue that came to mind was different lane markings, with the main differences being the colour or length of lane markings.

3) *Region of Interest*

Another issue that we considered was the excess of irrelevant information for the detection of lanes. The easiest solution that came to mind was to crop the image to the region of interest so that a lot of unnecessary information could be completely excluded. To establish an appropriate region of interest, we observed a substantial number of images and came to the conclusion of a trapezium as shown below as it covered the majority of the image relevant to lane detection.

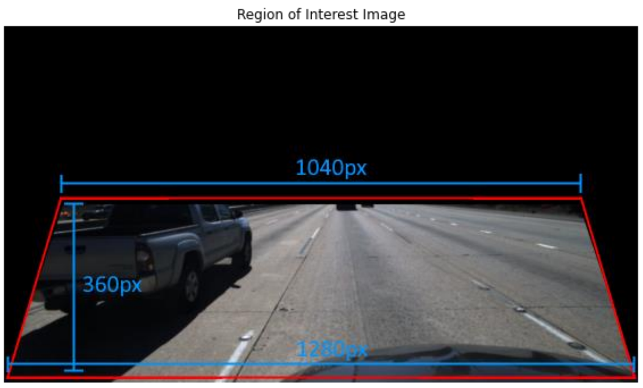


Fig 4.2: Region of interest

4) *Vehicle Masking*

By applying the region of interest to our image, we achieved much better results however, there was still one last point of extraneous information. That is, the vehicles that exist within the region of interest which include a variety of lines that would affect our later stages of lane detection.

We noted that in the first section of the project, that we had already created a model for the identification of vehicles. Using the output bounding boxes from an improved model, we applied a mask over the bounding box of each vehicle within the region of interest, further removing unrelated information from consideration.

5) *Hough Lines*

After considerable research into possible line detection techniques, the Hough line transform technique stood out as a possible solution. In combination with aforementioned techniques and the process, we used Hough lines to identify possible lines with which we extract relevant data and extrapolate it to produce desired output.

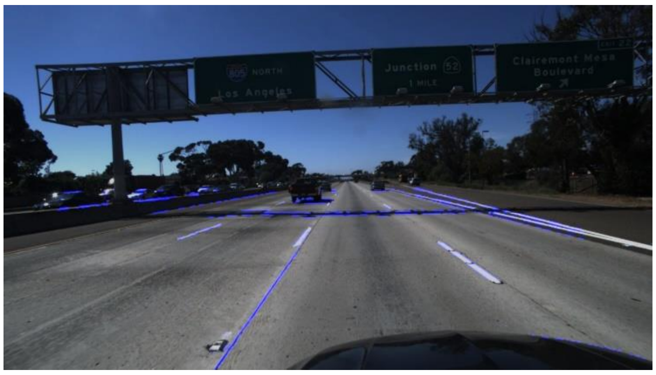


Fig 4.3: Hough lines

D. *Evaluation Methods*

We used a few different evaluation methods, as we believe that no single metric can provide a sufficient indicator on the performance of our models and algorithms.

1) *True/False Positives*

a) For Velocity/Distance Estimation

True positives and false positives provide a good indicator of how well the classifier model is able to detect vehicles. The results of these metrics influenced how we calculated other metrics for distance and velocity estimation. False positives (due to detection of non-annotated cars) detriment the distance/velocity error average (since values are measured for the wrong vehicle) so this influenced our decision to only consider the detected vehicles that were closest to ground truth bounding boxes, in other metrics.

b) For Lane Detection

While false positives and false negatives do not give an accurate representation of the extent at which lanes are detected. They give a rough indication of whether lanes are being detected. We believe that it is harder to binarily classify lane detection success from whether an existing or a non-existent lane is detected. Nonetheless, we have included results as an additional measure of evaluation, and regardless they still provide meaningful insight into the validity of the intrinsic process of our model.

2) *Jaccard Index*

Jaccard index is calculated as the area of overlap between predicted and annotated bounding boxes, divided by the area of the union [3]. This value will indicate how close our predicted bounding box was to the annotated box. This is an essential metric, because a high Jaccard index value with high distance/velocity error would indicate an issue with distance/velocity estimation rather than vehicle segmentation.

3) *Distance/Velocity Error*

We chose to measure the raw error of distance (m) and velocity (m/s), as well as error as a percentage of the ground truth. We felt both metrics of error are useful since a percentage will indicate relative performance to ground truths, but percentages will also be inflated by cases where ground truths are small values (and room for error is minimal).

4) *Accuracy*

To measure the accuracy of our lane detection, we modified the supplied demo code to suit our purposes.

In an overview, the accuracy was calculated by generating a set of threshold values from angles contrived from applying trigonometry upon the ground truth *x values* and height sample *y values*. The ground truth x values were then compared against the prediction *x values* and if their difference was less than the threshold value then they would be considered correct. Put simply, if the prediction was within reasonable proximity of the ground truth, then they would be considered correct. The number of correct predictions would be divided by the number of samples to produce our accuracy value.

V. Results & Discussion

A. *Distance Estimation*

Initial tests for distance metrics used an SVM classifier we had previously built solely for detection of vehicles. This classifier did not perform well for vehicle detection to begin with, as it missed around 33% of the annotated cars and had an extremely high rate of false positives particularly in photos of bright conditions. This poor detection performance extended itself to poor distance estimation due to the inaccuracies in bounding box fittings around vehicles.

To try and achieve better distance estimation, we built an adaptation of Matterport's Mask R-CNN Python implementation [2] to achieve better car segmentation and detection. A comparison of the metrics achieved by the SVM classifier and the MRCNN can be seen in Table 1.

TABLE I. Detection and Distance metrics for classifiers

Metric Type	**Object Classifier**	
	Mask R-CNN	***SVM Classifier***
True Positives	0.9796	0.6664
False Positives	0.5495	0.7662
Jaccard Index	0.7336	0.2784
Distance Error (m)	4.1467	18.3097
Distance Error (%)	11.6916	47.0650

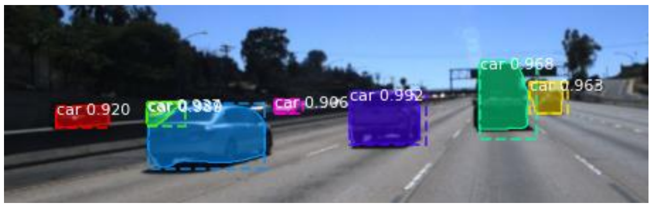


Fig 5.1: Mask R-CNN Vehicle Detection

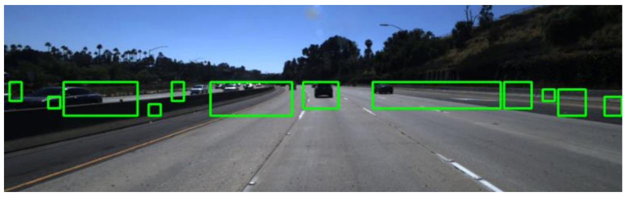


Fig 5.2: SVM Classifier Vehicle Detection

From Table 1, it is clear that the mask R-CNN performed much better at detecting the vehicles with a tighter bounding box as indicated by the true positive and Jaccard Index metrics. Given our method of estimating distance relies on the accuracy of the bounding box width, it becomes obvious that this detection accuracy by the mask R-CNN is also what resulted in the much lower distance error than that of the SVM classifier.

It is worth noting that distance calculations were only performed for true positive detections that correlated with a ground truth annotation, but not for false positives. Should the ground truth files have annotated more of the vehicles in the images, the results for mask R-CNN would have remained as good while the results for SVM classifier would have worsened. We know this from Fig 5.1 and 5.2 that show the MRCNN's false positives are of actual cars that weren't annotated whereas the SVM classifier's false positives were completely incorrect with ill-fitting bounding boxes.

B. *Velocity Estimation*

For velocity estimation we used only the Mask R-CNN because it had considerably better results for distance estimation, and our velocity calculation is heavily dependent on the accuracy of distance measurements.

Since our calculation of velocity was change in distance divided by change in time, the only factor we could test was over what time span we would take velocity measurements. We tested timespans of 2, 5, 10 and 40 frame periods, with the results presented in Table II. For context on calculation, using a time span of 10 frames for example would mean measuring the velocity over frames 1-10, 11-20, 21-30, and 31-40 then averaging these values.

TABLE II. VELOCITY METRICS FOR VARIOUS TIME MEASUREMENTS

Timespan (# of frames)	**Metrics**	
	Velocity Error (m/s)	***Velocity Error (%)***
2	6.1231	732.8696
5	1.8229	211.2546
10	2.5398	345.2911
40	4.3928	519.4693

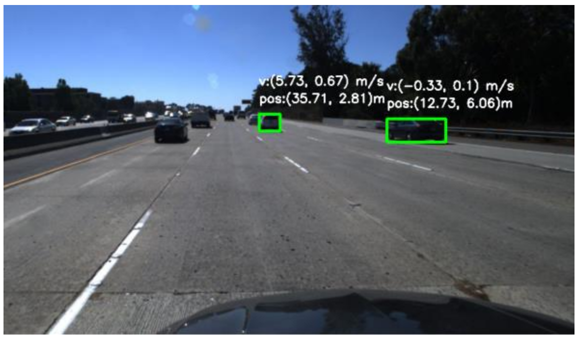


Fig 5.3: Detection of wider car from side angle

Overall, the results were not as good as we expected with the best average velocity error being just under 210% when measurements were taken over a given frame period. We also tried taking the median over the calculated values rather than the mean but for all timespans this resulted in errors over 1000% even after the removal of outliers.

In quite a lot of cases we did get good velocity estimations as low as 5-10%, but over an average of the whole dataset these results were outweighed by instances where velocity error was over 200%. One common cause of this high error is depicted in Fig 5.3, where the leftmost car is being viewed from a side angle, resulting in a wider bounding box. Since our distance calculation is heavily dependent on bounding box width, this wide bounding box caused a distance estimation error of 26%. In turn, since velocity estimation is dependent on distance, this resulted in a velocity error of 211%.

Additionally, in some cases where the ground truth velocity was a low value, like 0.1m/s in the x and y axes, the velocity error magnified. While a velocity error of 1m/s does not seem too high, in cases like these, that correlates to a 1000% error which is detrimental to our average. Had absolute velocity of the vehicles been calculated rather than relative velocity, the error as a percentage would be smaller. For example, a 1.8m/s error for a 16.6m/s velocity (approximately 60km/h) would be around 10%. For this reason, we believe that while our velocity estimation isn't perfect, it performs decently, and that absolute error is a better measurement of accuracy in this case.

With regards to time performance, distance estimation took 500ms and velocity estimation took around 30 seconds when using the Google Colab setup.

C. *Lane Detection*

We want to recognise that we conducted the creation of our model upon the velocity estimation dataset, with which we achieved desirable results. However, upon completion and beginning evaluation, we needed to use the lane detection dataset as they contained the ground truths necessary for evaluation.

TABLE III. EVALUATION METRICS FOR LANE DETECTION

Metric Type	**Metric Results (%)**
Accuracy	9.412
False Positives	58.64
False Negatives	99.83

Regarding the results that we achieved. With the accuracy method that we used; we obtained an accuracy of 9.412% which would correspond to our predictions coinciding with the ground truths 9.412% of the time. As for false positives and negatives; we detected non-existent lanes 58.64% of the time and did not detect existing lanes 99.83% of the time.

Despite the results and numbers obtained using the lane detection dataset (LDD), our lane detection algorithm worked extremely well when applied to images from the velocity estimation dataset (VED). Comparing the images from the two datasets side by side, the images from the VED are favourable compared to the images from the LDD. As shown in Fig 6.2, the lane dividers in the images from the VED are clear to the naked eye and can be easily distinguished after pre-processing by the Canny edge detector. Whereas in Fig 6.1, images from the LDD have almost no presence of lane dividers and lanes are only able to be detected via cracks in the road. As shown in Fig 6.3, this results in a large loss of information after image pre-processing and Canny edge detection. With these comparisons, it’s clear to see the first weakness of our algorithm, which is the reliance on lane dividers in the region of interest having a high enough contrast against the road to be detected by the Canny edge detector. Without this, the Hough transform won’t be able to generate the Hough lines, and as a result, the algorithm will not detect lanes in the image. An improvement to this, could possibly be to implement image to better detect faintly coloured lane dividers or cracks in the road, after pre-processing, so that the Canny edge detector can recognise them.

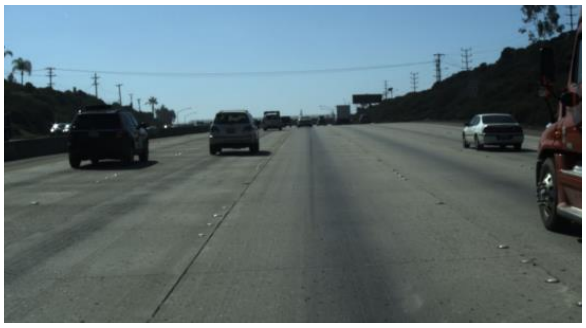


Fig 6.1 image from Lane detection dataset

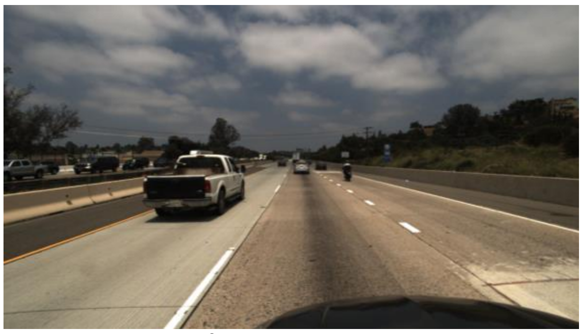


Fig 6.2 image from Velocity estimation dataset

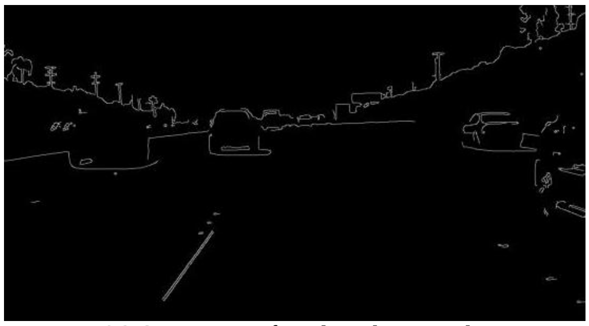


Fig 6.3 Canny image from lane detection dataset

Another weakness to our lane detection algorithm is that it cannot detect curved lanes. This is because the Hough transformation of the Canny image, generates straight lines which are then extrapolated for the algorithm to then pick as lanes. In this process, curved lanes are disregarded and extrapolated over. In addition, the process in our methodology where we choose to keep the extrapolated lines that converge to the centre of the images will only work well for straight lanes and therefore won’t consider curved lanes. However, an *idea* on how to detect curved lanes, is to take the *curve* of best fit between all start and endpoints for a particular horizontal sliding window using the start and end points of the lines generated by the Hough transformation. Following on from our algorithm’s inability to detect curved lanes, our algorithm will not be able to detect roads with more than three lanes (four lane lines in total). This is a limitation due our methodology of selecting a single Hough line from the side edges of the image, and one line from the left and right sides of the image’s base edge.

An improvement that can be made to the lane detection algorithm’s accuracy to reduce false positive lane detections, is to apply masks to regions of the image which contain cars. This improvement should not be very hard to achieve as the car detection model already exists from earlier stages of the project. Furthermore, we acknowledge that a machine learning approach to this problem may have been a better solution rather than pure image transformations. This is something that we would like to explore in more detail in an extension of the project.

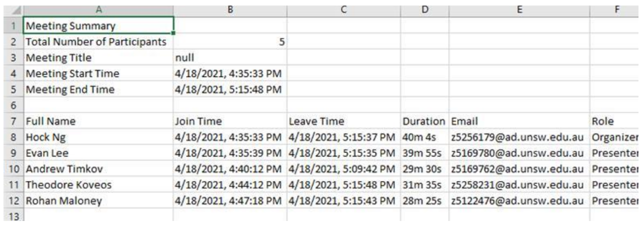
VI. Conclusion

This paper is a strong example that showcases the great number of methods that exist for the tasks of image classification and segmentation, from traditional feature computer vision techniques to modern deep learning models. This project also demonstrates that traditional computer vision methods still have a place in image segmentation, such as in lane detection.

There is still much room for investigation and improvement in the methods utilised in these tasks. To conduct vehicle detection and segmentation one could try using You Only Look Once [12], or Single Shot Detection [13] to perform object detection and segmentation much faster. To improve velocity estimation, an object tracking model such as Track-RCNN [14] or SORT [15] to trace a vehicle back through frames to ensure velocity calculation is accurate. With regard to the task of lane detection, deep learning techniques, as well as image segmentation and better feature extraction of lanes could be investigated.

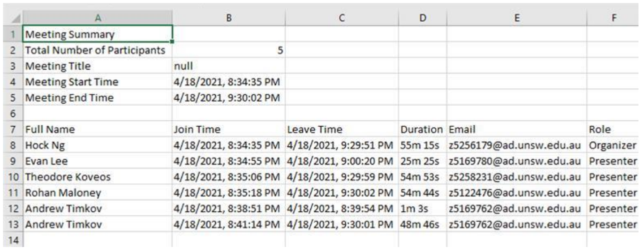
VII. Contribution of Group Members

For handling group communications and work tracking, we opted to not use Microsoft Teams with the exception of some video calls for presentation planning/rehearsing As shown in Fig 6.4-6.8. A majority of communication was done on Facebook Messenger since it was a service which everyone was familiar with and had easy access too. For managing our codebase, we used a private git repository hosted on GitHub.



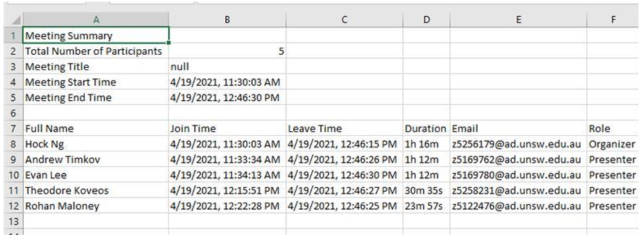
	A	B	C	D	E	F
1	Meeting Summary					
2	Total Number of Participants	5				
3	Meeting Title	null				
4	Meeting Start Time	4/18/2021, 4:35:33 PM				
5	Meeting End Time	4/18/2021, 5:15:48 PM				
6						
7	Full Name	Join Time	Leave Time	Duration	Email	Role
8	Hock Ng	4/18/2021, 4:35:33 PM	4/18/2021, 5:15:37 PM	40m 4s	z5256179@ad.unsw.edu.au	Organizer
9	Evan Lee	4/18/2021, 4:35:39 PM	4/18/2021, 5:15:35 PM	39m 55s	z5169780@ad.unsw.edu.au	Presenter
10	Andrew Timkov	4/18/2021, 4:40:12 PM	4/18/2021, 5:09:42 PM	29m 30s	z5169762@ad.unsw.edu.au	Presenter
11	Theodore Koveos	4/18/2021, 4:44:12 PM	4/18/2021, 5:15:48 PM	31m 35s	z5258231@ad.unsw.edu.au	Presenter
12	Rohan Maloney	4/18/2021, 4:47:18 PM	4/18/2021, 5:15:43 PM	28m 25s	z5122476@ad.unsw.edu.au	Presenter
13						

Fig 6.4 Meeting Summary of MS Teams meeting



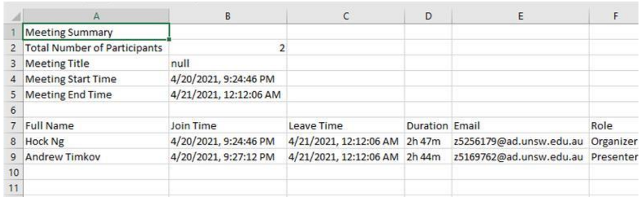
	A	B	C	D	E	F
1	Meeting Summary					
2	Total Number of Participants	5				
3	Meeting Title	null				
4	Meeting Start Time	4/18/2021, 8:34:35 PM				
5	Meeting End Time	4/18/2021, 9:30:02 PM				
6						
7	Full Name	Join Time	Leave Time	Duration	Email	Role
8	Hock Ng	4/18/2021, 8:34:35 PM	4/18/2021, 9:29:51 PM	55m 15s	z5256179@ad.unsw.edu.au	Organizer
9	Evan Lee	4/18/2021, 8:34:55 PM	4/18/2021, 9:00:20 PM	25m 25s	z5169780@ad.unsw.edu.au	Presenter
10	Theodore Koveos	4/18/2021, 8:35:06 PM	4/18/2021, 9:29:59 PM	54m 53s	z5258231@ad.unsw.edu.au	Presenter
11	Rohan Maloney	4/18/2021, 8:35:18 PM	4/18/2021, 9:30:02 PM	54m 44s	z5122476@ad.unsw.edu.au	Presenter
12	Andrew Timkov	4/18/2021, 8:38:51 PM	4/18/2021, 8:39:54 PM	1m 3s	z5169762@ad.unsw.edu.au	Presenter
13	Andrew Timkov	4/18/2021, 8:41:14 PM	4/18/2021, 9:30:01 PM	48m 46s	z5169762@ad.unsw.edu.au	Presenter
14						

Fig 6.5 Meeting Summary of MS Teams meeting



	A	B	C	D	E	F
1	Meeting Summary					
2	Total Number of Participants	5				
3	Meeting Title	null				
4	Meeting Start Time	4/19/2021, 11:30:03 AM				
5	Meeting End Time	4/19/2021, 12:46:30 PM				
6						
7	Full Name	Join Time	Leave Time	Duration	Email	Role
8	Hock Ng	4/19/2021, 11:30:03 AM	4/19/2021, 12:46:15 PM	1h 16m	z5256179@ad.unsw.edu.au	Organizer
9	Andrew Timkov	4/19/2021, 11:33:34 AM	4/19/2021, 12:46:26 PM	1h 12m	z5169762@ad.unsw.edu.au	Presenter
10	Evan Lee	4/19/2021, 11:34:13 AM	4/19/2021, 12:46:30 PM	1h 12m	z5169780@ad.unsw.edu.au	Presenter
11	Theodore Koveos	4/19/2021, 12:15:51 PM	4/19/2021, 12:46:27 PM	30m 35s	z5258231@ad.unsw.edu.au	Presenter
12	Rohan Maloney	4/19/2021, 12:22:28 PM	4/19/2021, 12:46:25 PM	23m 57s	z5122476@ad.unsw.edu.au	Presenter
13						

Fig 6.6 Meeting Summary of MS Teams meeting



	A	B	C	D	E	F
1	Meeting Summary					
2	Total Number of Participants	2				
3	Meeting Title	null				
4	Meeting Start Time	4/20/2021, 9:24:46 PM				
5	Meeting End Time	4/21/2021, 12:12:06 AM				
6						
7	Full Name	Join Time	Leave Time	Duration	Email	Role
8	Hock Ng	4/20/2021, 9:24:46 PM	4/21/2021, 12:12:06 AM	2h 47m	z5256179@ad.unsw.edu.au	Organizer
9	Andrew Timkov	4/20/2021, 9:27:12 PM	4/21/2021, 12:12:06 AM	2h 44m	z5169762@ad.unsw.edu.au	Presenter
10						
11						

Fig 6.7 Meeting Summary of MS Teams meeting

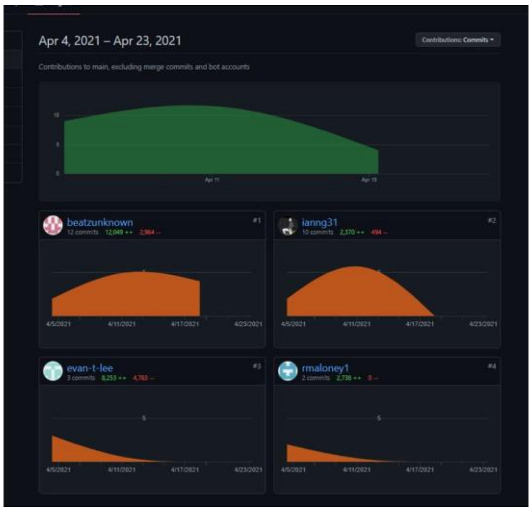
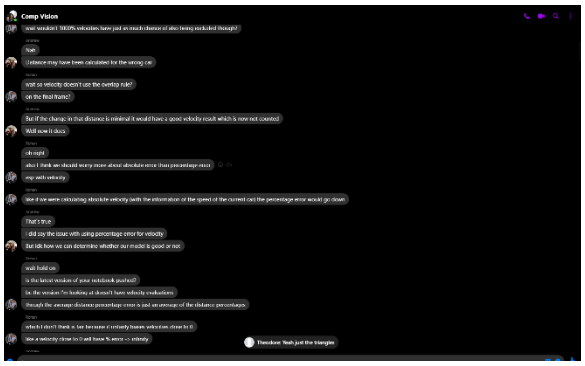

Apr 4, 2021 – Apr 23, 2021
Contributions: Commits
Contributions to main, excluding merge commits and bot accounts
10
5
0
Apr 11
Apr 18
beatzunknown
12 commits 12,049 ++ 2,964 --
#1
ianng31
10 commits 2,370 ++ 494 --
#2
evan-t-lee
3 commits 8,253 ++ 4,783 --
#3
rmaloney1
2 commits 2,738 ++ 0 --
#4
5
4/5/2021
4/11/2021
4/17/2021
4/23/2021

Fig 6.9 GitHub commit contributions



Comp Vision

wait wouldn't 1000% velocities have just as much chance of also being excluded though?

Andrew

Nah

Distance may have been calculated for the wrong car

Rohan

wait so velocity doesn't use the overlap rule?

on the final frame?

Andrew

But if the change in that distance is minimal it would have a good velocity result which is now not counted

Well now it does

Rohan

oh right

also I think we should worry more about absolute error than percentage error

esp with velocity

Rohan

like if we were calculating absolute velocity (with the information of the speed of the current car) the percentage error would go down

Andrew

That's true

I did say the issue with using percentage error for velocity

But idk how we can determine whether our model is good or not

Rohan

wait hold on

is the latest version of your notebook pushed?

bc the version I'm looking at doesn't have velocity evaluations

though the average distance percentage error is just an average of the distance percentages

Rohan

which I don't think is fair because it unfairly biases velocities close to 0

like a velocity close to 0 will have % error -> infinity

Theodore: Yeah just the triangles

Fig 7.0 Discussions via Facebook Messenger

A. *Individual Contributions*

1) *Evan Lee*

Researched and reported upon experimentation on possible implementations of lane detection. Created and managed collaboration on the git repository. Worked closely with Ian on methodology as well as working and reporting on lane detection evaluation techniques.

2) *Theodore Koveos*

Tested and improved upon LinearSVM and CNN vehicle detection methods prior to moving over to Mask-CNN. Tested Mask-CNN distance detection and velocity estimation results. Research on project topics for report and presentation.

3) *Rohan Maloney*

Implemented basic CNN model for vehicle detection to improve on original SVM model before Andrew adapted the Mask R-CNN model. Researched the Mask R-CNN model, the history of related models, how it works and why it is so accurate and wrote about such in the presentation and report. Spent several days trying (but failed) to adapt Task R-CNN and then SORT to track vehicles over multiple frames.

4) *Ian Ng*

Implemented and reported on, the methodology of the lane detection algorithm. Worked closely with Evan during the investigation and experimental stages. Facilitated team meet ups and organised meeting times.

5) *Andrew Timkov*

Researched and worked on implementation of Mask R-CNN based detection, distance estimation and velocity estimation. Experimented with SVM and Mask R-CNN classifiers. Facilitated planning and structure of the final report.

VIII. References

[1] M. Kampelmuhler, "Estimating the velocity of vehicles relative to a moving camera", Graz, Styria, 2018

[2] "Mask R-CNN for Object Detection and Segmentation". https://github.com/matterport/Mask_RCNN

[3] "What is the Jaccard Index?". https://deepai.org/machine-learning-glossary-and-terms/jaccard-index

[4] “Dissecting the Camera Matrix, Part 3: The Intrinsic Matrix”. https://ksimek.github.io/2013/08/13/intrinsic/

[5] “Canny Edge Detection Step by Step in Python – Computer Vision”. https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

[6] “Mask R-CNN”, He, Gkioxari, Dollár, Girshick, 2017

[7] “Rich feature hierarchies for accurate object detection and semantic segmentation”, Girschick, Jonahue, Darrell, Malik, 2014

[8] “Scale Invaraiant Feature Transform”, Lowe, 2004

[9] “SURF: Speeded up Robust Features”, Bay, 2006

[10] “Feature Pyramid Networks for Object Detection”, Lin, Dollár, Girschick, He, Hariharan, Belongie, 2016

[11] “Simple Understanding of Mask RCNN”, Zhang, 2018

[12] “You Onlu Look Once: Unified, Real-Time Object Detection”, Redmon, Divvala, Girshick, Farhadi, 2015

[13] “SSD: Single Shot MultiBox Detector”, Liu et al, 2015

[14] “MOTS: Multi-Object Tracking and Segmentation”, Voigtlaender et al, 2019

[15] “Simple Online and Realtime Tracking”, Bewley, Ge, Ot, Ramos, Ben Upcroft, 2016

[16] “Global Autonomous/Driverless Car Market Projections, 2020–2025: World Market Anticipating a CAGR of ~18%.” Markets, R. A. (2020, March 18) https://www.globenewswire.com/news-release/2020/03/18/2002529/0/en/Global-Autonomous-Driverless-Car-Market-Projections-2020-2025-World-Market-Anticipating-a-CAGR-of-18.html

[17] Royo, Santiago & Ballesta-Garcia, Maria. (2019). An Overview of Lidar Imaging Systems for Autonomous Vehicles. Applied Sciences. 9. 4093. 10.3390/app9194093.

[18] Nixon, Aguado. (2020). Feature Extraction and Image Processing for Computer Vision

[19] Wang, Ren, Qiu. (2018) LaneNet: Real-Time Lane Detection Networks for Autonomous Driving.