

Beyond Grand Theft Auto V for Training, Testing
and Enhancing Deep Learning in Self Driving
Cars

Mark Anthony Martinez II

A MASTER'S THESIS

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF MASTER OF ARTS

RECOMMENDED FOR ACCEPTANCE BY

THE DEPARTMENT OF

COMPUTER SCIENCE

Adviser: Professor Alain L. Kornhauser

June 2018

©Mark Anthony Martinez II, 2018. All rights reserved.

Abstract

As an initial assessment, over 480,000 labeled virtual images of normal highway driving were readily generated in Grand Theft Auto V's virtual environment. Using these images, a CNN (Convolutional Neural Network) and a RNN (Recurrent Neural Network) were trained to detect following distance to cars/objects ahead, lane markings, and driving angle (angular heading relative to lane centerline): all variables necessary for basic autonomous driving. Encouraging results were obtained when tested on over 50,000 labeled virtual images from substantially different GTA-V driving environments. This initial assessment begins to define both the range and scope of the labeled images needed for training as well as the range and scope of labeled images needed for testing the definition of boundaries and limitations of trained networks. It is the efficacy and flexibility of a "GTA-V"-like virtual environment that is expected to provide an efficient well-defined foundation for the training and testing of Convolutional Neural Networks and Recurrent Neural Networks for safe driving.

Contents

Abstract	ii
1 INTRODUCTION	1
2 Related Work	3
2.0.1 Interpreting Image Data	5
3 Calculating affordance variables from images	7
3.0.1 Data Collection Using Grand Theft Auto	8
3.0.2 Learning Model: CNN Model for Images	10
3.0.3 Learning Model: Using CNN Output as input for a Recurrent Model	11
3.0.4 Description of the Affordance Variable Collection	12
3.0.5 Diversity of Scenarios	14
4 GTA Results	15
5 Discussion	18
6 Conclusion	20

Acknowledgments

I'd like to thank Professor Kornhauser for giving me the opportunity to work on such an amazing application of deep learning and for being a great mentor.

My parents, and specifically my mom for making sure that I stay on track of priorities.

And the wonderful people who have made my experience at Princeton truly one to remember.

Chapter 1

INTRODUCTION

Self driving cars have the potential to change the paradigm of transportation. According to the U.S. Department of Transportation National Motor Vehicle Crash Causation Survey, 93% of all vehicle accidents are influenced by human error while driving; eliminating most of the accidents caused by human error would be a giant boon for public safety. Self driving cars also promote ride-sharing which would reduce environmental impact immensely. Additionally, parked cars take up tremendous space in highly populous cities that could be otherwise used for increased living spaces. There have been many advancements in the self driving car space with many companies investing heavily to be the first player in the market. Google's Waymo has accrued over 3 million miles autonomously. Tesla already has a functional autopilot system that allows a car to stay within lanes while it is driving with minimal human intervention. But as much progress has been made there is much work to reach self driving capabilities. There have been several highly public cases where self driving car technology has failed such as the case in Arizona and the publicized in Florida when a Tesla vehicle collided into a truck on the highway. Even more recently the self driving car division at Uber has had to cease public testing after a fatal crash in Arizona.

Part of the reasons that the systems set in place to drive did not function correctly is because of a lack of data to account for corner cases. The real world offers only one reality (scenario) at a time that is in many cases extremely challenging and costly

to create, replicate and iterate. Virtual reality, while only approximating reality, does provide the opportunity, if properly designed, to readily create scenarios, that readily capture the "labeled" data ("affordances") of those scenarios and effectively investigate neighboring variances of those scenarios. The availability of such environments to test and understand the robustness and effective range of automated driving approaches is thought to be an effective element in the design of those approaches. While safe behavior in the real world is the final test, the availability of a robust VR environment is thought to be an very effective tool.

Chapter 2

Related Work

Research on self driving cars has existed starting from the 1930s [17] and was more seriously considered in the 1980s with the ALVINN project. In that project, a neural network with one hidden layer was used to map input images directly to steering angles. [11, 12] Later on, Aly created a lane marking detector as well as a small testing dataset on roads in urban settings in 2008. [2]

CNNs have been used in driving like scenarios since 2005 when Muller [10] proposed a CNN based off road driving robot, DAVE, that mapped images to steering angles. In a more recent publication in 2015, Huval [8] described a CNN system that detects vehicles and lane markings for highway driving showing that CNNs have promise in autonomous driving.

Virtual data has had a successful history in computer vision. Taylor et al. (2007) used a computer game Half-Life to create a system for evaluating tracking in surveillance systems [13]. In a paper released in 2016 [16] video game data was used to augment real datasets to provide coverage for scenarios that were difficult to find data for in the real world.

In 2016-2017, there were several publications that used GTA-V and other video games for autonomous driving testing and training [6, 4]. In Filipowicz and Liu's paper [6] a system using GTA was used to detect the distance to stop signs from an in game image.

Since early on in the development of self-driving technology physical test tracks have been used to evaluate self-driving algorithms [15]. The final benchmark for system's performance will be real world testing, but with rigorous and complex examples of labeled datasets it is possible to evaluate many corner cases. In 2012 Geiger et al. released the KITTI dataset which has become the preeminent dataset in testing self-driving technologies. [7] The value in the labeled data stems from the complexity and the number of examples it contains. Just as the KITTI dataset provided immensely useful labeled data to train and test on, it is possible to use a virtual environment to create varied and difficult test cases for self-driving technologies.

The first neural networks that utilized temporal information were developed in the 1980s, but the first temporal approach at deep learning happened in 1993 with 1000 layers in a recurrent neural network (RNN). [14] The next big strides with temporal deep learning happened around 2007 with the paper by Fernandez et al. that revolutionized how speech detection worked [5]. Although there has been a significant amount of development in deep learning that utilizes temporal information most Deep Learning in the field of vision is divorced from the time component. It is perhaps because many tasks used for identification do not need to factor in changes in an image rather than identification or labeling. However, one area where temporal neural networks have had some impact is in the field of labeling an image based on it's contents. As evidenced by this paper published in 2016 by Byeon et al. we can see that LSTMs, Long Short Term Memory blocks, (a form of a recurrent neural network) were used to label an image of it's contents. [3] Largely RNNs have been limited to the scope of image labeling, and have not ventured far into deriving information that may be time sensitive as in video data.

2.0.1 Interpreting Image Data

Traditionally, approaches to model a system's reaction to visual input include behavior reflex and mediated perception. The behavior reflex approach uses learning models which internalize the world model. Pomerleau utilized this method to map images directly to the system's actions such as steering angles and speeds [12]. This scheme extensively exploits the capability of deep learning. Nevertheless, it obscures the decision process being made by the network, leaving only a black-box system that handles exceptionally high-risk tasks. This is certainly an undesirable property to manufacturers and users.

On the other end of the spectrum, mediated perception detects important features using learned models and then builds a world model based on these features. For example, a mediated perception system uses a combination of many subsystems such as vehicle detectors, and pedestrian detectors to locate as many objects as it can in the driving scene. The approach often creates unnecessary complexity by extracting information irrelevant to the driving task. With consideration for the shortcoming of these approaches, Chen et al. [4] proposed a direct perception approach.

Direct perception creates a model of the world using a few specific indicators extracted directly from the input data. These indicators, called affordances, are often distances that can be used directly by a simple controller to decide on actions for the system to take in real time. The direct perception approach lies in the middle of the spectrum between the other two approaches and takes advantage of both systems by utilizing the network's capability of accurately retrieving affordances and at the same time keeping the controller explicit and thus, accountable.

Using the open source driving virtual environment, TORCS, Chen et al. trained a computer to learn a transformation from images into several meaningful affordance indicators needed for highway driving. Their work demonstrated that this approach works

well enough to drive a car in a virtual environment, and generalizes to images in the KITTI dataset, and to driving on US 1 near Princeton. It is believed that an autonomous driving system can be designed with the camera playing the role of the human eye in a direct perception approach.

Chapter 3

Calculating affordance variables from images

Using the direct perception model as an inspiration, the following 8 affordance variables for the car were derived from a deep learning model: steering angle (angle), distance to the car in the left lane (car_L), distance to the car directly in front (car_M), distance to the car in the right lane (car_R), distance to the lane marking to the far left (lane_LL), distance to the lane marking immediately to the left (lane_L), distance to the lane marking immediately to the right (lane_R), and distance to the lane marking to the far right (lane_RR). In Chen et al., 14 similar variables are chosen; however, some of them appear to be redundant and only complicate the model. (a) and (b) are schematics demonstrating the measurement of all 8 affordances.

Some of the affordances (car_L, car_M, car_R, lane_LL and lane_RR) might not be applicable in some cases. For example, when there is no car in front, car_M should not contain any value. Or when the car drives in a two-lane road, either lane_LL or lane_RR must be invalid. In this situation, these affordances are set as "inactive." To indicate the inactive state, the value of these affordances is set to a specific number slightly out of their normal range. This design decision comes into play later on in the training process of the neural network.

3.0.1 Data Collection Using Grand Theft Auto

CNNs have shown to be extremely powerful in image classification. The downside that makes CNNs, as well as other supervised machine learning models, often impractical is the lack of appropriate training data. It is not only the dearth of the data, but the quality of the data as well. In order to be effective, machine learning models must be trained on a large number of annotated data that provides enough information for the models to learn from. The greatest obstacle to obtain a usable dataset is often its cost.

There are several major hurdles if one were to use real data for this task. The largest hurdle is the number of data points that would be necessary. In general, 100,000 images would be barely sufficient to train a complicated model. For our purpose, a car would need to drive around recording driving scenes. It is not only expensive to do this manually, but also extremely time consuming. People would have to manually drive a car and collect video data which is limited to geo-location of the car as well working conditions of drivers.

The other hurdle that one would encounter in using real data, is obtaining the ground truth labels for each image. In our case, the model needs to pick up eight different affordance variables from each image. Even if a car is equipped with the latest in tracking technology it is not possible for a car to pick up the exact distances of cars in front of it as well as the distance to the lanes in all occasions. For instance, lidar is extremely effective but fails when rain is introduced into the scene. The instruments would not be perfectly able to pick up the distances because of normal mechanical error and even if the instruments do pick up a distance to a lane marking or a car there is no assurance that that value will indeed be correct. A human would need to go through the data to annotate the information and look for incorrect labeling which is again extremely time consuming.

The benefit of GTA-V is that data can be collected without regard to limitations

of the real world in terms of time or cost. There is no cost associated with each test run and the game can run indefinitely until the adequate amount of data is collected. Additionally, GTA provides a multitude of different environments that allow the model to train and test on a variety of scenarios that mimic the real world. Of equal importance is that the game will have perfectly annotated data with the ground truth labels. Costly and possibly imprecise sensors are not needed to find the exact measurements of the affordance variables. The labels needed for each image can be found exactly as the game interprets them.

GTA-V is not normally an academic tool and therefore fan game modification scripts were used to obtain the values for the distances, to set up the simulations, and to run the simulations used for this research. The GTA mod Rage Plugin Hook [1] was used with collaboration from Lennart Meincke, the creator of the mod, to extract the information from GTA-V to find the affordances.

In the data collecting process, a simple car model called "Futo" was chosen and a loop which has the character strictly drive on highways to a random position was set up. The car frequently respawns at a random position. Since the data collection takes at least several days of in-game time the data covered all different time of a day. The camera for taking a screenshot is positioned behind a windshield at a similar location to that of a front-facing camera in an autonomous vehicle.

A screenshot is taken from the window of the game screen, rescaled to 210×280 pixels and stored as a Bitmap file. One screenshot is taken every 5 iterations of the script's loop which is approximately 250 ms. Because of how the script works, taking a screenshot every iteration of the loop introduces a higher variation of the interval between each screenshot. For future work a neural network model that incorporates temporal information will be used necessitating the screenshots to be as equally spaced in time as possible.

3.0.2 Learning Model: CNN Model for Images

The direct perception CNN is based off of the standard single-stream AlexNet architecture. [9] The model is built in Keras using TensorFlow as the backend. Since estimating real values is a regression task instead of an Alexnet's classification based approach, some modifications to the network were made:

- The shape of the input layer is changed from $224 \times 224 \times 3$ pixels to $210 \times 280 \times 3$ pixels to match the resolution of the screenshots from GTA-V.
- An extra fully-connected layer is added at the end of the network to serve as the output layer consisting of 8 neurons. Alternatively, instead of adding a layer, one could modify the last layer of AlexNet to have 8 neurons.
- The activation function of the output layer must also be changed from softmax to hyperbolic tangent to output a vector of real values in a limited range $[-1, 1]$.

The network is trained with Adam optimizer with an initial learning rate of 0.001, and mean-squared error is chosen as a loss function. The model is trained for 21 epochs under the training set of approximately 350,000 images with a batch size of 32 images. The model quickly converges after the first few epochs.

All input images are mean-subtracted before being fed into the network. Before training and evaluating, all active affordances are rescaled to lie in a range $[-0.9, 0.9]$ to match the output layer of the network, $[-1, 1]$, and still leave some margin which is necessary to distinguish between active and inactive affordances.

Inactive affordances are rescaled with the same factor. Consequently, they result in a value larger than 1 which is outside of the achievable output range of the network. As a result, the loss will never reach zero regardless of training epochs. Still, this method is desirable for determining whether the affordances are active (i.e. if the output of the network is close to 1, then, with high probability, it is inactive). The inactivity threshold

was set to be 0.99, and thus, any output above this threshold is considered inactive.

3.0.3 Learning Model: Using CNN Output as input for a Recurrent Model

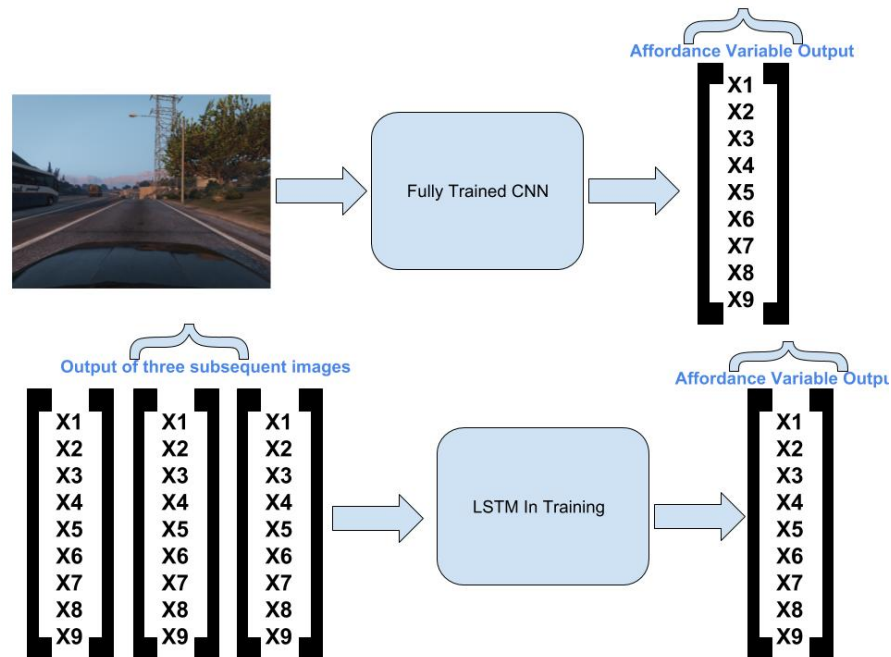


Figure 3.1: An image is run through a fully trained CNN and its output is collected. If the subsequent two images also exist then the three outputs will be stitched together to form one input for the RNN model where the label is the affordance vector of the next image in the sequence.

In addition to using a Convolutional Model to interpret the image data to output affordance variables a Recurrent Neural Network Model (RNN) was used to see if the output for the model could be improved by incorporating the time dimension. This model was built in Pytorch and took as input the outputs of the CNN from the previous section. Because it is not possible to have a purely recurrent model analyze an image the image first needed to be run through a fully trained CNN.

Once an image was run through the trained CNN mentioned in the previous section its output was saved and then stitched together with the outputs of two subsequent images snapshots. These output are all combined to create one data input for the RNN where the label would be to predict the affordance variable for the next, unseen frame.

Figure 1 illustrates this process.

The RNN model has two RNN cells that get fed sequentially the affordance vectors for the images as person would see it. This sequential input into the cells is important so that the network gets a sense of what the change in the images over time. As such the images are not shuffled as they would normally be in training and testing to maintain this integrity.

The number of images used to create training data is 116764 and the number of training data created for training the RNN model is 116764.

3.0.4 Description of the Affordance Variable Collection

The affordance variables were calculated by using in-game constructs that were available through the mod.

The steering angle was easily calculated by comparing the direction of the road with the direction of the car. The lanes of the highway were calculated by using GTA-V's node system that demarcates where roads are. The nodes in the game denote roads, but do not show exactly where lanes are. GTA however indicates whether the road is a one way and if it has two, three, four, or five lanes. Depending on the direction and the number of lanes of the road the node is placed on the road differently. Using this information the distances to each lane marking were calculated using trigonometry.

The distance to the cars in the left, middle, and right lanes were calculated by looking for cars that were in the direction of the next node in front of the camera. This assured that the cars were in front of the camera. To determine whether the cars were in either the left, middle, or right lanes the same calculations were used for determining the lanes markings and made sure that the cars were enclosed within those same lane markings.

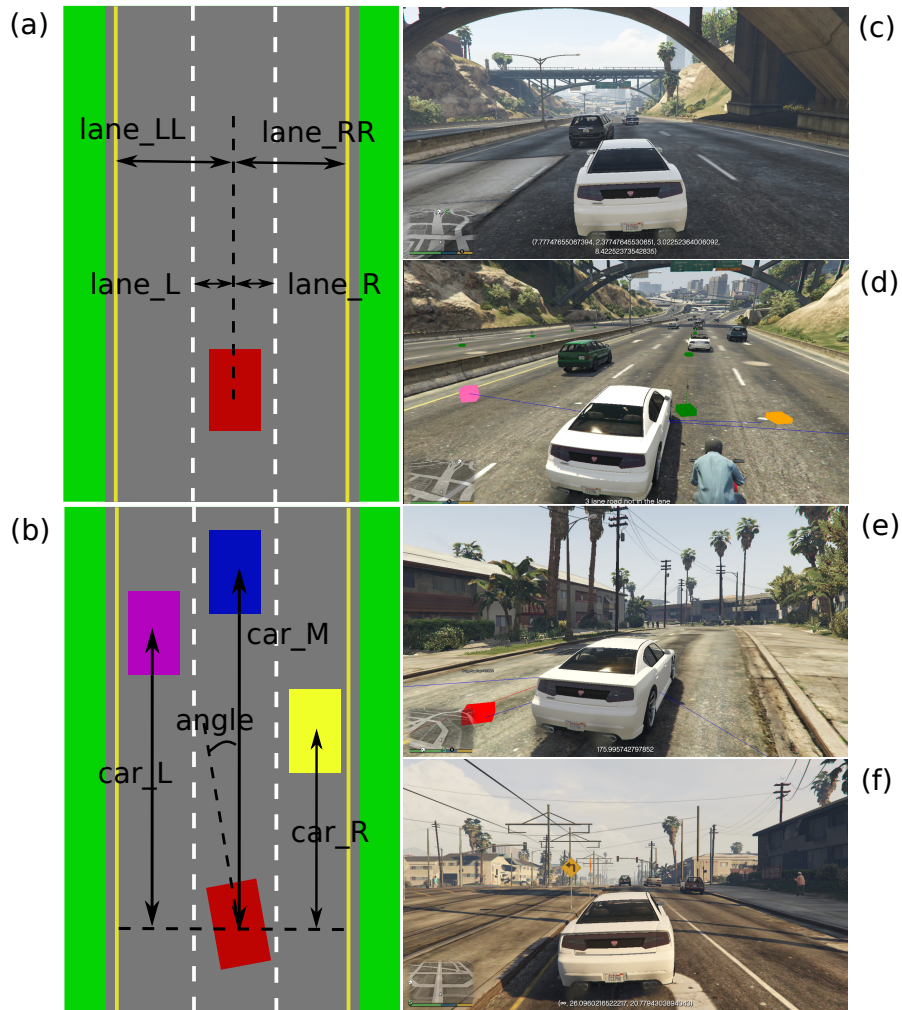


Figure 3.2: (a) A schematic showing 4 affordance variables (lane_LL, lane_L, lane_R, lane_RR). (b) A schematic showing the other 4 affordance variables (angle, car_L, car_M, car_R). (c) An in-game screenshot showing lane_LL, lane_L, lane_R and lane_RR. (d) An in-game screenshot showing the detection of the number of lanes in a road. (e) An in-game screenshot showing angle. (f) An in-game screenshot showing car_L, car_M and car_R.

3.0.5 Diversity of Scenarios

GTA-V offers a variety of different scenes that can be used for testing. The city of Los Santos, the in game environment, is mostly an urban environment but also includes suburban and rural environments.



Figure 3.3: Shows different times of the day and different geographic sceneries at which the screenshots are taken

Additionally GTA-V has a slowly changing time of day that runs 30 times faster than normal time. While collecting data the time of day was not specified and as such collected an evenly distributed amount of data from each in game time. Another powerful asset that GTA-V provides is many realistic weather scenarios. At this time different weather patterns were not included in the data and only used what the game describes as "Extra Sunlight".

Chapter 4

GTA Results

After the driving simulation in GTA-V had been running for 40 hours, over 500,000 screenshots were taken. After the data cleaning process to remove erroneous screenshots, approximately 483,000 images were left. 350,000 images were separated for training the CNN. The rest were kept for testing and validating purposes.

The CNN was then trained on a desktop computer running Ubuntu 14.04 equipped with an Nvidia Tesla K40. The training process ran for 21 epochs or approximately 36 hours before manually stopped after showing some sign of overfitting. The training went slightly more slowly than expected because of the bottleneck in reading images stored on a hard disk. The speedup could be achieved by storing the training images in a different database format that requires a shorter access time.

The final training loss per sample per variable is 0.151 whereas the final validation loss per sample per variable is 0.314. The loss is defined as mean squared error of 8 affordance variables combined. As mentioned earlier, the loss is expectedly high because the values of inactive affordances were set to be above the output range of the CNN. To display the loss in more detail, TABLE 4.1 shows the mean squared error or the loss of the validation set at various training epochs on each affordance variable separately. The losses shown in TABLE 4.1 also more accurately represent the true error of the system since they exclude errors that incur from the inactive affordance variables. From TABLE 4.1, it is important to note that the losses of all variables decrease as the training

progresses. However, the losses of some variables, namely lane_L, lane_M and lane_R, increase after a certain epoch of training, potentially due to overfitting. Regardless, the CNN trained at epoch 21 is chosen as the final model for its overall accuracy.

Table 4.1: Mean squared error of the validation set on each affordance variable at different epochs of training

Epoch	angle	car_L	car_M	car_R	lane_LL	lane_L	lane_R	lane_RR
6	0.008	0.441	0.618	0.138	0.114	0.030	0.012	0.064
11	0.010	0.323	0.518	0.115	0.043	0.023	0.009	0.020
16	0.009	0.322	0.491	0.109	0.038	0.024	0.014	0.018
21	0.007	0.309	0.489	0.108	0.036	0.025	0.016	0.015

Figure 4.1 also shows some images from the test set, their corresponding ground truth and the predicted label the final model outputs. Generally, as TABLE 4.1 suggests, the lane distance estimation is more precise than that of the car distance. Nevertheless, when the surrounding cars are closer, the error is arguably small and should not negatively affect the decisions made by the controller. Based on our observation, the large error of car_M often arises from cars on the left and the right lane at a further distance which are difficult to distinguish.

The RNN model was trained on the ORFE Clusters and were run on a Linux 2.23.2 system and a Tesla P100-PCIE-16GB. The RNN model did not converge unfortunately. The model




(a)		<table border="0"> <thead> <tr> <th></th> <th>Ground truth</th> <th>Predicted label</th> </tr> </thead> <tbody> <tr> <td>angle:</td> <td>- 0.679</td> <td>- 5.884</td> </tr> <tr> <td>car_L:</td> <td>na</td> <td>na</td> </tr> <tr> <td>car_M:</td> <td>8.758</td> <td>7.437</td> </tr> <tr> <td>car_R:</td> <td>na</td> <td>na</td> </tr> <tr> <td>lane_LL:</td> <td>na</td> <td>na</td> </tr> <tr> <td>lane_L:</td> <td>3.291</td> <td>2.935</td> </tr> <tr> <td>lane_R:</td> <td>2.108</td> <td>2.295</td> </tr> <tr> <td>lane_RR:</td> <td>na</td> <td>na</td> </tr> </tbody> </table>		Ground truth	Predicted label	angle:	- 0.679	- 5.884	car_L:	na	na	car_M:	8.758	7.437	car_R:	na	na	lane_LL:	na	na	lane_L:	3.291	2.935	lane_R:	2.108	2.295	lane_RR:	na	na
	Ground truth	Predicted label																											
angle:	- 0.679	- 5.884																											
car_L:	na	na																											
car_M:	8.758	7.437																											
car_R:	na	na																											
lane_LL:	na	na																											
lane_L:	3.291	2.935																											
lane_R:	2.108	2.295																											
lane_RR:	na	na																											
(b)		<table border="0"> <thead> <tr> <th></th> <th>Ground truth</th> <th>Predicted label</th> </tr> </thead> <tbody> <tr> <td>angle:</td> <td>- 0.051</td> <td>- 0.010</td> </tr> <tr> <td>car_L:</td> <td>na</td> <td>na</td> </tr> <tr> <td>car_M:</td> <td>na</td> <td>na</td> </tr> <tr> <td>car_R:</td> <td>na</td> <td>na</td> </tr> <tr> <td>lane_LL:</td> <td>9.064</td> <td>8.966</td> </tr> <tr> <td>lane_L:</td> <td>3.114</td> <td>3.566</td> </tr> <tr> <td>lane_R:</td> <td>2.082</td> <td>1.833</td> </tr> <tr> <td>lane_RR:</td> <td>na</td> <td>na</td> </tr> </tbody> </table>		Ground truth	Predicted label	angle:	- 0.051	- 0.010	car_L:	na	na	car_M:	na	na	car_R:	na	na	lane_LL:	9.064	8.966	lane_L:	3.114	3.566	lane_R:	2.082	1.833	lane_RR:	na	na
	Ground truth	Predicted label																											
angle:	- 0.051	- 0.010																											
car_L:	na	na																											
car_M:	na	na																											
car_R:	na	na																											
lane_LL:	9.064	8.966																											
lane_L:	3.114	3.566																											
lane_R:	2.082	1.833																											
lane_RR:	na	na																											
(c)		<table border="0"> <thead> <tr> <th></th> <th>Ground truth</th> <th>Predicted label</th> </tr> </thead> <tbody> <tr> <td>angle:</td> <td>-0.078</td> <td>0.694</td> </tr> <tr> <td>car_L:</td> <td>na</td> <td>na</td> </tr> <tr> <td>car_M:</td> <td>3.983</td> <td>6.085</td> </tr> <tr> <td>car_R:</td> <td>na</td> <td>na</td> </tr> <tr> <td>lane_LL:</td> <td>na</td> <td>na</td> </tr> <tr> <td>lane_L:</td> <td>2.766</td> <td>3.196</td> </tr> <tr> <td>lane_R:</td> <td>2.460</td> <td>2.203</td> </tr> <tr> <td>lane_RR:</td> <td>na</td> <td>na</td> </tr> </tbody> </table>		Ground truth	Predicted label	angle:	-0.078	0.694	car_L:	na	na	car_M:	3.983	6.085	car_R:	na	na	lane_LL:	na	na	lane_L:	2.766	3.196	lane_R:	2.460	2.203	lane_RR:	na	na
	Ground truth	Predicted label																											
angle:	-0.078	0.694																											
car_L:	na	na																											
car_M:	3.983	6.085																											
car_R:	na	na																											
lane_LL:	na	na																											
lane_L:	2.766	3.196																											
lane_R:	2.460	2.203																											
lane_RR:	na	na																											

Figure 4.1: Sample images from the test set along with their respective ground truth and predicted labels (na = car/lane does not exist). Please note that angle is in degree and the rest of the variables are in meters.

Chapter 5

Discussion

The results show that the lowest mean squared errors came from the angle of the car to the road and the detection of the lane markings. I theorize that the car distance variables were less successful because the range of possible values and the objects that count as a vehicle is far more varied than lane markings. Lane markings are all in a very similar position in an image and have very little image variation whereas a vehicle could be anything from a motorbike with a pedestrian on it to a freight truck.

As the network trained, the error for car distance continued to decrease with more training while the lane marking estimation started to overfit. A possible solution would be to have two separate neural networks running, one to estimate the car distances and one to estimate the lane marking distance along with the road angle. The disadvantage of this solution is that it requires extra computational power. Another solution might be to use a weighted loss. More precisely, a larger weight can be put on losses incurred from the car distances and a smaller weight on those from the lane marking distances. Doing so could help the car and the lane marking distance estimations converge at a more similar rate.

The error of the car distance estimation is observed to be larger with the cars further away and the cars driving on the left and the right lane. In addition, the error of night drives, in general, tends to be larger than that of drives happen in the day. Certainly, through a longer training, a more extensive training set and a deeper architecture,

the network's performance can be significantly improved and thus, the error would be minimized. Nevertheless, it is believed that such properties of the CNN pertain to an inevitable downfall of a system that solely relies on visual inputs. For humans and machines alike, a poor weather and lighting condition as well as a far distance could impair a distance estimation.

Regardless, distances of cars further away and in a different lane affect the controller's decisions in a lower degree compared to those of cars that are closer and in the same lane. Generally, a driving controller maintains the distance to the car directly in front by adjusting the speed accordingly. [4] When a lane change is required, only cars that are close in distance affect the decisions made by the controller. Consequently, the errors in distances of vehicles further, though indeed undesirable, are much more forgivable.

Chapter 6

Conclusion

I examined GTA-V as a model to test, train and enhance deep learning in self driving car research. I found that GTA-V allows researchers to create, train, and test on photo-realistic data to accurately estimate from the driver's perspective the distance to lane markings, distance to cars, and angle of driving. The best results of the CNN came from estimating the driving angle and the lane markings. The car distance estimation could be improved with more training time, but was terminated before it plateaued because the lane marking detection had started to overfit.

Bibliography

- [1] Rage plugin hook. <https://ragepluginhook.net/>.
- [2] Mohamed Aly. Real time detection of lane markers in urban streets. *CoRR*, abs/1411.7113, 2014.
- [3] Wonmin Byeon, Thomas M. Breuel, Federico Raue, and Marcus Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, pages 3547–3555. IEEE Computer Society, 2015.
- [4] Chenyi Chen, Ari Seff, Alain L. Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. *CoRR*, abs/1505.00256, 2015.
- [5] Santiago Fernandez, Alex Graves, and Jurgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *Proceedings of the 17th International Conference on Artificial Neural Networks, ICANN 07*, pages 220–229, Berlin, Heidelberg, 2007. Springer-Verlag.
- [6] Artur Filipowicz, Jeremiah Liu, and Alain Kornhauser. Learning to recognize distance to stop signs using the virtual world of grand theft auto 5.
- [7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, June 2012.

- [8] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, Fernando Mujica, Adam Coates, and Andrew Y. Ng. An empirical evaluation of deep learning on highway driving. *CoRR*, abs/1504.01716, 2015.
- [9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [10] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L. Cun. Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, P. B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 739–746. MIT Press, 2006.
- [11] Dean A. Pomerleau. Alvin: An autonomous land vehicle in a neural network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 305–313. Morgan-Kaufmann, 1989.
- [12] Dean A. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.
- [13] Geoffrey R. Taylor, Andrew J. Chosak, and Paul C. Brewer. Ovvv: Using virtual worlds to design and evaluate surveillance systems. pages 1 – 8, 07 2007.
- [14] Jurgen Schmidhuber. Habilitation thesis: System modeling and optimization. In *System modeling and optimization*, page 150. Springer, 1983.
- [15] R. Schmidt, H. Weisser, P. Schulenberg, and H. Goellinger. Autonomous driving on vehicle test tracks: overview, implementation and results. In *Proceedings of the*

IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511), pages 152–155, 2000.

- [16] Alireza Shafaei, James J. Little, and Mark Schmidt. Play and learn: Using video games to train computer vision models. *CoRR*, abs/1608.01745, 2016.
- [17] Tom Vanderbilt. Autonomous cars through the ages, Feb 2012.