DeepFollowing: Vision-Based Distance Estimation on Synthetically-Generated Driving Video using 3D Convolution

Author: Nayan Bhat

Supervisor: Dr. Alain Kornhauser

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering
Department of Operations Research and Financial Engineering
Princeton University

June 2017
I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

___________________________
Nayan Bhat

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

___________________________
Nayan Bhat
To my little brother, Milan.
Acknowledgments

I am incredibly grateful to all of the people who have helped me during this process.

Professor Kornhauser - since encouraging me to take ORF 245 my freshman fall you have shaped my Princeton experience in many ways. Thank you for your continued support during these past four years. Your sharp insights and insistence on “doing it for real” have elevated my work.

Artur - your helpfulness throughout this year is much appreciated. I literally could not have completed my thesis without your ScriptHook assistance or the GPU you gave me. Thank you for being an excellent resource and friend.

Dan Pease, Nick Yang, Ben Hicks, and the rest of the OIT / PICSciE team - thank you for going out of your way to facilitate this project’s technical needs.

Princeton SEAS and NVIDIA Corporation - thank you for your generous grants that allowed my thesis to come to fruition.

Amma, Appa, and Milan - thank you for all of your support, both tangible and intangible. From helping me build my computer to taking me around capturing driving video, you have made every effort possible to help me realize my aspirations.

Patton 408, APES, TI - thank you for being a much needed respite from this thesis and for giving me some incredible experiences I will cherish dearly.
Abstract

The recent surge in popularity of autonomous driving vehicles among the public, coupled with real roll-outs of self-driving systems, supports continued academic research in all aspects of this space. Modeling the relationship between the driving vehicle and a vehicle in front of it, known as vehicle-following, is a necessary consideration for any autonomous control system. Beyond valuable applications, such as reducing energy costs in the $700B trucking industry (ATA 2016), studying this highly sequential process provides interesting insight into the temporal structure of driving recognition.

A simple vehicle-following system can be reduced to the problem of accurately gauging the distance to a leading vehicle (DTLV). The most efficient vehicle-following system would allow vehicles to stream information between one another and coordinate decisions; however, the likelihood of two randomly-paired vehicles possessing the same notification system is low, at least in the foreseeable future. While lidar and radar systems have been popular solutions, particularly in Adaptive Cruise Control applications, they are expensive and not easily scalable. A promising alternative is the application of vision-based distance estimation to the car-following problem.

Producing an annotated data set of real vehicle-following footage is laborious and error prone, so realistic images are captured using the virtual driving environment in Grand Theft Auto V (GTA V). The advantage of such an environment is that driving conditions such as weather, road type, and leading vehicle model can be easily varied. Furthermore, true distances are known and do not have to be inferred.

This thesis finds that a novel 3D Convolutional Neural Network (3D CNN) estimates DTLV with mean average error of 4.13m on synthetic test data, a 6.8% improvement over a comparable 2D CNN. This performance improvement appears to be consistent across nearly all studied virtual driving conditions. Furthermore, training on the synthetic data provides qualitatively reasonable estimation on real-world driving data.
## Contents

1 Introduction .................................................. 1  
1.1 Overview: Autonomous Driving ............................ 1  
1.2 Thesis Outline .............................................. 2  
1.3 Contributions .............................................. 2

2 Vehicle-Following Theory and Control ................. 3  
2.1 Simplified Decision Model ................................ 3  
2.2 Classical Vehicle-Following Models .................... 5  
2.3 String Stability ........................................... 6  
2.4 Direct Perception .......................................... 7

3 Vision-Based Distance Estimation ....................... 9  
3.1 Motivations ............................................... 9  
3.2 Problem Statement ....................................... 10  
3.3 Convolutional Neural Networks ......................... 10  
3.4 3D Convolution ........................................... 13  
3.5 Classification vs Regression ............................. 14

4 Data .......................................................... 16  
4.1 Existing Datasets ......................................... 16  
4.2 Virtual Environment ...................................... 17  
4.3 Domain Adaptation ....................................... 19  
4.4 Data Generation ......................................... 21  
4.5 Data Visualization ....................................... 27

5 Procedure .................................................... 32  
5.1 Data Preparation ......................................... 32  
5.2 Model Training ........................................... 32  
5.3 Model Testing ............................................ 36
6 Results and Discussion

6.1 Distance Estimation ........................................... 39
6.2 Percentage Error .............................................. 41
6.3 Effects of Driving Environment Variation ....................... 44
6.4 Transfer Learning .............................................. 47
6.5 Open Road Driving ............................................. 49
6.6 Real World Driving ............................................. 50

7 Conclusion

7.1 Further Discussion ............................................ 53
7.2 Limitations and Future Research ................................. 55
1 Introduction

1.1 Overview: Autonomous Driving

Intelligent transportation technology has made tremendous advancement since its initial roots in Tokyo’s 1967 exploration of traffic control systems (Miles 2014). The current focus of research has shifted away from road network intelligence toward embedding technology within cars themselves. In 2017, there is perhaps no technology more eagerly awaited than autonomous vehicles, an excitement bred from expected improvements in safety, efficiency, and access.

Safety: Automotive crashes in the U.S. alone resulted in 2.44 million injuries and claimed 35,092 lives in 2015, a year-over-year increase of 4.3% and 7.2% respectively (NHTSA Aug 2015). According to a Department of Transportation study, 94% of crashes are attributable to human error, such as speeding or driving while impaired (NHTSA Feb 2015). By reducing and eventually eliminating the human element of driving, autonomous technology should bring about reductions in crashes and fatalities.

Efficiency: The reduction in accidents should also reduce highway clog and therefore travel time. A 2012 study of British parking policy found that cars are only driven an average of 4% of the time (Bates and Leibling 2012). Autonomous vehicles have the capability to remain active for the remaining 96% of time, particularly through shared car-ownership and ride-hailing models. By paring down the number of cars needed to serve a population, traffic and travel time should reduce, and raw materials used in production will be more efficiently consumed.

Access: Just 68.5% of the U.S. population is licensed to drive; the advent of autonomous driving technology will be welcomed by the blind, elderly, underaged and other groups with limited access to transportation (Federal Highway Administration).

Despite the flood of capital into this area of research, there remain many obstacles to a full implementation of autonomous driving. On the technical side, one of the most challenging problems is designing an artificial intelligence system that can successfully navigate difficult conditions it has never before experienced. Simultaneously,
there has been a focus on extreme accuracy in perception and control units, because accidents cause bad publicity and ultimately endanger human lives. Finally, given the commercialization of autonomous driving technology, there is a relative lack of data with which to study these problems in an academic context.

This thesis attempts to explore these outstanding technical issues through the lens of one objective - how well can we estimate the distance to the vehicle in front of us?

1.2 Thesis Outline

Chapter 2 provides context on the role of distance estimation in intelligent longitudinal control systems through a survey of Vehicle Following Theory. Chapter 3 details the motivations for estimating this distance using a vision-based cognition system and explains the theory behind the state of the art Convolutional Neural Network (CNN) algorithms used to do so. Chapter 4 provides a comprehensive examination of the data set generated for this thesis. Chapter 5 provides details regarding the structure and training of two CNN models - a traditional 2D CNN and a less-studied 3D CNN model. Chapter 6 discusses the performances of the models on both synthetic and real data. Chapter 7 qualifies the results and outlines steps for further research.

1.3 Contributions

This thesis aims to make several contributions to the fields of autonomous driving and machine learning. First, this project extends the current academic literature on simulated driving by studying a relatively untapped but feature-rich virtual driving environment - Grand Theft Auto V (GTAV). The details provided in Chapter 4 on the GTAV data collection process should be helpful to future researchers looking to access this resource. From a machine learning perspective, this thesis studies 3D CNNs, a rather uninvestigated neural network structure with the interesting capacity to learn time-dependent features. The empirical study of a 3D CNN’s performance relative to a comparable 2D version will be a useful data point for more comprehensive studies on optimal model structure.
2 Vehicle-Following Theory and Control

The mathematical modeling of driving has been a research focus for several decades, dating back to early single lane models proposed in the 1950s (Pipes 1953). Indeed, many elements of traffic theory, which studies driving systems at both a microscopic and macroscopic level, have been utilized in autonomous control architecture (Gazis 2002).

Traffic theory categorizes the highly complex process of driving into two major components: vehicle-following and lane-changing (Panwei and Dia 2005). Vehicle-following covers all longitudinal decisions, specifically acceleration and deceleration, while lane-changing focuses on lateral decisions such as steering angle. In practice these two components are interconnected - oftentimes one must decide between changing lanes and deceleration; however, they can be still be modeled as independent systems linked by a separate control system. This thesis focuses exclusively on the vehicle-following component, leaving lane-changing for future research.

2.1 Simplified Decision Model

Consider a single-lane highway consisting of two cars driving in a line without any external factors such as curves, slopes, or obstacles. Denote the leading car as $V_L$ and the following car as $V_F$. If $V_F$ intends to follow $V_L$, there are three possible situations that can occur:

1. $V_L$ is driving faster than $V_F$, in which case $V_F$ must eventually accelerate.

2. $V_L$ is driving slower than $V_F$, in which case $V_F$ must eventually decelerate, what Kohler (1979) calls the approaching range.

3. $V_L$ and $V_F$ are driving at the same speed, in which case the separating distance remains constant.

Accordingly, at every time step $t$, each car $k$ must make a decision on whether to accelerate, brake, or maintain speed. This decision $D_{t,k}$ can be represented as:
Figure 2.1: Vehicle-Following Cases. The separating distance increases (top), decreases (middle) or remains constant (bottom) at each time step.

\[ D_{t,k} = \alpha_{t,k} - \beta_{t,k} \]  \hspace{1cm} (2.1)

subject to:

\[ \alpha_{t,k}, \beta_{t,k} \geq 0 \]  \hspace{1cm} (2.2)

\[ \alpha_{t,k} \leq A_{t,k}, \beta_{t,k} \leq B_{t,k} \]  \hspace{1cm} (2.3)

\[ \min(\alpha_{t,k},\beta_{t,k}) = 0 \]  \hspace{1cm} (2.4)

where \( \alpha_{t,k} \) is the chosen acceleration, \( \beta_{t,k} \) is the chosen deceleration, \( A_{t,k} \) is the maximum acceleration possible at time \( t \) for car \( k \), and \( B_{t,k} \) is the maximum brake possible. Note that constraint 2.4 implies either an acceleration or deceleration decision can be made at each time step but not both simultaneously. For a simple control system, \( \frac{\alpha_{t,k}}{A_{t,k}} \) and \( \frac{\beta_{t,k}}{B_{t,k}} \) represent the fraction of gas and brake pedal compression respectively.

From the following vehicle’s perspective, \( D_{t,L} \) can be taken as exogenous, reducing the model to a single decision \( D_{t,F} \) at each time step. Assuming negligible reaction time, a simple decision strategy would be for \( V_F \) to select \( D_{t,F} = D_{t,L} \). This is a poor strategy in practice, however, because reaction times are not negligible and \( D_{t,L} \) is not precisely known to \( V_F \) unless the cars share an information stream. Furthermore,
if $V_L$ is making irrational or illegal decisions, such as driving much higher than the speed limit, $V_F$ may not desire to mimic such actions.

### 2.2 Classical Vehicle-Following Models

Vehicle-following models attempt to provide more comprehensive strategies for selecting $D_{t,F}$. Brackstone and McDonald (1999) have re-examined several major classes of vehicle-following models, the two most important of which are briefly explored below. As will be seen in Eqs. 2.5 and 2.7, the selection of $D_{t,F}$ depends on $\Delta x_{t-T,F}$, the relative separating distance referred to in this paper as the Distance-To-Leading-Vehicle (DTLV). Many applications build off these models, including adaptive cruise control (Liang and Peng 2000), collision avoidance systems (Vahidi and Eskandarian 2003), and platooning systems (Santini 2015, Saeedna and Menendez 2017).

#### 2.2.1 Gazis-Herman-Rothery (GHR) Model

The Gazis-Herman-Rothery (1961) model is the most popular vehicle-following model, attempting to map relative velocity to an acceleration or deceleration decision:

$$D_{t,F} = \lambda \Delta v_{t-T,F}$$

where $\Delta v_{t,F}$ is the relative speed of the following vehicle at time $t$ and $T$ represents the driver’s reaction time. $\lambda$ is a sensitivity with general form:

$$\lambda = \alpha \left( \frac{\nu_{t,F}}{\Delta x_{t-T,F}} \right)^m$$

where $\nu_{t+T,F}$ is the speed, $\Delta x_{t-T,F}$ is the relative position of the following vehicle, and $\alpha$ is some linear coefficient. $m$ and $l$ are tunable parameters which have been the focus of much study. Based on a meta-analysis of 13 studies in Brackstone and McDonald (2000), optimal parameter combinations from experimental data have been found within the ranges $[-0.8, 2.7]$ and $(0, 3)$ respectively. While the inconsistency
in optimal parameters across studies, likely due to differences in observed traffic flow and limited data, has cast some doubt on this approach, new research using this model such as Gunawan et al. (2016) continues to be published.

2.2.2 Collision Avoidance (CA) Models

Unlike the GHR model which focuses on relative velocity, Collision Avoidance models define a separating distance threshold that is large enough to prevent a crash in the event that $V_L$ applies its brakes unexpectedly. Originally derived by Kometani and Sasaki (1959), CA models have the following form described by Brackstone and McDonald (2000):

$$\Delta x_t = c_1 v_{L,t}^2 + c_2 v_{F,t+T}^2 + c_3 v_{F,t+T} + b_0$$

(2.7)

where $v_{L,t}$ is the speed of $V_L$ at time $t$ and $v_{F,t+T}$ is the speed of $V_F$ at time $t + T$ adjusted for the reaction time. $c_1$, $c_2$, $c_3$, and $b_0$ are all coefficients that are situation specific. Gipps (1981) related the $c_1$ coefficient to the expected maximum braking rates of $V_L$ and $V_F$, which allows for easier interpretation of the model. In such a model, computing an optimal separating distance is only useful if one can gauge the existing separating distance, providing a control system rationale for this paper’s focus. Also of interest is the relationship between driving speed and DTLV. Under normal driving situations, one would expect higher speeds to mean that drivers keep a higher separating distance. However, this relationship may introduce a bias into driving data that is discussed in Chapter 6.

2.3 String Stability

Extending the simple decision model provided in 2.1 to a larger vehicle-following system leads to the issue of string stability, which governs the propagation of error through a vehicle platoon. Simply put, if a line of vehicles is not string stable, then a small range error at the beginning of the line may compound as it flows toward the
end (Liang and Peng 1999).

Seiler et al. (2004) demonstrated that a vehicle-following policy solely fixated on maintaining a constant DTLV necessarily leads to string instability. They note that one way to ensure stability while using a constant separating distance is to consider distance to both the relative leading car and the absolute leader of the platoon. Sheikholeslam and Desoer (1993) found that communication links between vehicles can also improve string stability for a constant DTLV policy. Swaroop (1997) found that a constant headway time policy, in which DTLV is set proportionally to speed, can add robustness to the string.

These results show that DTLV cannot be the only consideration in a truly functional vehicle-following strategy. This limitation is discussed further in Chapter 7.

2.4 Direct Perception

An essential design decision for an autonomous driving system is choosing the relationship between cognition, which interprets raw sensor data, and control, which makes physical decisions such as turning the steering wheel. Chen (2016) delineates two popular approaches, Mediated Perception and Behavior Reflex, and proposes a third, Direct Perception. Mediated Perception separates the cognition and control units entirely, first constructing a full representation of the driving scene and then making decisions accordingly. Behavior Reflex, on the other hand, uses a single machine learning algorithm to perform both cognition and control, mapping raw input data directly to a physical decision.

Chen’s Direct Perception approach takes a middle ground by estimating fourteen essential affordance indicators which are then used to make control decisions. This method avoids the overcomplications of Mediated Perception while extracting more interpretable information than Behavior Reflex. Direct Perception was shown to outperform these other two methods in Chen’s study.

Distance estimations comprise five of Chen’s DeepDriving affordance indicators. In a study of the risk residuals of these affordance indicators, Filipowicz (2016a) finds that distance error was the leading source of error in 94% of images. This highlights
the importance of further research into distance estimation, which is the focus of the remainder of this thesis.
3 Vision-Based Distance Estimation

3.1 Motivations

As the discussion in Chapter 2 demonstrates, DTLV is a key element of any vehicle following or longitudinal control system, as well as an important component of Chen’s Direct Perception approach. The rest of this paper is related to the prediction of this distance using visual driving information.

There are four major technologies for distance detection: Lidar, Radar, Sonar, and Optical. The first three emit signals - lasers, radio waves, and sound waves respectively - and triangulate distance to an object by the time it takes the reflected pulse to return. Lidar has the best accuracy, detecting objects up to 100m away with just 5cm error (Ackerman 2016); however, it tends to perform worse under adverse weather conditions such as rain or snow. Radar is not usually affected by such conditions and has range up to 150m, but tends to be expensive like Lidar (Sun, Bebis, and Miller 2006). Sonar, on the other hand, is effective only at short ranges, typically under 2.5m (Hikita 2010).

Cameras tend to have a shorter detection range, can be negatively affected by weather conditions, and are ultimately passive sensors. Despite this, optical distance detection is worth studying for numerous reasons. The most obvious is cost - compared to $7,500 Lidar systems (Naughton and Bergen 2017), cameras are cheap and easily available. Furthermore, certain driving tasks, such as traffic sign recognition and lane detection (Bertozzi and Broggi 1998), necessitate the use of optical sensors; visual distance estimation is a natural next step that would not introduce compatibility or installation issues into an existing driving system. Most compelling, however, is the simple argument that as humans, we drive only using our eyes. While other technologies may provide redundancy or reduce error, our goal should be to develop a system that mimics our own driving.
3.2 Problem Statement

Suppose vehicle $V_F$ follows vehicle $V_L$ from time 0 to time $T$. Let $X_t \in \mathbb{R}^4$ be the tensor representing the sequence of images observed by $V_F$’s front-facing vehicle camera from time 0 through time $t$. Let $y_t \in \mathbb{R}$ be the separating distance between $V_L$ and $V_F$ at time $t$. Then the vision-based distance estimation problem is defined as finding a function $f$ that minimizes:

$$\min_f \sum_{t=0}^{T} \left| f(X_t) - y_t \right|$$  \hspace{1cm} (3.1)

While there exists some true mapping of $X_t$ to $y_t$, such an $f$ is highly nonconvex with intractably large input size. A standard 1080p video taken on an iPhone for just one minute, for example, yields over 11 billion pixel data points. As finding a global optimal solution is an impractical exercise, much of the academic focus has turned to machine learning algorithms that can efficiently discover acceptable local minima.

3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a subclass of machine learning algorithms that have performed empirically well in vision-based prediction when given a large training data set. While this thesis does not aim to provide a comprehensive coverage of neural network theory, certain elements are included to provide context for why 3D CNNs, outlined in section 3.4, are particularly interesting to study. Please note that the remaining part of this chapter relies heavily on Seung (2017) and retains his notation style.

3.3.1 Simple Feedforward Networks

The CNNs in this paper are structured as feedforward networks, which are designed to repeatedly perform linear operations on inputs followed by nonlinear activations.
An artificial neural network is feedforward if it satisfies the relationships outlined in Eq. 3.2:

\[ x_i := f(\sum_{j=1}^{N} W_{ij} x_j + b_i) \quad \text{for } i = 1 : N \]  
\[ W_{ij} = 0 \quad \text{for } i \leq j \]  

where \( x_i \) is the value of node \( i \), \( N \) is the total number of nodes, \( W_{ij} \) is the weight of the connection from node \( j \) to node \( i \), \( b_i \) is the bias of node \( i \), and \( f \) is some nonlinear function.

Feedforward networks are typically organized in layers, the most basic structure being the multilayer perceptron (MLP) defined as:

\[ x^l_i := f(\sum_{j}^{n_{l-1}} W^l_{ij} x^{l-1}_j + b^l_i) \quad \text{for } l = 2 : L; \ i = 1 : n_l \]  

where \( x^l_i \) is \( i \)th node in layer \( l \), \( n_l \) is the number of nodes in layer \( l \), \( L \) is the number of layers, and \( W^l_{ij} \) is the weight of the connection from node \( j \) in layer \( l-1 \) to node \( i \) in layer \( l \). The nodes in each layer connect to every node in the subsequent layer. This layered structure enables deep learning, as weight parameters in the later layers learn to interpret weight parameters in preceding layers rather than the underlying data. Layered learning is quite powerful - an MLP with just two stacked layers can compute any boolean function, albeit with possibly exponential number of conjunctions (Anthony 2003).

### 3.3.2 2D Convolution

The simplicity of the MLP comes at the price of slow computation, as \( W \) scales quadratically with \( n_l \). Given the large input size in computer vision applications, pure MLPs will either be painfully slow to optimize or too small to perform well.
LeCun (1998) found that a more intelligent mapping of nodes, known as convolution, would shrink the parameter space while providing useful translation-invariant feature recognition. Rather than linking all nodes together, convolution only connects nodes within a nearby spatial vicinity and shares weights across regions.

![Figure 3.1: Reduction of Parameters](image)

Convolution regularizes the parameter space through local connectivity (transition 1) and weight sharing (transition 2). The number of weights in the first layer has been reduced from 64 to 3. Adapted from Gwardys (2016).

Convolution in one dimension is mathematically defined as the integral in Eq. 3.4, where \( f \) and \( g \) are any functions of \( z \). Eq. 3.5 extends convolution to two-dimensions as applied to the discrete case of CNNs, where \( f \) is replaced by \( s \), the input vector or feature map, and \( g \) is replaced by \( w \), the constrained weight vector known as the kernel or filter. Unless otherwise specified, “valid” convolution with stride length of 1 is used.

\[
(f * g)(z) = \int_{-\infty}^{\infty} f(z) \ast g(a - z)da \tag{3.4}
\]

\[
(w \ast s)(i_1, i_2) = \sum_{j_1, j_2} s_{j_1, j_2} \ast w_{i_1 - j_1, i_2 - j_2} \tag{3.5}
\]

A convolutional layer has a defined number of input feature maps and kernels. Each output feature map is the sum of the convolutions of each input feature map.
with one kernel. Output feature map $I^{\alpha} \in \mathbb{R}^2$ is defined as:

$$ I^{\alpha} = f(\sum_{\beta} w^{\alpha\beta} \ast I^{\beta} + b^{\alpha}) \tag{3.6} $$

where $w^{\alpha\beta}$ is the 2D slice of kernel $\alpha$'s constrained weight matrix corresponding to input feature map $\beta$, and $b^{\alpha}$ is the bias for feature map $\alpha$.

### 3.4 3D Convolution

While 2D convolution has become the norm in image recognition and computer vision, higher dimensional convolution remains a relatively unexplored area. 3D convolution is a mathematically simple extension from its two-dimensional counterpart, as shown in Eq. 3.7. Feature maps are updated exactly as in Eq. 3.6, where $w^{\alpha\beta}$ instead represents a 3D slice of a 4D tensor.

$$ (w \ast s)(i_1, i_2, i_3) = \sum_{j_1, j_2, j_3} s_{j_1,j_2,j_3} \ast w_{i_1-j_1, i_2-j_2, i_3-j_3} \tag{3.7} $$
Much of the existing academic literature on 3D convolution focuses on data with three spatial dimensions, such as seismic cubes representing the Earth’s surface (Aqrawi 2009), spectral depths (Clifton 2016), and RGBD object detection (Maturana and Scherer 2015). Baccouche et al. (2011) introduced 3D convolution to video classification by treating time as the third dimension. This study, along with Ji et al. (2013), found that 3D CNNs produce competitive classification results in Action Recognition. Karpathy et al. (2014) found that 3D CNNs highly outperform feature-based classification algorithms but only moderately outperform 2D CNNs. Tran et al. (2015) found that 3D CNNs produce best-in-class results for action similarity labeling, scene classification, scene classification, and object recognition.

The major advantage of 3D CNNs is that temporal dependencies of image sequences can be preserved through multiple network layers, facilitating deep learning of features both within and across images. This approach, known as slow fusion, has empirically performed better than other time-augmented strategies (Karpathy et al. 2014). As human driving is ultimately a dynamic process that relies on memory and successive frames of reference, one should expect improved prediction from the incorporation of temporal elements.

### 3.5 Classification vs Regression

Although early neural network theory can be traced back to linear regression models from the 1800s, most of the academic focus has been on classification rather than regression, particularly within computer vision (Schmidhuber 2015). The reason for this is twofold: The current state of research) categorical labeling is typically easier and more accurate than continuous labeling; and 2) it is possible to imply distance and other continuous metrics from classified objects using bounding boxes and scaling factors (Ying and Yuhui 2010). The lack of publicly available computer vision data sets amenable to regression further hinders progress in this space. Chapter 4 discusses the use of a virtual simulator to accurately generate a large image-based data set tailored for regression.
Figure 3.3: Preservation of Temporal Features. This figure from Chen (2016) demonstrates how a 3D feature map is produced only when the kernel convolves across all three dimensions. In the middle case, known as 2.5D convolution, the 3D input and 3D kernel produce a 2D output, limiting the temporal learning capacity to just one layer.
4 Data

The endurance of Moore’s law has led to significant increases in computational capabilities, facilitating the current ubiquity of big data and machine learning algorithms. While advances in both hardware and software have enabled efficient large-scale data analysis, the collection of data itself remains a constraint in many applications.

Image labeling, in particular, continues to frustrate data scientists due to the largely manual annotation process. Programs such as Amazon’s Mechanical Turk have enabled some degree of scale but are expensive and not necessarily reliable. Automatic data labeling, such as active learning (Yan, Yang, and Hauptmann 2004) and segmentation-based labeling (Chen et al. 2014), has made strides in recent years but still requires human annotation on a subset of images. Furthermore, tradeoffs must be made between labeling accuracy and efficiency (Wigness, Draper, and Beveridge 2015). Introducing higher label noise into the training data set may be unacceptable for applications such as car driving, where human lives are at stake.

4.1 Existing Datasets

The competitive search for a commercial driverless platform has kept much autonomous driving data out of the public domain. Despite this, several useful car driving data sets are available, having been labeled either manually or via sensors such as radar or lidar.

The CBCL StreetScenes database contains 3.5K images with hand-labeled bounding boxes for cars, pedestrians, bicycles, buildings, trees, roads, skies, sidewalks, and stores (Bileschi 2006). The Ford Campus Vision and Lidar data set contains approximately 7K images and corresponding 3D point cloud of a Ford F-250 driving around downtown Dearborn, Michigan (Pandey, McBride, and Eustice 2011) The KITTI dataset contains 12K driving images labeled for 3D Object Detection (Geiger, Lenz, and Urtasun 2012). Udacity recently released 70 minutes of car driving footage in Mountain View under both sunny and overcast weather conditions. This includes 24K human and machine labeled images of cars, trucks, pedestrians, and in some
cases streetlights (Cameron 2016).

There are several issues with using these existing data sets for this thesis. First, the data sets are quite small - the largest has fewer than 25K labeled images. Second, as these data sets were not collected specifically for car-following, the relevant image sequences are an even smaller subset of data. Third, these data sets do not exhibit much variation in driving conditions, as they are typically taken during clear weather in one region. Furthermore, these data sets are labeled for classification; while DTLV may be extractable from the lidar data, this process is nontrivial and may induce labeling error.

4.2 Virtual Environment

Generating data through a virtual environment offers substantial advantages over real world data. Because the virtual universe is a closed system, ground truth labels can be computed easily and exactly, facilitating collection of data magnitudes larger than the existing real world data sets. Driving conditions, such as road type and weather, can be adjusted at-will. Furthermore, virtual worlds can simulate edge cases that may be difficult or dangerous to capture in real life. Of course, a model trained on synthetic data may not perform well on real data, which is a consideration discussed in Section 4.3.

For car-driving simulation, there are several open-source virtual environments from which to choose. The Open Racing Car Simulator (TORCS) is a popular option built specifically for research purposes; however, its focus on racing tracks, limited driving condition settings, and simple graphics make it less amenable for urban driving studies. The Udacity simulator offers higher customizability of road conditions and provides multiple simultaneous camera views (Brown and Cameron 2017). However, it currently lacks a significant amount of development and is also not particularly tailored to urban settings.

Grand Theft Auto V (GTAV) is the simulator of choice for this thesis as it is the most realistic and thoroughly developed, a byproduct of its commercial rather
Set in the city of Los Santos, a fictional representation of Los Angeles, GTAV provides a comprehensive suite of roads, vehicles, pedestrians, and other objects. As there is little formal documentation of GTAV, much of the information presented in this section is crowdsourced from GTAV community pages.

**Road Network** Based on user submitted lists (GTAV Street Names), there are at least 217 county roads and 17 highways that run through the GTAV virtual world. Roads are modeled using a navigation mesh comprised of nodes and links, providing guidance for the in-game driving AI. There are four types of nodes: Asphalt Only Road, Asphalt/Simple Path, Under the Map, and Water (GTAV Native DB). Only the first two are relevant for driving data collection.

**Car Models** GTAV offers 262 vehicle models, including passenger cars, SUVs, trucks, emergency and military vehicles, motorcycles, aircraft, watercraft, and bicycles. Each model has a vast set of permutations, ranging from primary color to whether it has been recently washed (GTAV Vehicle Models).

**Pedestrians** GTAV has over 536 pedestrian models, ranging from police officers to hairdressers (GTAV Pedestrians). Realistic wildlife such as deer and mountain lion are also available. One notable omission is children due to the graphically violent nature of the game.

**Weather and Time** GTAV has 14 distinct weather conditions and enables grad-

---

1Grand Theft Auto 5 is property of Rockstar Games, Inc. It is used in this thesis for strictly research and educational purposes.
The figure on the left is a map of the GTAV virtual world (from gta-V-map.com). The figure on the right highlights the road network on the map (from Fowley Jr.). As can be seen, GTAV includes a large urban center with sparse surroundings.

4.3 Domain Adaptation

Studies using data generated from virtual environments are useful for two reasons. First, they can provide insight into the nature of real-world car driving. Finding time dependencies in virtual data, for example, may suggest that time dependencies exist in real data. The more important use case, however, is that models trained on virtual data may be directly applicable to the real world. The viability of this second option is the subject of domain adaptation research.

Domain adaptation aims to understand cross-dataset generalization where source and target sets reference the same underlying data category but have different distributions (Gopalan et al. 2015). Much of the research in visual domain adaptation focuses on physically related images that vary for environmental reasons, such as shifts in lighting or angle (Patel, Gopalan, and Chellappa 2015). The relationship
<table>
<thead>
<tr>
<th>Weather</th>
<th>Code</th>
<th>Weather</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtraSunny</td>
<td>0</td>
<td>ThunderStorm</td>
<td>7</td>
</tr>
<tr>
<td>Clear</td>
<td>1</td>
<td>Clearing</td>
<td>8</td>
</tr>
<tr>
<td>Clouds</td>
<td>2</td>
<td>Neutral</td>
<td>9</td>
</tr>
<tr>
<td>Smog</td>
<td>3</td>
<td>Snowing</td>
<td>10</td>
</tr>
<tr>
<td>Foggy</td>
<td>4</td>
<td>Blizzard</td>
<td>11</td>
</tr>
<tr>
<td>Overcast</td>
<td>5</td>
<td>Snowlight</td>
<td>12</td>
</tr>
<tr>
<td>Raining</td>
<td>6</td>
<td>Christmas</td>
<td>13</td>
</tr>
</tbody>
</table>

Figure 4.3: Table of Weather Codes

Figure 4.4: GTAV driving under different lighting and weather conditions

between synthetic data and real-world data is more abstract and less studied. Nevertheless, the existing research in synthetic domain adaptation is promising. Busto et al. (2015) find that weakly supervised learning can achieve successful viewpoint refinement of real image labels using virtual 3D object data. Ganin and Lempitsky (2015) use backpropagation to align virtual and real domains for deep neural networks. Shrivastava et al. (2016) take the opposite approach, using real world data to generate synthetic images via adversarial training.

Deep neural networks, such as those implemented in this thesis, may be particularly amenable to domain adaptation. Yosinski et al. (2014) discovered that the
initial convolutional layers in AlexNet, a standard deep learning structure, learn to recognize general broadly applicable features, rather than specific ones tailored to the training data set. As a result, fine-tuning pre-trained CNNs has become a popular method to save training time; virtually every deep learning framework now offers pre-trained models off-the-shelf. Furthermore, Soekhoe et al. (2016) found that freezing the initial layers of AlexNet during training can yield better performance, particularly when the target data set is small.

While this thesis briefly explores domain adaptation from GTAV to real driving, such analysis is only done at a qualitative level due to lack of data. Further discussion on this topic can be found in Chapter 7.

4.4 Data Generation

This section outlines the data generation process within GTAV, as applied to the specific case of vehicle-following. Data collection consists of repeatedly generating sequences of a host vehicle driving behind a leading vehicle under randomized environmental conditions. The resultant fully-labeled data set of 1.3 million images is referred to in this paper as the Princeton Virtual Vehicle Following (PVVF-1.3M) database.

While the PVVF-1.3M data set is itself usable for future research, the real advantage of virtual simulation is the ability to produce tailored data sets for specific purposes at minimal cost. As data collection in GTAV is an involved operation with little existing formalized discourse (Filipowicz 2016b being the exception), this section provides specific procedural details to guide future collection for academic research.

4.4.1 Script Writing

GTAV is a closed source game, requiring behind-the-scenes modifications of the game environment in order to simulate driving and collect data. Modification scripts can be written in C# using Alexander Blade’s Script Hook V and Patrick Mours’s ScriptHookV.NET. Users have uncovered approximately 5,200 native functions to ma-
nipulate vehicles, drivers, roads, and other game elements, all of which are listed on the GTAV Native DB website. These functions can be called either directly through hash codes or through wrappers developed for commonly used purposes. A particularly useful class of functions relates to the in-game AI, enabling easily implementable directives such as driving to specific coordinates or wandering.

As GTAV was not built for academic research, many functions and inputs remain unknown. Furthermore, software assistance does not formally exist and updates to GTAV can affect script functionality. The large and active gaming community is a critical resource to uncover unknown game elements and debug code. Many helpful posts can be found on GTA Forums and GTA Wiki.

4.4.2 Initialization

For the data collection to work well, sequence starting conditions must be carefully selected to prevent the introduction of bias into the sample.

Let $P_{T_0}^L, P_{T_0}^F \in \mathbb{R}^3$ be the three dimensional position of the leading car $V_L$ and following car $V_F$ respectively at starting time $T_0$. Let $F_{T_0}^L, F_{T_0}^F \in \mathbb{R}^3$ be the unit vector describing the initial forward orientation of each car. Next, let $R^L, R^F$ be the set of
road points such that $V_L$ and $V_F$ are fully on the road at time $t$ if Eq. 4.1 holds. Let $A(R_i)$ be the function that gives the road orientation at road point $R_i$. Let $D_{max}$ be the maximum distance from the following car at which the leading car is visible. Note that this depends on weather conditions, time of day, size of leading car, and other factors. For simplification, this value is assumed to be 60m. Finally, let $W$ be the set of all other objects within a $D_{max}$ radius of $V_F$. Then the initial starting conditions are comprehensively defined as:

\[ P_{T_0}^L \in R_L^L, P_{T_0}^F \in R_F \]  \hspace{1cm} (4.1)

\[ ||P_{T_0}^L - P_{T_0}^F|| \leq D_{max} \]  \hspace{1cm} (4.2)

\[ F_{T_0}^L = A(P_{T_0}^L), F_{T_0}^F = A(P_{T_0}^F) \]  \hspace{1cm} (4.3)

\[ F_{T_0}^L \approx F_{T_0}^F \]  \hspace{1cm} (4.4)

\[ P_{T_0}^{W_i} \neq \lambda P_{T_0}^L + (1 - \lambda) P_{T_0}^F, \quad \lambda \in (0, 1) \]  \hspace{1cm} (4.5)

Eq. 4.1-4.5 ascertain that the vehicles are appropriately placed on roads, within following range, oriented in the legal direction of the road, oriented in approximately the same direction as each other, and within line of sight to begin the sequence. While Eq. 4.2-4.5 are straightforward, Eq. 4.1 is difficult to implement in GTAV, as the road network is mostly texture-based and therefore difficult to access. Accessing road information through nodes is not completely reliable because their native functions are not yet fully understood.

A more practical, albeit less clean, solution was to randomly select points within following range, spawn the cars at the road nodes closest to these points oriented along the legal road direction, and then perform initialization checks to make sure all conditions hold. Initialization for each sequence was repeated until all of the
Figure 4.6: Following Vehicle Model. The larger Seminole SUV model (right) was chosen to keep the camera position high above the ground. Users have likened the Seminole to the Jeep Grand Cherokee (left, from CarsForSale.com).

Initialization checks were passed.

Certain driving conditions and settings were also updated at initialization in line with those made by Filipowicz, Liu, and Kornhauser (2017). Time-of-day was pushed forward by one hour at the start of every sequence. Weather changes were made every ten sequences, as they appeared to be more computationally intensive to render. The leading car model was randomly selected from a manually curated list of 111 vehicles that included compact vehicles, sedans, pickup trucks, SUVs, sports cars, vans, pickup trucks, buses, and taxis. The following car model was not updated in order to keep camera view and physical driving attributes constant.

4.4.3 Image Collection and Annotation

After initialization, images were screenshotted and saved at approximately 20 frames-per-second (FPS) throughout each sequence. As the lowest GTAV graphics setting is 800x600, images were downsampled upon collection to 280x210, to match the format used by Chen (2016) and Filipowicz et al. (2017) to train their CNNs. Images were captured from a forward-oriented camera attached to the hood of the following vehicle and capture footage of the leading vehicle. The camera is repositioned every time-step to the front of the car, similar to the procedure described by Filipowicz (2016b) but adjusted for the larger following car model. 1.6 million raw images were collected before data cleaning was performed, taking approximately 22 hours. Each image can
Figure 4.7: Feasible Starting Positions on GTAV Map. Initial starting points were randomly selected within this subset grid. The closest road nodes to these starting points were the spawning coordinates.

be identified by its sequence and frame number, enabling easy manipulation during the analysis phase.

As GTAV is a closed-system, every object has known location coordinates and features at all times, allowing ground truths to be accurately and efficiently labeled. The data set generated for this analysis labels each image along 13 dimensions, which are described below.

Exit Code The first label describes the exit code for the image, described in the next section. Images that do not end a sequence are given an exit code of 0.

Image ID Labels 2-4 together provide a unique identification for each image. Label 2 refers to the session of data collection. The PVVF-1.3M data set was collected over four separate sessions of 400K images each, the first three used in training and the last split into validation and testing. Label 3 refers to the car-following sequence and label 4 refers to the frame within that sequence.

Separating Distance Label 5 refers to the distance between the host vehicle and the leading vehicle. There are many ways to calculate this distance. This data set
uses the physical distance rather than road distance as this is a more conservative measure.

**Speed** Label 6 refers to the speed of the host car in meters per second. The speeds are capped at 30 m/s which is approximately 67 mph.

**Location** Labels 7-9 refer to the X,Y,Z dimensions of the host car’s position.

**Road Type** Label 10 refers to whether the car is on a local road or a highway. As this label is not directly accessible in GTAV, a workaround solution was found. At each frame, the name of the road is collected and checked against the data set of highway names. As the list of highway names is small, this procedure does not add a noticeable lag to the data collection.

**Car Model** Label 11 refers to the car model of the leading car. Both cars and trucks were used. Only half of the possible vehicles were randomly selected, in order to leave room for future testing on vehicle models which the network has not been trained on.

**Weather and Time** Labels 12 and 13 refer to the weather and time at that frame. These are easily accessible in the game.

### 4.4.4 Exiting the Sequence

While an optimal collection process would generate a consistent number of frames per sequence, this is difficult in GTAV due to an imperfect AI driving system that occasionally gets stuck or veers off course. Such situations yield misleading data that could confuse the training algorithm. Steps have been taken to mitigate these effects by quickly identifying when the host car is stuck and ending the sequence. Exit codes are attached to each sequence referring to the specific check that was broken, allowing the post-collection data cleaning process to remove bad data. Sequences are automatically ended after 3000 frames, or 2.5 minutes at the 20 FPS collection rate.

### 4.4.5 Data Cleaning

While the data generation process involves internal checks for bad data, a subsequent data cleaning process is required to remove some retained errors. Due to the repeated
Initialization, rendering often slows during the collection process, so initial frames of each sequence are removed in order to account for rendering issues. Since the internal checks require several time steps to identify problem situations, the frames preceding sequence exits are also removed. The number of frames depends on the type of sequence exit involved. Furthermore, sequences that were too short (less than 50 frames), either during the collection process or after cleaning, were removed in order to facilitate time-related analysis. The entire data cleaning process removed approximately 300K images, leaving 1.3M images as usable. Future data collection improvements can likely reduce this error rate substantially.

4.5 Data Visualization

This section explores the data distribution within the PVVF-1.3M database across all labeled dimensions. There are exactly 1,314,720 labeled images within this data set representing 1,660 distinct driving sequences. Table 4.1 provides summary statistics for sequence length and Figure 4.8a shows the full distribution. As can be seen, longer sequence lengths tend to have lower occurrence within the data set, with the exception of the largest bucket which is the cutoff length. This suggests that events that force a sequence to exit, such as too large of a separating distance or one of the vehicles getting stuck, may be modeled as a random variable. Based on the Quantile-Quantile plot in Figure 4.8b, such a random variable appears to have a thinner tail than the exponential distribution.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Low</th>
<th>Median</th>
<th>Mean</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Count</td>
<td>50</td>
<td>517</td>
<td>793</td>
<td>2997</td>
</tr>
<tr>
<td>Duration (min)</td>
<td>0.04</td>
<td>0.43</td>
<td>0.66</td>
<td>2.50</td>
</tr>
</tbody>
</table>

Separating distance is one of the most important distributions, as this is the value regressed upon. Figure 4.9a demonstrates that the separating distance is not normally
Figure 4.8: **Sequence Length.** Figure 4.8a shows the distribution of sequence length as a histogram. Figure 4.8b plots sequence length quantiles against an exponential distribution.

distributed; instead there is a disproportionate number of 4-6m distances. According to Figure 4.9b the host vehicle is stopped for a particularly high percentage of frames. In fact, the 24% of images with host speed of 0 corresponds directly to the 24% of images with distance less than 6m. These statistics are a result of the stop-and-go nature of city driving, which comprises 86% of the data set.

Figure 4.10 plots all of the routes driven during the data generation process. The labeled vehicle coordinates of the PVVF-1.3M database provide a set of feasible driving locations, which may be useful for future data generation.

Figures 4.11-4.13 show the distributions of other driving environment conditions, specifically weather, daylight, and leading vehicle model. Note that while these variations will be evenly distributed at a sequence level, they may be skewed at the frame level due to the variability in sequence length. Any such skew should be kept in mind when considering analysis results.
Figure 4.9: Stop-and-Go City Driving. The high proportion of 4-6m separating distances (left) and 0 m/s speed (right) are a direct result of the GTAV urban driving environment.

Figure 4.10: Map of Sequence Locations. As can be seen, the Los Santos road network is quite dense.
Figure 4.11

Weather Distribution

- Clouds
- Clearing
- Thunderstorm
- Raining
- Overcast
- Foggy
- Extra Sunny
- Clear

Figure 4.12

Distribution of Time

Frequency

Time of Day (hour)
Figure 4.13

Histogram of Car Model Distribution
5 Procedure

5.1 Data Preparation

As described in Chapter 4, a data set of 1660 vehicle-following sequences, representing a combined 1.3 million images, was generated using the GTAV virtual world. This data was split into training, validation, and test sets containing 980K, 150K, and 180K images respectively. Each sequence is represented only once among these subsets for two reasons. First, as images within a sequence will be highly correlated, training and testing on the same sequence may give biased results. Furthermore, since a focus of this paper is on understanding temporal dependencies in car following, the sequential structure of the data must be left intact.

One may argue that images from different sequences still share the same data generation process and will therefore still introduce bias into the results. This is certainly true to an extent; however, the combination of purposefully selected variations, such as weather conditions and leading car model, with random environmental states, such as traffic patterns and driving routes, make it statistically unlikely that a given sequence has been effectively seen before. Thus, for the purposes of this paper, each sequence can be considered an independent sample.

The synthetic images are captured at an 800x600 resolution within GTAV but are downsampled at the time of collection to 280x210. Images are normalized and mean-zeroed in line with typical computer vision preprocessing practice. As the training data set consumes approximately 225GB in memory, it is too large to fit on a GPU. To solve this problem, custom python generators were built to iteratively feed batches of images and corresponding ground truth distances to the GPU. This process runs in parallel to the CNN computations, enabling faster training and evaluation.

5.2 Model Training

Estimation of DTLV is executed by two variations of CNNs - a traditional 2DConv model and an experimental 3DConv model. These CNNs were structured in Python
using the Google TensorFlow deep learning framework with Keras library add-on. Training was performed on a 12GB NVIDIA TitanX Superclocked GPU kindly provided through NVIDIA’s GPU Grant Program. Weights of both CNNs were updated using Stochastic Gradient Descent implemented with decay and momentum. Mean Absolute Error (MAE) was used as a loss function because it showed significant learning gains over Mean Squared Error (MSE). Hyperparameters were tuned by observing performance on the validation set.

5.2.1 RC Model

A baseline model performance is critical to understand the results of the 2DConv and 3DConv models. This baseline is given by the “Random Chance” (RC) Model, the regression analog to “flipping a coin”. The RC Model simply guesses the mean training set DTLV, regardless of input. Evaluated on the test set, the RC Model has an MAE of 8.82m across the test set. If a model performs worse than this, it has no predictive capability; on the other hand, models that produce lower error rates are able to find correlation between car following images and DTLV.

5.2.2 2DConv Model

The 2DConv Model is a nine-layer CNN similar to the standard AlexNet (Krizhevsky, Sutskever, and Hinton 2012). There are five convolutional layers and four fully-connected (FC) layers, each with Rectified Linear Unit (ReLU) activation function shown in Figure 5.1. The first four convolutional layers are identical to AlexNet, while the fifth layer has 384 kernels. Likewise, the first two FC layers have 4096 nodes, as in AlexNet, but the last two layers have 256 and 1 nodes as implemented by Chen (2016). Max Pooling after the first, second, and fifth convolution layers and dropout after the first two FC layers help reduce the parameter space.

The 2DConv Model was trained for 120K iterations using a batch size of 64 and a learning rate of 0.005. Validation was performed using a random sample of 30K images from the validation set and used to select hyperparameters. Training took approximately 20 hours.
In order to understand the stability of the model’s solution, a second 2DConv model was trained with identical structure and hyperparameters but using a different random seed. Figure 5.2 shows that the models reached approximately the same validation loss, suggesting that the models are stable relative to their initial starting condition.

Figure 5.1: ReLU Activation Function

5.2.3 3DConv Model

The 3DConv Model is based on the model proposed by Tran et al (2015). Just as in the 2DConv structure, there are five convolutional layers and four FC layers, with maxpooling and dropout occurring at the same locations. Taking 16 consecutive images as a four-dimensional input, 3DConv takes a slow fusion approach by maintaining this dimensionality until reaching the first FC layer.

By increasing the input dimension by a factor of 16 and retaining this dimensionality for several layers, the 3DConv parameter space was naturally larger than that of 2DConv. Shazeer et al (2017) note that the size of the parameter space is directly
Figure 5.2: Training and Validation of two 2DConv Models. Despite being initialized with different random seeds, and therefore training on images in a different order, both 2DConv models quickly reach the same validation loss. Note that variation in iterations per epoch gives the appearance of a divergence in training loss.
tied to the learning potential of a neural network. In order to make the models more comparable, the first two FC layers were reduced to 2048 nodes. As demonstrated in Table 5.1, 3DConv has fewer parameters than 2DConv.

### Table 5.1: Number of Convolution and Fully-Connected Layer Parameters (in millions)

<table>
<thead>
<tr>
<th>Layer</th>
<th>2DConv</th>
<th>3DConv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv Layers</td>
<td>4.1</td>
<td>4.7</td>
</tr>
<tr>
<td>FC Layers</td>
<td>30.4</td>
<td>28.3</td>
</tr>
<tr>
<td>Total</td>
<td>34.6</td>
<td>33.0</td>
</tr>
</tbody>
</table>

The 3DConv Model was trained for 80K iterations using a batch size of 16 and a learning rate of 0.003. The smaller batch size was implemented in order to accommodate the higher loading requirements of the 3DConv model (256 images per batch vs 64 in 2DConv). Training took approximately 4 days and was stopped early after validation loss stagnated. As demonstrated in Figure 5.3, 3DConv reaches a lower equilibrium validation error than 2DConv.

### 5.3 Model Testing

Evaluation of CNNs is much faster than training because only a forward pass is required and the process is even more parallelizable. The 2DConv model can predict at a rate of 70 frames per second, much faster than the 3DConv prediction of 16 frames per second. However, the slow speed of the 3DConv model is a result of its lack of optimization for prediction. In its current implementation, the 3DConv model repeatedly loads all 16 images at every evaluation. However, as driving is a sequential process, 15 of the 16 images can be retained within memory at every time step. Since image loading is the bottleneck for evaluation, this change would significantly speed up 3DConv prediction.

The 2DConv and 3DConv models were tested on four different sets of data. The primary analysis was conducted on the subset of PVVF-1.3M images originally allo-
Figure 5.3: Training and Validation Performance. 3DConv outperforms 2DConv on the validation set.
cated for testing. This testing set has the same theoretical distribution as the training set, so we should expect the best performance here.

The second testing set aims to understand the transfer learning capabilities of the 2DConv and 3DConv models by testing prediction of distance to motorcycles, bicycles, and other alternative vehicles. This data set of 102K images was generated through a similar process to that described in Chapter 4; as a result, ground truths are still fully labeled, allowing for proper assessment of performance.

The third testing set involves the host car driving through an empty road system to understand what the models do when there is no car in front. This data set followed a similar generation process as testing sets 1 and 2. 55K images in total were generated for this test set.

Finally, both models are evaluated on unlabeled real-world driving sequences to qualitatively gauge the cross-domain applicability of the trained networks. These images are captured in HD '.mp4' format using a Samsung Galaxy S7 attached to an Acura RDX dashboard. These images are cropped, scaled, and then downsampled to better match the synthetic training data.
6 Results and Discussion

6.1 Distance Estimation

A major objective of this research was to understand the extent to which 3D Convolution can improve regression accuracy as applied to the task of distance estimation. Based on the results given in Figure 6.1, both CNN models significantly outperform the baseline RC model. Figure 6.2 and Table 6.1 show that the 3DConv model had moderately higher prediction accuracy than the 2DConv model, improving median performance by 5.6% and mean performance by 6.8%. The 3DConv model’s boost in accuracy is complemented by an 11% reduction in variance shown in Table 6.2.

Figure 6.1: Residual Error Density Distribution. The RC model has the weakest performance and experiences a spike at 12m due to the high proportion of low distance data.

Since this analysis was conducted on a brand new data set, it is difficult to compare 3DConv’s distance estimation performance to the results of prior literature. This difficulty is compounded by the general lack of vision-based regression research as described in Section 3.5. An attempt to provide some context, however, is made in
Figure 6.2: 2DConv vs 3DConv Error Density. The left graph shows that the error distribution is similar between 2DConv and 3DConv. The right graph plots the difference in density (3DConv - 2DConv) by bucket. 3DConv appears to have a slightly more accurate error density curve.

Table 6.3, which places 3DConv’s results on GTAV video against those of models tested on TORCS by Chen (2016). GIST refers to the scene recognition algorithm proposed by Oliva and Torralba (2001), which Chen uses to train Support Vector Regression and Classification models that correspond to the affordance indicators in his Direct Perception ConvNet, referenced here as DP ConvNet for clarity. The GIST half algorithm uses just the bottom half of the image, which Chen theorizes may have all relevant information to drive. The DP ConvNet is structured very similarly to 2DConv, with the exceptions outlined in Section 5.2. The DP ConvNet sub and full refer to whether the model is trained on the full data set of 480K samples or an 86K sample subset.

An issue with this comparison is that Chen’s distance affordances differ slightly from the DTLV studied in this paper, as they estimate the distance to cars in each lane without direct consideration for the host car’s lane position. In order to account for this discrepancy, Table 6.3 lists the most accurate distance indicator for each model.
Table 6.1: Model Prediction Error (in m)

<table>
<thead>
<tr>
<th>Model</th>
<th>Low</th>
<th>Median</th>
<th>Mean</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>0.00</td>
<td>8.21</td>
<td>8.82</td>
<td><strong>38.99</strong></td>
</tr>
<tr>
<td>2DConv</td>
<td>0.00</td>
<td>2.31</td>
<td>4.43</td>
<td>42.69</td>
</tr>
<tr>
<td>3DConv</td>
<td><strong>0.00</strong></td>
<td><strong>2.18</strong></td>
<td><strong>4.13</strong></td>
<td>45.96</td>
</tr>
</tbody>
</table>

Table 6.2: Model Variance (in $m^2$)

<table>
<thead>
<tr>
<th>Model</th>
<th>RC</th>
<th>2DConv</th>
<th>3DConv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variance</td>
<td>42.89</td>
<td>27.95</td>
<td><strong>24.79</strong></td>
</tr>
</tbody>
</table>

While this heuristic benchmarking may not be a true apples-to-apples comparison, there are a number of reasons why this is actually an interesting result. First, the similarity of 2DConv with DP ConvNet full suggests that their performance differences must have to do with either the differences in objective or domain. The vehicle-following problem may be easier in some ways, but the GTAV driving environment is far more complex. Further research into the predictive dynamics between these virtual worlds - specifically what makes estimation in one world more difficult - may shed light on some open problems in domain adaptation.

Furthermore, the similarity in scale of the error distribution serves as a much-needed sanity check that the 2DConv and 3DConv models are producing reasonable results. This then leads to the question of whether any vision-based cognition algorithm can predict distance much better than on the order of four meters.

6.2 Percentage Error

Ultimately, absolute distance errors are less practically important than relative ones. Being off by one meter is irrelevant if a leading car is fifty meters away, but is unacceptable if that car is just two meters away. Table 6.4 outlines the percentage error
Table 6.3: Comparing Distance Prediction Errors

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Absolute Error</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIST whole</td>
<td>13.08</td>
<td>11.53</td>
</tr>
<tr>
<td>GIST half</td>
<td>12.75</td>
<td>12.20</td>
</tr>
<tr>
<td>DP ConvNet sub</td>
<td>8.61</td>
<td>10.68</td>
</tr>
<tr>
<td>DP ConvNet full</td>
<td>4.738</td>
<td>7.816</td>
</tr>
<tr>
<td>2DConv</td>
<td>4.43</td>
<td>5.29</td>
</tr>
<tr>
<td>3DConv</td>
<td><strong>4.13</strong></td>
<td><strong>4.98</strong></td>
</tr>
</tbody>
</table>

and paints a less rosy picture. Both 2DConv and 3DConv have a median percentage error of 24% but have huge outliers. It appears that these models may not have been particularly accurate at detecting vehicles at short distances. Furthermore, 3DConv appears to perform at least as poorly as 2DConv, if not worse.

Table 6.4: Model Prediction Percentage Error by Quantile

<table>
<thead>
<tr>
<th>Model</th>
<th>Low</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>99%</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>0%</td>
<td>22%</td>
<td>53%</td>
<td>220%</td>
<td>820%</td>
<td>11,000%</td>
</tr>
<tr>
<td>2DConv</td>
<td>0%</td>
<td><strong>10%</strong></td>
<td><strong>24%</strong></td>
<td>57%</td>
<td>670%</td>
<td><strong>7,900%</strong></td>
</tr>
<tr>
<td>3DConv</td>
<td>0%</td>
<td>11%</td>
<td>24%</td>
<td><strong>52%</strong></td>
<td>460%</td>
<td>11,000%</td>
</tr>
</tbody>
</table>

A somewhat mitigating factor is that 2DConv and 3DConv appear to be relatively accurate in predicting within reasonable thresholds. As Table 6.5 indicates, 3DConv correctly guesses distance within three meters well over half of the time. These results imply that the models would be quite accurate in classifying distance into “near”, “midrange”, and “far” buckets.

Figure 6.3 shows both models’ percentage errors plotted against their corresponding ground-truth distance. In general, there appears to be a trend of percentage error increasing as the separating distance widens. This makes sense as the number of
Table 6.5: Prediction within Threshold

<table>
<thead>
<tr>
<th>Model</th>
<th>1m</th>
<th>3m</th>
<th>5m</th>
<th>10m</th>
<th>20m</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>7.4%</td>
<td>21.8%</td>
<td>34.0%</td>
<td>57.7%</td>
<td>93.8%</td>
</tr>
<tr>
<td>2DConv</td>
<td>21.8%</td>
<td>57.9%</td>
<td>71.8%</td>
<td>87.8%</td>
<td>99.6%</td>
</tr>
<tr>
<td>3DConv</td>
<td>22.4%</td>
<td>60.0%</td>
<td>74.3%</td>
<td>89.4%</td>
<td>99.7%</td>
</tr>
</tbody>
</table>

pixels representing the leading vehicle reduces with distance, making prediction more difficult. At short distances, small absolute errors translate into large percentage errors, particularly when the CNN fails to recognize the close-range car. This effect is compounded by the MAE loss function used during training, which equally weights a deviation between 1-2m and 50-51m.

Figure 6.3: Percentage error distribution by separating distance.

Figure 6.4 plots image frame locations in two buckets separated by a DTLV threshold of 10m. Coordinates are colored based on whether they had high relative error in the 3DConv model, defined as 50% or higher. One interesting observation is that high percentage error for longer DTLV often occurs at intersections, which are certainly complex driving situations. This finding suggests that attention should be paid to intersection recognition in future work. Another point to note is the relative sparsity of points for the less than 10m case, even though approximately 41% of points fall
into this bucket. This is because short DTLV tends to only occur at speeds near or equal to 0, meaning that many of these images are taken at the same location.

### 6.3 Effects of Driving Environment Variation

As mentioned in Chapter 4, the GTAV virtual platform allows easy manipulation of driving conditions that are difficult to study at scale in the real world. Figures 6.6 and 6.7 show that estimation errors tend to be higher in rain and thunderstorms as well as during sunrise and sunset. Both of these results are intuitive - visibility is heavily affected under those weather conditions and relatively rapid brightness shifts during those hours may disorient a visual perception system.

Since the 3DConv model has the capability to learn time-dependent features, we may have expected it to perform even better under higher speeds relative to the other models. Instead, the results in Table 6.6 indicate that the 3DConv absolute residuals are more correlated with higher speed than the 2DConv absolute residuals. One explanation may be that the bias in data toward lower speeds causes 3DConv to misinterpret or underweight motion elements at high speed. A possible fix would be to supply 3DConv with a velocity input, which should be easily accessible in any...
Figure 6.5: **Locations of Relative Error** > 20% This graph shows that 2DConv and 3DConv errors are generally within 20% of each other. A notable observation is that 2DConv high errors are typically scattered while 3DConv high errors appear to persist for several frames due to its time-dependent prediction structure.

Figure 6.6: **Mean Distance Error by Weather.** Rain and Thunder have the highest mean error.
Figure 6.7: Mean Distance Error by Time of Day. Note that the hours of sunrise and sunset appear to have highest error.

Driving system. One further consideration to note is that higher speeds tend to occur when DTLV is larger, and larger DTLV tends to produce higher error as shown in Figure 6.3, so this may just be correlation rather than causation.

Table 6.6: Correlation of Residuals with Host Speed

<table>
<thead>
<tr>
<th>Model</th>
<th>RC</th>
<th>2DConv</th>
<th>3DConv</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \rho_v )</td>
<td>0.06</td>
<td>0.25</td>
<td>0.30</td>
</tr>
</tbody>
</table>

3DConv appears to be more robust against variation in leading vehicle model than 2DConv. Figure 6.8 shows that the vehicle models with highest mean absolute error in 3DConv tended to have lower mean error than those in 2DConv. Figure 6.9-6.10 show the vehicle models with highest mean error for 2DConv and 3DConv. The downside of having such an extensive number of car models was that the test set was only able to include a few sequences of each model, despite its large size. Since there are many other factors that could influence distance error and due to the high correlation between pictures within a sequence, we refrain from making judgments on
the effects of leading vehicle models.

Figure 6.8: Mean average error by vehicle model (left and middle) and relative differences between 2DConv and 3DConv (right). Note that negative values on the right graph correspond to better performance by 3DConv.

6.4 Transfer Learning

The flexibility and feature-rich nature of GTAV enables many types of experimentation, transfer learning being an obvious example. An additional test data set was generated consisting of 14 vehicle models - 10 motorbikes, 1 golf cart, and 3 bicycles. As 2DConv and 3DConv had not been trained on these alternative forms of transportation, this offered an interesting perspective on the generalizability of their learning. Table 6.7 shows that 3DConv performed significantly better than 2DConv across all new vehicle models. While 3DConv appeared to fare no better than chance in predicting distance to bicycles, 2DConv actually underperformed the RC model on
Figure 6.9: Least Accurate Vehicle Models for 2DConv. These models had a combined MAE of 10.56m.

Figure 6.10: Least Accurate Vehicle Models for 3DConv. These models had a combined MAE of 8.96m.
both bicycles and golf carts. One explanation for 3DConv's relative success is that it may have been able to understand vehicles not just as shapes, but as moving objects due its temporal learning capacity. This in turn may have led to a recognition that motorcycles and golf carts are also moving objects which should be detected.

**Table 6.7: Mean Absolute Error**

<table>
<thead>
<tr>
<th>Model</th>
<th>RC</th>
<th>2DConv</th>
<th>3DConv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorcycles</td>
<td>7.78</td>
<td>6.67</td>
<td><strong>6.10</strong></td>
</tr>
<tr>
<td>Golf Cart</td>
<td>5.39</td>
<td>5.42</td>
<td><strong>4.76</strong></td>
</tr>
<tr>
<td>Bicycles</td>
<td><strong>6.49</strong></td>
<td>6.90</td>
<td>6.50</td>
</tr>
<tr>
<td>Full Data Set</td>
<td>7.12</td>
<td>6.73</td>
<td><strong>6.24</strong></td>
</tr>
</tbody>
</table>

**Table 6.8: Median Absolute Error**

<table>
<thead>
<tr>
<th>Model</th>
<th>RC</th>
<th>2DConv</th>
<th>3DConv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorcycles</td>
<td>6.61</td>
<td>4.96</td>
<td><strong>3.87</strong></td>
</tr>
<tr>
<td>Golf Cart</td>
<td>4.94</td>
<td>4.08</td>
<td><strong>3.53</strong></td>
</tr>
<tr>
<td>Bicycles</td>
<td>5.26</td>
<td>5.50</td>
<td><strong>4.46</strong></td>
</tr>
<tr>
<td>Full Data Set</td>
<td>5.69</td>
<td>5.14</td>
<td><strong>4.10</strong></td>
</tr>
</tbody>
</table>

### 6.5 Open Road Driving

While 3DConv and 2DConv have only been trained on vehicle-following data, it is unrealistic to assume that there will always be a leading vehicle within sight. In order to understand how these models would treat an open road driving scenario in which there are no other cars, a third test set was created consisting only of the host driving on an empty road network. Predictions on this data set appear to be nearly normally distributed for the 2DConv model. The 3DConv model has a large spike
in predictions of low distances, which may suggest it has found correlations in the external environment that signal low distances, such as traffic lights. Implications of this result are discussed in Section 7.2.

6.6 Real World Driving

The end goal of research into virtual driving is to produce relevant real world results. This is difficult for the vehicle-following problem being studied due to the data constraints discussed in Section 4.1. As an initial exploration into this space, raw footage was captured while driving on Route 206 in central New Jersey. While unlabeled, one can get a general sense of distance by qualitatively comparing images to a set of reference frames, for which the true distance to parked cars is measured and labeled. Based on initial testing, 2DConv appeared able to detect real cars, despite never having been trained on such data. 3DConv predictions appeared to be more volatile, although this may be due to an inconsistent treatment of frame frequency. Nevertheless, these are promising results and warrant further research.
Figure 6.12: Prediction Distribution for Open Road Driving

Figure 6.13: Open Road Driving Sequence Map. Coordinates are color coded based on which distance prediction bucket the image at that point yielded.
Figure 6.14: Prediction on Real Driving Footage. Sequence order is clockwise from top.
7 Conclusion

7.1 Further Discussion

The relatively strong estimation performance of the 3DConv model across almost every dimension studied, despite its smaller parameter space, suggests that 3D convolution may enable more powerful feature detection than standard 2D convolution. One question that is worth asking is - under what circumstances does the 3DConv model outperform the 2DConv model? The predictions of both models are highly correlated ($\rho = 0.86$), which may mean that they detect similar features in images. Based on the results in Section 6.3, the 3DConv model appears to be more invariant to environmental differences, which may result in its better performance. However, the underlying reason for such robust behavior is still unknown.

While 3DConv did perform better, it is possible that this may have more to do with 2DConv being overtrained than 3DConv itself, as Figure 7.1 suggests may be the case. This risk is partially mitigated by the fact that the other trained 2DConv model produced similar results.

Another key question is whether the algorithm is actually learning or simply taking advantage of correlation within the data. One important feature of driving, for example, is brake lights during the night. In order to test whether the 2DConv model recognizes this feature, I generated two night-time driving images in GTAV that are identical in every way except that the leading vehicle brake lights are turned on in one of them. 2DConv’s prediction was slightly less accurate when the brake light was turned on, suggesting that the network may not have properly learned to recognize this feature.

The brake light experiment touches on a more practical idea - it may be possible to more intelligently train networks by feeding them sequences of images that only differ in one aspect that we want them to learn. While impossible to do in real life, such an undertaking would certainly be feasible within a simulator.
Figure 7.1: Training vs Test Error Distribution. As is typical, the model performs better on the training data set for both models. However, it is worth noting that the training set for 2DConv has significantly lower MAE than the training set for 3DConv. This may indicate a possible overfitting of 2DConv.

Figure 7.2: Analysis of Brake Light Feature. Turning on the brake light actually reduced prediction accuracy.
7.2 Limitations and Future Research

A major limitation of this work is that all of the training and testing is done offline on static data. In a real-time system, predictions made at time \( t \) influence input received at \( t + 1 \), which can lead to instability issues that are not seen in offline prediction. The next steps moving forward would be implementation of Chen’s Direct Perception model in GTAV, in order to more accurately compare results between TORCS and GTAV. One roadblock had been a compatibility issue between Windows-based GTAV and Linux-based Deep Learning Frameworks. However, the recent Windows-support for TensorFlow eliminates this concern.

Another limitation is that the network does not seem to perform well at predicting short distances. This may stem from being trained using an MAE loss function rather than by percentage error. Regardless, it may be more appropriate to have separate networks trained for short and far distances, similar to Filipowicz et al. (2017). Furthermore, the networks only provide prediction of distance conditioned on the fact that they are following a car. As shown in Section 6.4, the prediction algorithm fails when there is no car in front. A separate classification network that decides whether the car is in a vehicle-following environment would be essential to proper distance estimation.

There also exists much room to experiment with transfer learning and domain adaptation. While the initial forays discussed in this paper are promising, a more comprehensive study is needed. For testing on real data, it would be useful to acquire some type of distance sensor, such as lidar or radar, to quantitatively evaluate the cross-domain applicability of virtual world-trained networks.

Finally, while 3D convolution is one way to incorporate time into the CNN, there are other strategies such as Long Short-Term Memory units that can store temporal information far longer than 3D convolution. A detailed empirical analysis comparing the various types of networks would be helpful in setting the standard for time-dependent CNNs.
References


