CS 543 / ECE 549 Computer Vision Final Project

Lane Tracking in Atypical Conditions

Amber Sahdev, Justin Yang, Nathan Narasimhan, Navid Mokhlesi
University of Illinois at Urbana-Champaign
Champaign–Urbana, Illinois
{asahdev2,justiny2,nanaras2,navidm2}@illinois.edu

6 May 2019

1 Abstract

Recent developments in autonomous vehicles have been fueled by improvements in object recognition by such vehicles' systems. In order, however, for these systems to reach high-level autonomy, whereby they can function without human intervention in many environments, object recognition must be improved in not just typical conditions of clear, lighted driving in the daytime, but also in atypical conditions, such as raining conditions at night. These atypical conditions present certain challenges for prior art: existing models are trained largely on data from the daytime and rainy conditions introduce patches of water on the windshield or in the field of view. Additionally, light sources introduced by the reflections of streetlights or taillights off puddles on the road make recognition even more difficult. Here, we present techniques we have undertaken to improve the accuracy of object recognition, in particular, other vehicles, under these conditions and to detect lanes in order to plot a course for the vehicle to drive.

2 Introduction

Autonomous vehicles are complex devices with many sensors. In particular, visual cameras and lidar are used to provide specific details about the environment the vehicle is driving in. These details, along with other contextual details such as the GPS location and route, allow the vehicle to make inferences about driving paths and speed. We use images from vehicles perspectives to plot an instantaneous path to drive given a frame, and determine the location of nearby vehicles on the road. We then plot a path that follows the vehicle immediately ahead, drawing lane lines in the process.

3 Details of the approach

We primarily tried three approaches to determine the location of vehicles and lane lines in rainy conditions at night.

The first approach involved object detection of tail lights of vehicles. We began by using traditional computer vision approaches to determine the location of tail lights as high-intensity spots of concentrated red light. One problem that we ran into immediately was the probability of false positives. Other objects that are not actually tail lights were detected as tail lights, such as red traffic lights and building signs. But the most glaring problem was the reflections in puddles on the road which created our input and processed images, both, very noisy.

Figure 1: City driving at night in rain

Figure 2: Highway driving in rain in the day

Figure 3: Driving after the rain at dusk

As evident in Figure 1, using the traditional approach failed us because of the reflections off the water on the road.

Initially, we thought that isolating the channels in the image could allow us to detect the brake lights in a fairly cohesive way, but there were substantial problems with using the same methods on multiple images. The results look like what is shown in Figures 4 and 5.

Figure 4: Red channel of Figure 3

Figure 5: Red channel of Figure 1

Figure 4 shows a reasonable car image in the center, but Figure 5 does not. These results happen because the light noise is unpredictable as of now. This idea does not seem to have too much promise, so we decided to move on.

Another interesting pitfall that we noticed with only extracting the red color from the images and processing that was that a lot of cars are red. This realization, in part, made us take the deep learning approach more seriously.

The second approach we tried to was running object detection on tail lights of cars. We tried to implement Faster R-CNNs[1], however, we had a hard time finding suitably labeled datasets for our approach.

Finally, we pivoted to recognizing whole cars instead of just taillights. Our new approach used YOLOv2[2] to detect cars in images and use this information to fit a polynomial to find the drivable area.

While the results were passable, we needed better accuracy and confidence. We employed two methods.

1. Preprocessing and enhancing the low light images before inference
2. Retraining YOLO on labeled low-light car dataset

For preprocessing the images, we wrote `image_enhancement.ipynb`. The results after preprocessing are more representative of the training data in YOLO. In particular, we performed image transformations to brighten the images and provide additional contrast based on an approach by Ying et al.[3] The results of the preprocessing step on some images are shown in Figure 6.

Figure 6: Image preprocessing where originals (left) are enhanced and produce results (right) with better contrast

Following the results of this approach, we attempted a new approach where we re-trained YOLO on a low-light image dataset, Exclusively Dark Image Dataset (ExDark).[4] We had to figure out how to train our approach of YOLO, darkflow, on this dataset. To do this, first we wrote a script to reformat the ExDark labels in the format that darkflow expects them; this code can be found in `txt_to_xml_annotations.py`.

Next, because we were running inference on only one object, we had to change the standard YOLO configuration and the last regional and convolutional layers to match our desired output size.

For the convolutional layers, we had to change the filters parameter to `filters=(number of classes + 5)*5=30` Similarly, for the region layer, we had to change the classes parameter to 1. Next, we re-trained YOLO on this new ExDark dataset for 50 epochs with a learning rate of 10e-5.

In the end, our inference procedure included running the images through our image enhancement algorithm and then through the neural net.

We also expanded our approach to infer on video data, however, we weren't able to reasonably perform image enhancement on video due to speed issues because we weren't able to parallelize the image enhancement code enough. Therefore, video inference runs without enhancement.

4 Results

Figure 7 shows some of the early results we came up with using YOLO without image preprocessing enhancement.

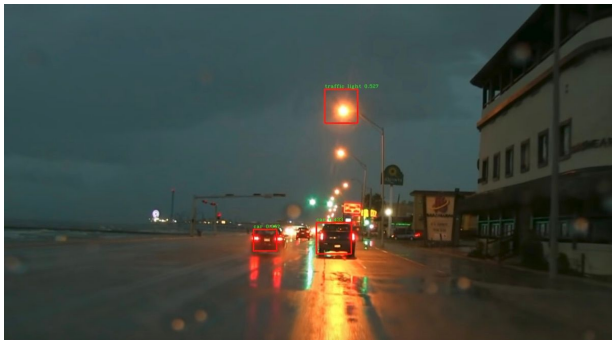

Figure 7: Tail light detection with street light false positive

In figure 7, we can see a case where our detection algorithm works and is able to find the cars in front of us. We later fit the 2D polynomial to find the lane. This is done by using points on the bounding box and the relative size of the image. We then draw the lane

estimators whose parameters have been tuned empirically.

Figures 8-11 are some examples of where we detect the car and draw the lane lines properly.

Figure 8: Original image with vehicle in front stopped at traffic light

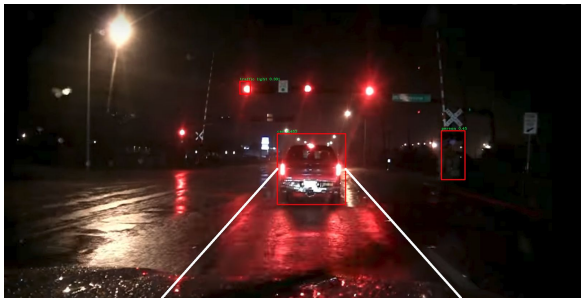

Figure 9: Bounding box and lane line generation based on figure 8

Figure 10: Original image with vehicle driving ahead

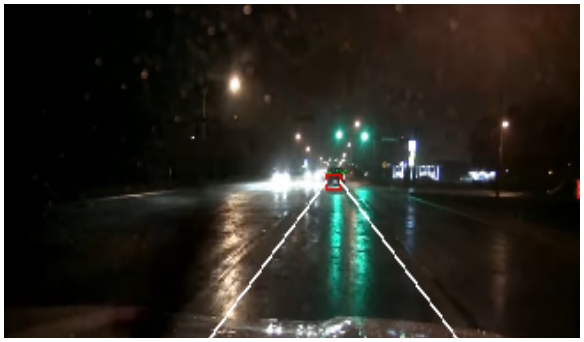

Figure 11: Bounding box and lane line generation based on figure 10

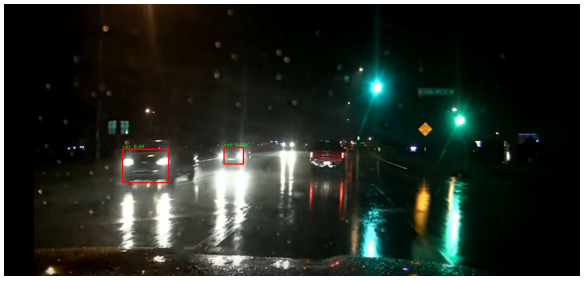

Figure 12: Failure where bounding boxes only recognize vehicles on the opposite side of road

Figure 12, however, is a case that we intend to solve but fail for now. The object detection algorithm was not able to recognize the car in front of us. We plan to solve this by training our deep net on more images of cars in low-light, rainy conditions or by adjusting the images to make the different cars more apparent so that the current model can find the cars.

After obtaining these results, we have extracted the coordinates of the autonomous vehicle and fit a curve to its position to determine the drivable area.

Initially, we ran our implementation on our local machines but we were also able to install all the required dependencies on a GPU cluster we have access to. On the cluster, we are also able to perform the same inference but much faster, which is necessary for the real-time application of our inference.

The results run with our YOLO technique using image preprocessing enhancements are shown in Figure 13.

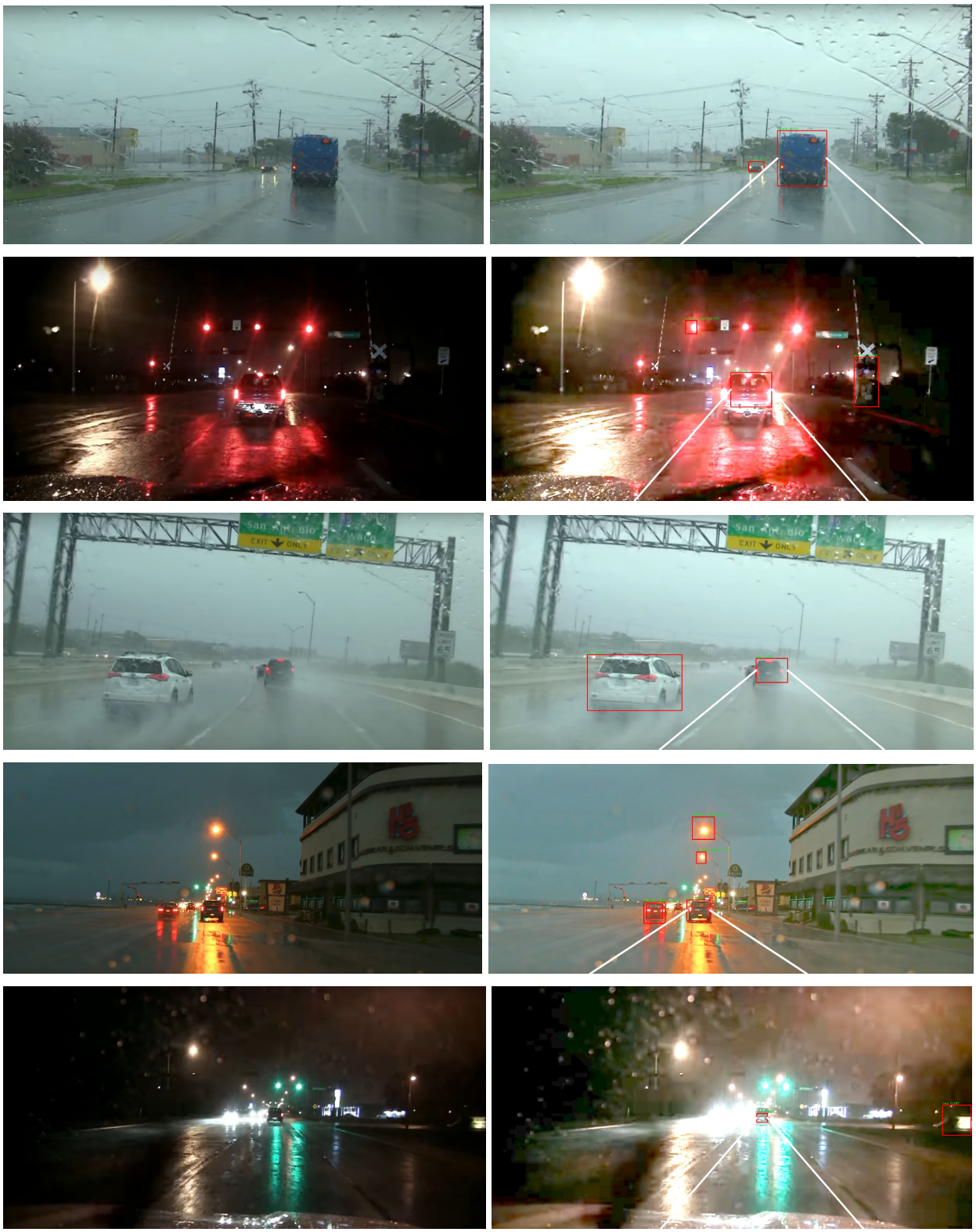

Figure 13: Original images (left) are preprocessed and generate results (right) through our technique. Note that most of the input images don't have any hint of the lane markings on the road; however, the output images from our technique have clearly marked drivable areas.

5 Discussion and conclusions

YOLO is trained on well-lit datasets. The problem with this training dataset is that the training set is not representative of the scenarios that are we interested in—specifically, the use of low-light images in rainy conditions. Therefore, we used ExDark to further train YOLO, which has 639 labeled low-light vehicular images.

For training, first, we wrote a simple Python script which transformed the raw label data provided into the data format expected by YOLO. We then wrote a configuration file `yolo_darkcars.cfg` based on the standard YOLO `configuration yolo.cfg` to change the size of the last convolutional layer and region layers because instead of outputting 20 labels, the labeled data only outputs one label, corresponding to the singular vehicle in the image. Following the modification to the configuration file, we loaded the standard `bin/yolo.weights` and trained on this new ExDark dataset.

Additionally, we researched R-CNN[5], Fast R-CNN[6], and tried implementing Faster R-CNNs but in the end, went with YOLO because of the speed. Additionally, YOLO is a popular framework, which was useful in terms of community support.

6 Individual contributions

The authors regularly meet, typically for a number of hours on the weekends on a weekly basis. We used this time to make progress on the project, including discussing next steps and writing code. J. Yang set up accounts for the team to use on a GPU cluster that we have access to, so we maintain code in Git and push to the cluster the latest changes.

We have largely been working together on most of the projects. N. Narasimhan looked into different cropping and color spaces; for example, cropping out the front of the vehicles at the bottom of the field of view, since the presence of the vehicle hood itself is not desirable. A. Sahdev worked on object detection techniques using TensorFlow, YOLO, and darkflow and trained a couple of different models. Narasimhan helped Sahdev with different models to try out and has also looked into RGB segmentation. Narasimhan also wrote the lane visualization code. Yang took care of setting up some of the scaffolding of the notebook, including reading and displaying the images/videos. N. Mokhlesi has been responsible for researching different techniques we can use, as well as looking for data sources and labeling some of the data that we are using for training. Sahdev, Narasimhan, and Yang worked on some of the later work such as the image preprocessing enhancements.

7 References

[1] S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017. doi: 10.1109/TPAMI.2016.2577031. URL: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7485869&isnumber=7919342`.

[2] J. Redmon, Joseph and Farhadi, “YOLO9000: Better, Faster, Stronger,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 6517-6525. doi: 10.1109/CVPR.2017.690. URL: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8100173&isnumber=8099483`.

[3] Z. Ying, G. Li, Y. Ren, R. Wang and W. Wang, “A New Image Contrast Enhancement Algorithm Using Exposure Fusion Framework”, Computer Analysis of Images and Patterns, pp. 36-46, 2017. doi: 10.1007/978-3-319-64698-5_4. URL: `https://www.researchgate.net/publication/318730125_A_New_Image_Contrast_Enhancement_Algorithm_Using_Exposure_Fusion_Framework`.

[4] Y. Loh and C. Chan, "Getting to know low-light images with the Exclusively Dark dataset", *Computer Vision and Image Understanding*, vol. 178, pp. 30-42, 2019. doi: 10.1016/j.cviu.2018.10.010. URL: `https://github.com/cs-chan/Exclusively-Dark-Image-Dataset`.

[5] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich Feature Hierarchies for Accurate Object

Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 580-587. doi: 10.1109/CVPR.2014.81. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6909475&isnumber=6909393.

[6] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448. doi: 10.1109/ICCV.2015.169. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7410526&isnumber=7410356.