Vehicle Object Tracking

Kunal Nandanwar
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
kgnandanwar@wpi.edu

Sameer Malik
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
smalik@wpi.edu

Shivam Sharma
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
ssharma4@wpi.edu

Sushmitha Belede
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
sbelede@wpi.edu

***Abstract*—In this work we study the object detection and tracking problem for autonomous vehicle navigation by tracking cars. Our objective is to track cars from a video sequence by drawing a bounding box over these objects. We plan to use the Nvidia images dataset [5] to do this task. We seek experiment with methods that do the task at hand using geometric computer vision based feature and object detector, such as Histogram of Oriented Gradients [4] and learning based convolutional neural networks for object detection such as YOLOV3 [13] and YOLOV5 [15]. We observed that the performance of YOLOV5 was the best with the best Mean Average Precision among the three models accounting for occlusions and multiple detections. We study the fusion of the detectors with the trackers using association methods based on intersection over union and Kalman Filters and trackers based on convolutional neural networks such as SORT [2] and DeepSORT [12]. We also study the performance of different combinations of detectors and trackers.**

***Index Terms*—Histogram of Oriented Gradients, Convolutional Neural Networks, Joint detection and tracking, Kalman Filters, Tracking by Detection, Online Real-time Tracking**

I. Introduction

Significant advances have been made in 2D perception tasks applied to autonomous vehicles, such as object detection, classification, and image segmentation, tracking [6]. However, as autonomous vehicles are being manufactured one of the most significant to autonomous vehicle navigation tasks is object detection and tracking. Our work is focused on experimenting and trying out classical computer vision and deep learning based methods to carry out the task at hand.

II. Related Work

Existing methods for object detection and tracking:

- Pure geometric computer vision detectors: Classical computer algorithms such as SIFT keypoint extraction [7], Harris interest point detector [10] and Fast Retina Keypoint feature descriptor [1] etc. have been in the literature which extract features form the object of interest and group them together to localize the object invarient to scale and rotations. We have used the descriptors such as above to extract features from our objects.
- Pure CNN detectors: Pure CNN detectors such as YOLO [9] are object detectors designed to create features from input images and then to feed these features through a prediction system to draw boxes around objects and predict their classes. Detection based on deep learning comprises of two components: (1) a regression step that encompasses estimating the location and orientation of 2D bounding boxes representative of physical objects; and (2) a classification step that identifies the object inside the bounding box.
- Tracker associations using Kalman Filters:
  Kalman filtering is an algorithm that allows us to estimate the states of a system given the observations or measurements. The basic idea of the Kalman filter is by using the prior knowledge of the state, the filter makes a forward projection state or predicts the next state [8]. This prior knowledge would be in our model would be the detected objects of interest and then filters can help predict future state of these objects even under occlusion conditions.
- Tracking using detection: Tracking based on detection using a neural network has to associate the three tasks which involves, detection, segmentation and tracking together [11]. Usually from detectors, many correct detections might get discarded considering their confidence scores. Hence, we improve the trackers performance by considering detections from multiple temporal frames and associating the object's index with the bounding box detected through time.
- Joint detection and tracking methods:
  It is seen from the previous sections that object tracking was based on object detection methods to initialize and update new tracks and existing tracks respectively. Although, these two models were employed independently. Join detection and tracking methods make use of the fact that detectors and trackers are strongly interconnected. The trackers in these models can utilize the features extracted by detectors and detectors can utilize the past information from trackers. Multiple neural networks which conduct end-to-end detection and tracking have been proposed in literature. [14] As a part of our project we plan to explore a few popular trackers such as Single Shot MultiBox Detectors and Detection Embeddings for Tracking as a part of our project. [3] They have Recurrent Neural Networks with Long short-term memory based architectures along with Convolutional Neural Nets which carry out joint object detection and tracking.

III. Proposed methods

We plan to implement two different types of methods based on tracking by detection and joint detection and tracking.

- Tracking-by-detection: We first plan to implement an object detector for classifying and localizing objects in an image followed by methods which associate the detections into tracks. For detectors, we want to compare pure CNN based detector which uses the YOLOV3 and YOLOV5 architectures. For a geometric computer vision approach we use Histogram of Gradients. For tracker associations, we want to implement Kalman Filters or IoU based methods and compare their performance with the existing deep learning methods. We used YOLOv5 because of its high performance in training and detection. YOLOv5 training procedure involves data augmentation and loss calculation, which improves the performance of training. In order to make box predictions the You Only Look Once: Unified, Real-Time Object Detection [9] network predicts bounding boxes as deviations from a list of anchor box dimensions.

- Joint detection and tracking: As opposed to the detection based tracker, where tracker and the detector are independent from each other, there are approaches with an embedded tracker which do joint detection and tracking. These methods such as Detection Embeddings for Tracking [3], Fairness of Detection and Re-Identification in Multiple Object Tracking [14] etc. jointly learn detection and feature matching in a single network.

IV. Data sets

For this project we used Nvidia Image data set.The Nvidia Image data set data set is a publicly available multi-modal data set by Nvidia. We trained our models on 1200 images of vehicles class. The training samples represents objects with different height and width, different lighting conditions. It also includes multiple object within a single image. Training YOLOv5 on this data set for 10 epochs only took 3 minutes. It is a collection of 1000 driving scenes in two cities that are known for their dense traffic and highly challenging driving situations. We are also going to use some of the NVIDIA-devkit code to create images with bounding boxes and annotations.

V. Evaluation Metrics

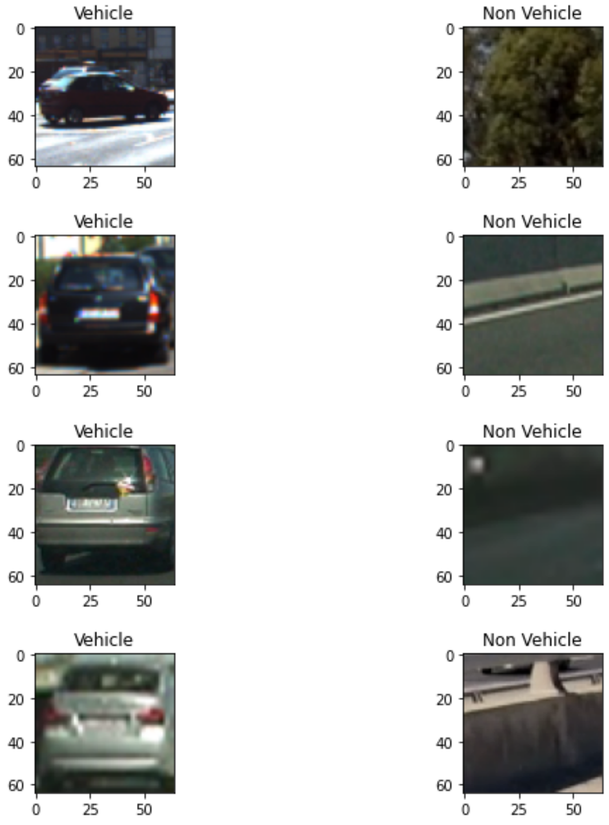
Different standard metrics for detection and tracking are employed to discuss the proposed models and discuss the differences among the approaches taken. By this, found the pros and cons for among the alternatives. Evaluation metrics for detection: Recall and Precision for object identification, Intersection over Union for bounding box's to evaluate the bounding box's localization, Mean Average Precision which accounts for classification accuracy and the bounding box. Evaluation metric for tracking we used are Identified Detections (IDF1) and Multi Object Tracking Accuracy(MOTA).

VI. Approach

A. Vehicle Detection using Histogram of Oriented Gradients(HOG):

We made an attempt to perform a Histogram of Oriented Gradients (HOG) feature extraction on a labelled training image. HOG is a feature descriptor that counts occurrences of gradient orientation in an image. We used a Linear SVM classifier for training. We also implemented a technique of sliding windows and used our trained classifier to search for vehicles in images. Later, we ran our self-made pipeline on a video stream and created a heat map of recurring detections to detect vehicles and reject the outliers.

Fig. 1. Sample Images for Vehicles and Non-Vehicles



Our Approach was as follows:

- We loaded the vehicle and non-vehicle images dataset
- We defined function *GetFeaturesFromHog* which returns hog features and hog images and function *ExtractFeatures* which stacks all features returned from *GetFeaturesFromHog*
- We shuffled the data and then split it into training and testing datasets using sci-kit. This was done to avoid overfitting
- We normalized the data using the Standard Scaler function of Sklearn preprocessing
- We used Linear SVC with default parameters and achieved an accuracy of 98.76%
- We defined the function *slide_window* to implement the sliding window search
- Once we were able to detect the car using the above sliding window method, then we decided to use heat up to plot the bounding boxes around the cars. We defined the function *add_heat* that increments pixel value in the black window at each detected box

- To remove false positives, we used an ‘average’ approach that sums all the heats from the previous 15 frames and then blacks out the pixels that have values less than the threshold value
- Later we implemented the developed pipeline for Video Processing

1) Pipeline Implementation: The Video Processing Pipeline is defined in the *Pipeline* function. We did exactly the same thing in this function as we did in the Image pipeline, with the exception that we saved prior refined windows in a class called *KeepTrack*. After a lot of trial and error, we ultimately kept *25 + len(windows)/2* as the threshold to eliminate the false positives found on the window’s left half.

In each frame of the movie, we noted the positions of positive detection. we constructed a *heatmap* using the positive detection and then thresholded it to find car placements. To identify individual blobs in the heatmap, we used *scipy.ndimage.measurements.label()*. Then we assumed that each blob represented a car. To cover the area of each blob observed, we created bounding boxes.

B. Vehicle Detection using Histogram of Oriented Gradients(HOG):

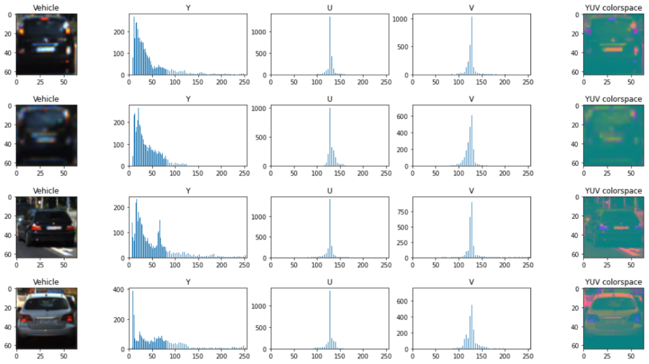


Fig. 2. Extracting Color Spaces for Vehicles Dataset

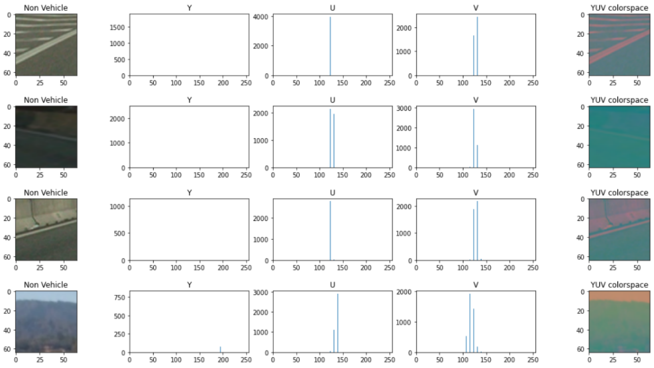


Fig. 3. Extracting Color Spaces for Non-Vehicles Dataset

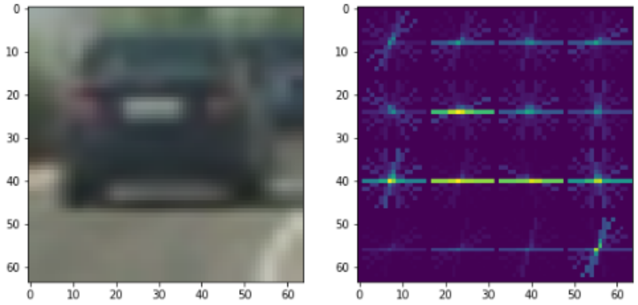


Fig. 4. Extracting HOG of images

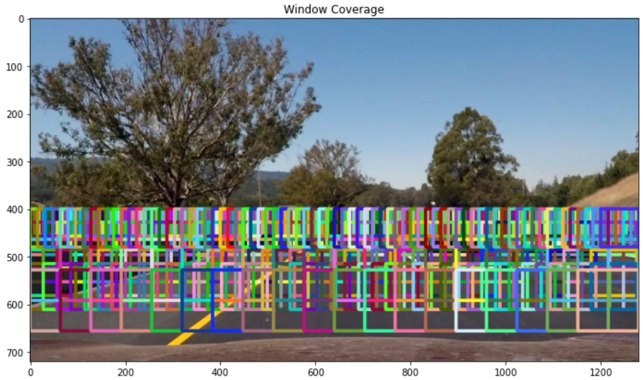


Fig. 5. Image with Sliding Window Coverage

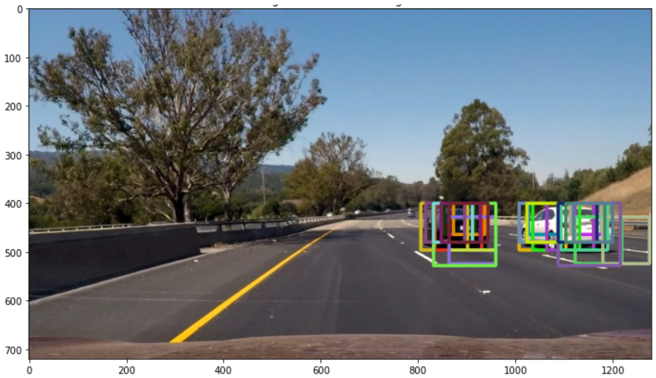


Fig. 6. Image with Refined Sliding Window

In HOG detection technique, we used the labeled sets of vehicles and non-vehicles. We then extracted the color space by creating Histograms. Y, i.e. Luma, determines the brightness of the color. U and V, determines blue and red projections respectively. We chose YUV color space, since they reduce the bandwidth more than RGB capture can. Later, we computed HOG on the test images.

After This we trained Linear SVM classifier and implemented sliding window technique. We searched only the lower half of the image for the cars. We then created the windows where the classifier predicts the output to be a car.

To further refine the search, we implemented Heatmap technique in our sliding window zone. Heatmap will increase the pixel value above the set threshold and black out the pixel below the set threshold, giving us the refined window. Finally,

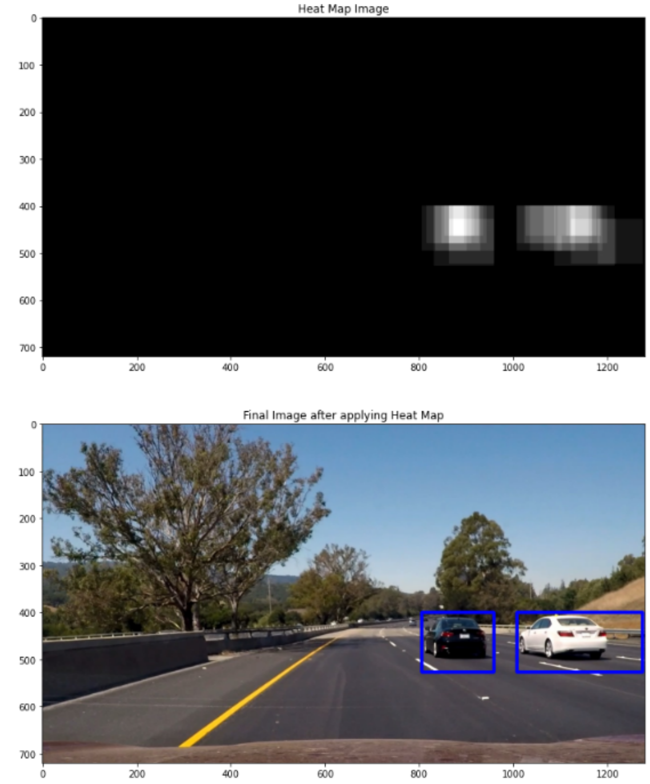


Fig. 7. Heatmap Image on vehicle

we tested the pipeline on the test images and test video.

C. Vehicle detection using YOLOv3

First we trained YOLOv3 on Nvidia images. We trained the model on vehicle classes with 1200 images for 10 epochs. We used the Darknet framework to train YOLOv3 because it is optimized for YOLO object detection. We had to do transfer training after the training it on 600 images. Moreover, we used pre-trained weights from the coco data set before training our custom model. The figure attached below shows the Loss with each cycle. The loss values convereged after a fair number of epochs and that's when we had stopped training. Though, the pre-trained YOLOV3 had known to perform well for objects that are not occluded, it the sampling space of our training data did not seem to generalise the model. So, we couldn't see it perform well and it had performed poorly than the HOG detector. The results for YOLOV3 and its confidence in detecting the vehicle is fairly low as represented in figures 10 and 11.

D. Vehicle detection using YOLOv5

Since the results from YOLOv3 detections were not good enough we decided to use state-of-the-art YOLOv5. We trained YOLOv5 for 1200 vehicle images. The training time was only 3 minutes because YOLOv5 is highly optimized. Instead of using Darknet we used Pytorch for YOLOv5 because Darknet doesn't have a YOLOv5 architecture. Following are the steps to train YOLOv5 on custom dataset:

1) *Install YOLOv5 dependencies:* YOLOv5 only requires installation of Pytorch and some python libraries

2) *Download Custom YOLOv5 Object Detection Data:* Nvidia images data set was used in our project

3) *Define YOLOv5 Model Configuration and Architecture:* Configuring the architecture to get the best performance, i.e change the number of GPUs to use, number of epochs, and network architecture for the best results. We changed the configuration files according to our need and data set by following the guidelines provided by the creators of YOLOv5.

4) *Train a custom YOLOv5 Detector:* Using pretrained weights from the coco data set we started training our model on custom data set.

5) *Evaluate YOLOv5 performance:* After training the YOLOv5 model we used automatically saves the plot of IoU, mAP@0.5, precision, and recall.

6) *Run YOLOv5 Inference on test images:* Testing the model on test data set(Refer Results section)

7) *Export Saved YOLOv5 Weights for Future Inference:* Save the weights for transfer learning if needed.

E. Tracking using DeepSORT

Tracking using SORT is dependent on the on state estimation uncertainty. This makes it has its limitations when it comes to occlusions. However, Sort presents the limitation that if a person hid behind an object and then reappeared, it is assigned a different ID. DeepSort solves this problem by using an AI model that compares similarity between vehicles, thus reducing the issue of switching vehicle's identities [12]. We tried and approached this by first looking and applying already implemented single and multi object trackers. In particular we investigated GOTURN. We learnt about the bullet differences between the offline and online trackers and thus we moved forward with DeepSort. Video containing results and presentations can be found in this link for GOTURN and components of DeepSORT.

There are a few limitations associated with SORT as it is highly dependent on association metric which depends on the state estimation accuracy. For those objects with low state estimation accuracy, identity switches are observed during multiple times. This also leads to ineffective performance in occlusions. To overcome these issues, a deep learning method is proposed which is an extension to SORT. as DeepSORT [12] It replaces the association metrics with a combined motion and appearance metric. Thus implementing a CNN which is trained on large-scale object re-identification data-set gives us robustness against misses and occlusions.

VII. Results

A. Results YOLOv5

1) *Performance Evaluation:*

2) *Training results:*

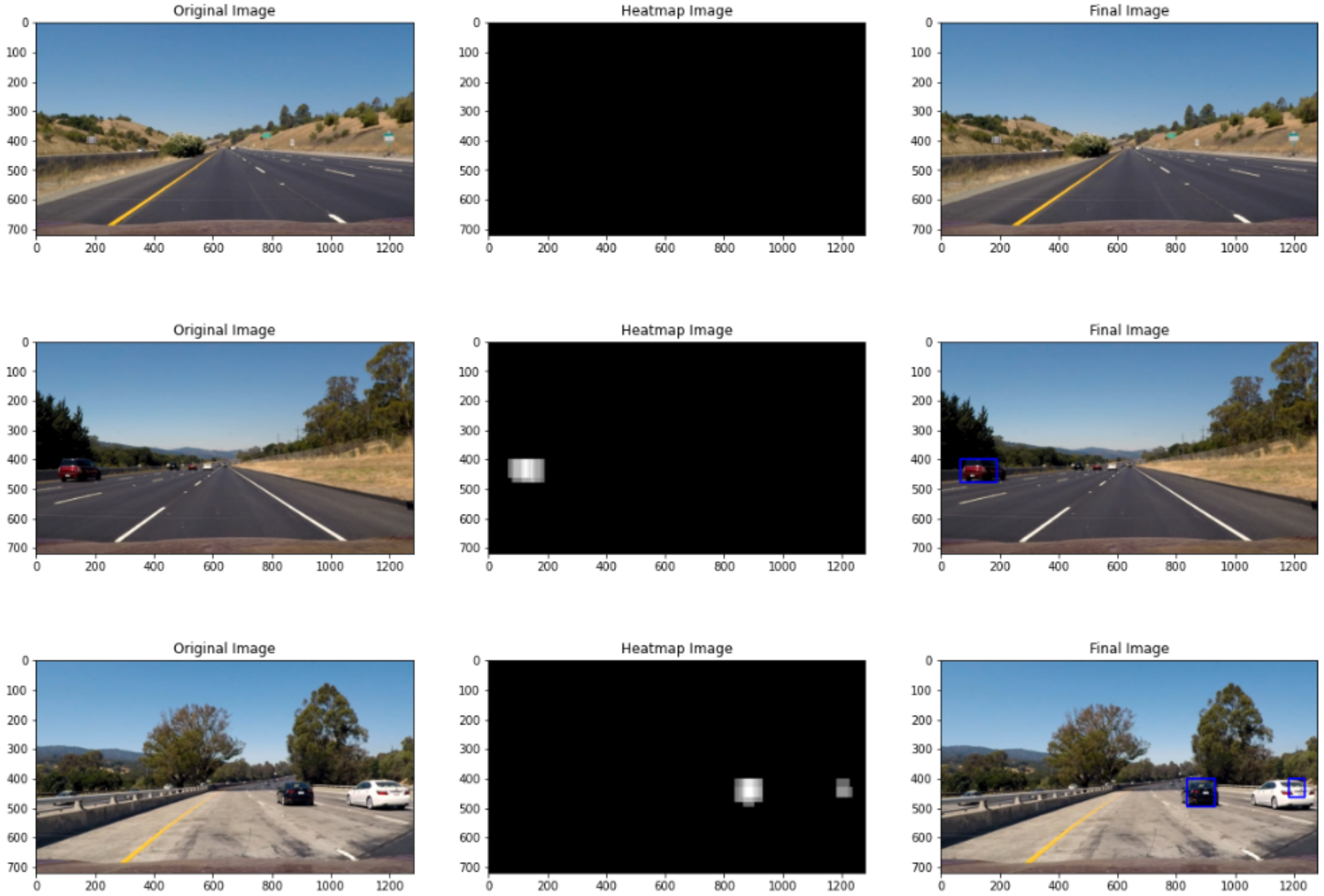


Fig. 8. Heatmap Images: Non-vehicle and Vehicle Images

TABLE I
MAP, Precision, Recall and Bounding Box IoU of HoG, YOLOv3 and YOLOv5

Model(	MAP	Precision (%)	Recall(%)	Bounding Box IoU (%)
HoG	52.29	54.48	55.21	0.67
YOLOV3	0.25	30.78	25.75	0.18
YOLOV5	0.88	81.78	86.75	0.86

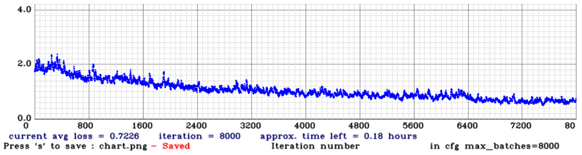


Fig. 9. Loss(Y-axis) vs Iteration no.(X-axis)

TABLE II
Multi Object Tracking Accuracy and Identified Detection Percentage NVIDIA cars dataset

Metric	IDF1	MOTA
SORT ^{1}	51.4	48.2
DeepSORT ^{2}	**61.7**	**59.4**

^{1} [2] ^{2} [12]

VIII. Conclusions

In HOG detection technique, we observed that the algorithm was able to predict the vehicles which were present in the lower half of the image frame. However, following limitations of the implemented algorithm was observed:

1) The algorithm was unable to detect the vehicles when they entered the frame. In other words, partial vehicles could not be recognised.
2) The algorithm was able to detect only those vehicles that were big enough(or takes more area) in the zone of identification. In the below figure, the white car could not be identified, even though it was present in the zone

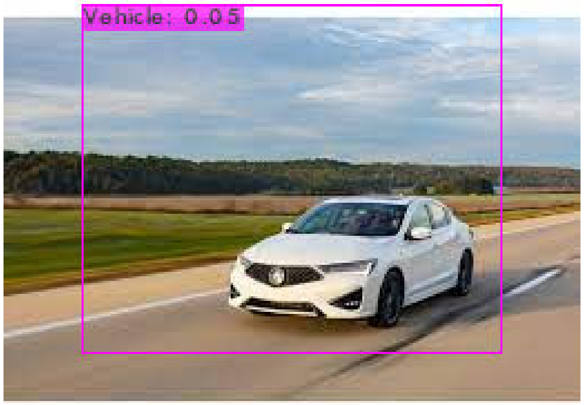


Fig. 10. Test results after training yolov3 on 600 images

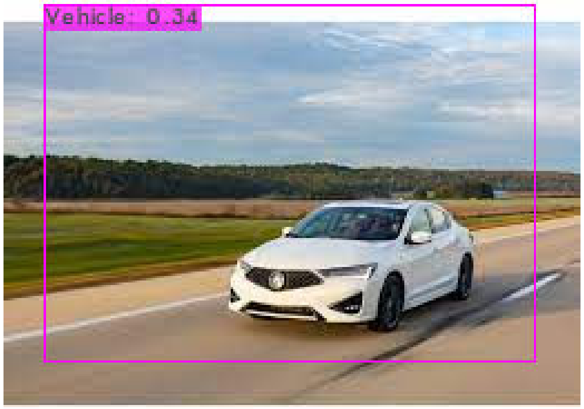


Fig. 11. Detections on test images after training on 600 more images

Fig. 12. training loss ofr

of identification.

3) The algorithm was intended to identify vehicles only in the lower half of the image frame. It could not capture the vehicles that were present in the above half of the image frame. In the below image, the vehicle present in the red circle could not be identified.

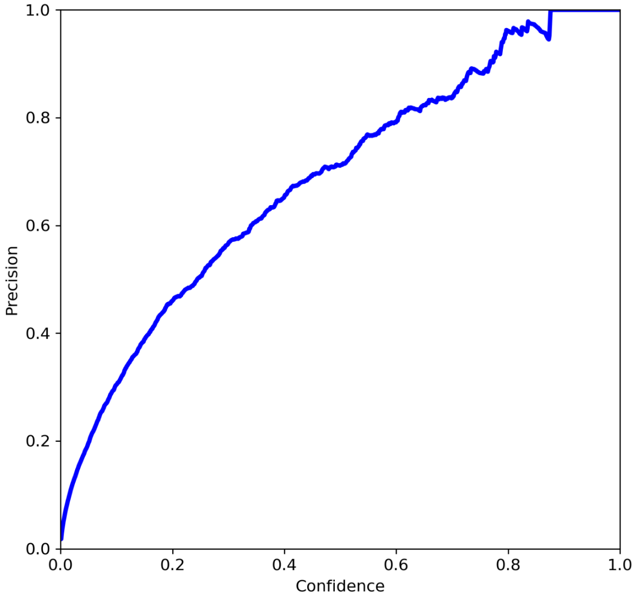


Fig. 13. Average precision= 87.88

4) The algorithm recognized only the vehicles rear part and not the front part. In other words the algorithm predicted vehicles present only in the same lane and not the vehicles coming from the opposite direction in the other lane, i.e., it is unable to detect the cars highlighted in the red circle.
5) Some false positives and false negatives were also visible. In the below image, no vehicle was present in the center blue box.
6) Only cars were detected and the algorithm fails to identify the vehicles other than cars, say bikes

These can be improved further by:

1) diversifying the dataset containing images of partial models and exhaustively training the model.
2) expanding the dataset that contain images of vehicles occupying less area
3) modifying the algorithm so that it can detect the vehicles in the entire image frame
4) training the algorithm with images containing the front part of the vehicles as well
5) fine-tuning the threshold values to minimize false positives and false negatives
6) augmenting the dataset containing images of multiple vehicles, like trucks, bikes, motorbikes, aeroplanes, helicopter etc.

The output video can be found at this link and detailed presentation on HoG can be found in this video.

Acknowledgment

We respect and thank Prof Gennert for providing us a golden opportunity to do the project work on this topic under his guidance. We owe our deep gratitude to him for taking keen interest in this project work and guiding us all along, by

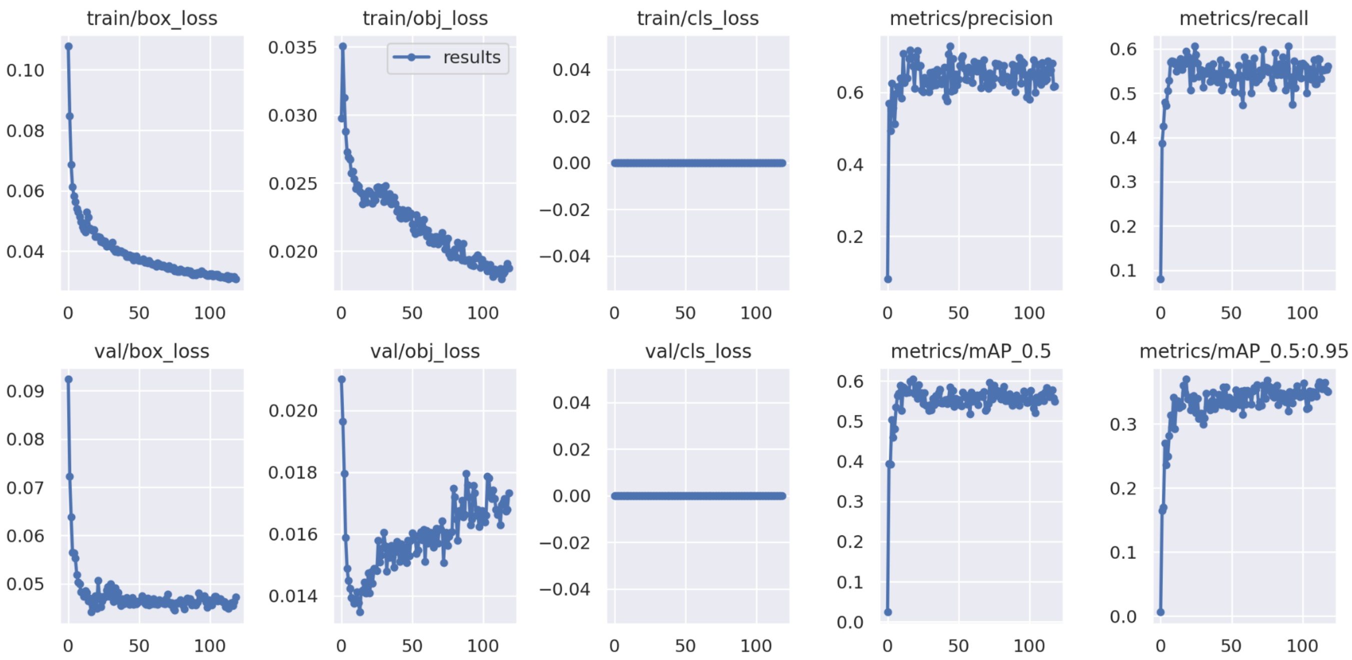


Fig. 14. YOLOv5 training results

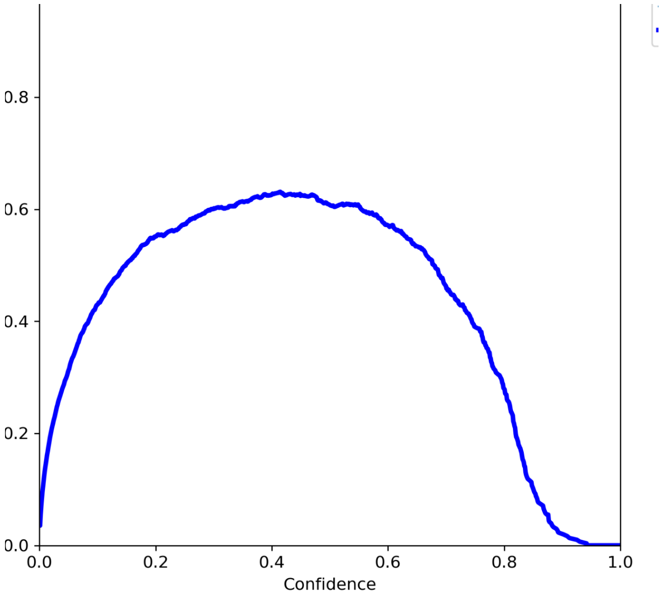


Fig. 15. Confidence vs F1 score

providing all the necessary information and suggestions during this project work.

IX. Appendix

A. Vehicle Detection using Histogram of Oriented Gradients(HOG):

- https://github.com/AlexeyAB/darknet
- https://github.com/ultralytics/yolov5
- https://github.com/sameermalikjmi/trackingOnline

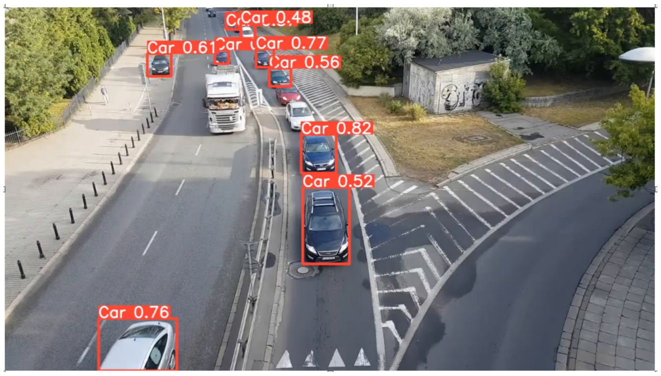


Fig. 16. Dectecion on test image

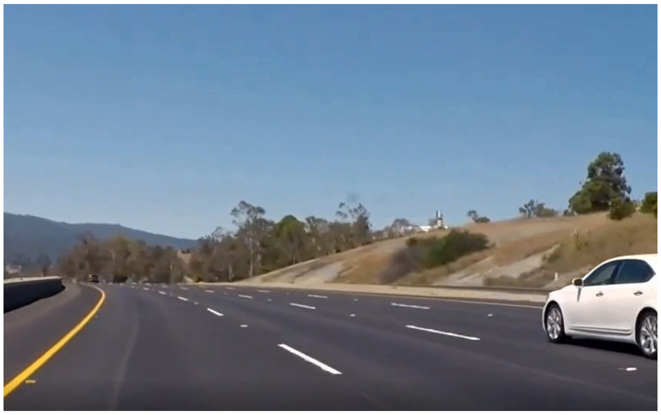


Fig. 17. HOG detection issue 1: Unable to detect partial vehicle

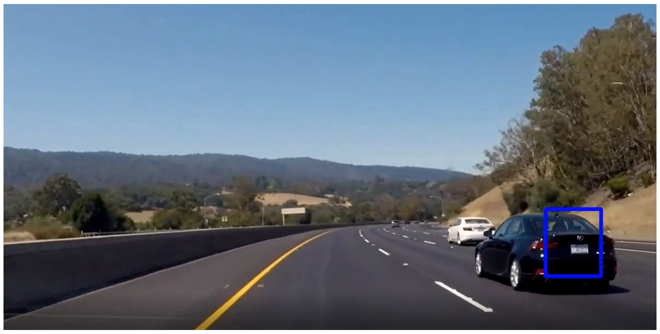


Fig. 18. HOG detection issue 2: Unable to detect white car

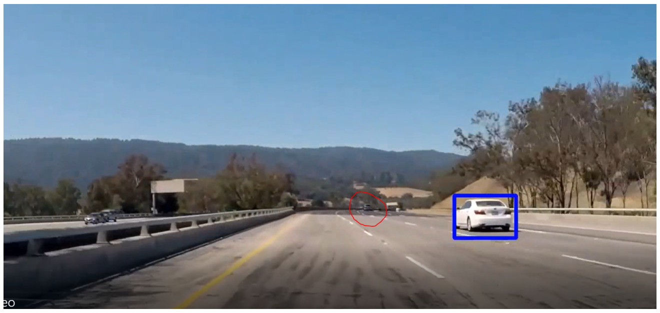


Fig. 19. HOG detection issue 3: Unable to detect car in the red circle

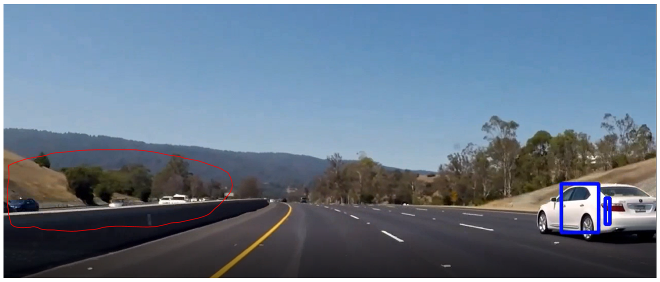


Fig. 20. HOG detection issue 4: Unable to detect incoming cars

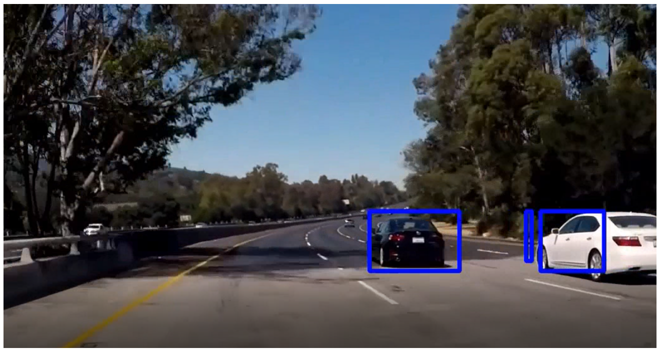


Fig. 21. HOG detection issue 5: False positive

References

[1] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, 2012.

[2] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016.

[3] Mohamed Chaabane, Peter Zhang, J. Ross Beveridge, and Stephen O'Hara. DEFT: detection embeddings for tracking. *CoRR*, abs/2102.02267, 2021.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[5] Milind Naphade, David C. Anastasiu, Anuj Sharma, Vamsi Jagrlamudi, Hyeran Jeon, Kaikai Liu, Ming-Ching Chang, Siwei Lyu, and Zeyu Gao. The nvidia ai city challenge. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–6, 2017.

[6] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017.

[7] Paolo Piccinini, Andrea Prati, and Rita Cucchiara. Real-time object detection and localization with sift-based clustering. *Image and Vision Computing*, 30(8):573–587, 2012. Special Section: Opinion Papers.

[8] Nicola A. Piga, Ugo Pattacini, and Lorenzo Natale. A differentiable extended kalman filter for object tracking under sliding regime. *Frontiers in Robotics and AI*, 8:251, 2021.

[9] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[10] Rohit Kumar Tiwari and Gyanendra K. Verma. A computer vision based framework for visual gun detection using harris interest point detector. *Procedia Computer Science*, 54:703–712, 2015. Eleventh International Conference on Communication Networks, ICCN 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21-23, 2015, Bangalore, India.

[11] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. MOTS: multi-object tracking and segmentation. *CoRR*, abs/1902.03604, 2019.

[12] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017.

[13] Norhidayah Mohamad Yatim and Norlida Buniyamin. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767(20), 2018.

[14] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. A simple baseline for multi-object tracking. *CoRR*, abs/2004.01888, 2020.

[15] Xingkui Zhu, Shuchang Lyu, Xu Wang, and Qi Zhao. Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios. *CoRR*, abs/2108.11539, 2021.