Simulation Framework for Driving Data Collection and Object Detection Algorithms to Aid Autonomous Vehicle Emulation of Human Driving Styles

by

Yashaswy Govada

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2020 by the
Graduate Supervisory Committee:

Spring Berman, Chair
Kathryn Johnson
Hamidreza Marvi

ARIZONA STATE UNIVERSITY

December 2020

ABSTRACT

Autonomous Vehicles (AVs), or self-driving cars, are poised to have an enormous impact on the automotive industry and road transportation. While advances have been made towards the development of safe, competent autonomous vehicles, there has been inadequate attention to the control of autonomous vehicles in unanticipated situations, such as imminent crashes. Even if autonomous vehicles follow all safety measures, accidents are inevitable, and humans must trust autonomous vehicles to respond appropriately in such scenarios. It is not plausible to program autonomous vehicles with a set of rules to tackle every possible crash scenario. Instead, a possible approach is to align their decision-making capabilities with the moral priorities, values, and social motivations of trustworthy human drivers. Toward this end, this thesis contributes a simulation framework for collecting, analyzing, and replicating human driving behaviors in a variety of scenarios, including imminent crashes. Four driving scenarios in an urban traffic environment were designed in the CARLA driving simulator platform, in which simulated cars can either drive autonomously or be driven by a user via a steering wheel and pedals. These included three unavoidable crash scenarios, representing classic trolley-problem ethical dilemmas, and a scenario in which a car must be driven through a school zone, in order to examine driver prioritization of reaching a destination versus ensuring safety. Sample human driving data in CARLA was logged from the simulated car's sensors, including the LiDAR, IMU and camera. In order to reproduce human driving behaviors in a simulated vehicle, it is necessary for the AV to be able to identify objects in the environment and evaluate the volume of their bounding boxes for prediction and planning. An object detection method was used that processes LiDAR point cloud data using the PointNet neural network architecture, analyzes RGB images via transfer learning using the Xception convolutional neural network architecture, and fuses the outputs of these two net-

works. This method was trained and tested on both the KITTI Vision Benchmark Suite dataset and a virtual dataset exclusively generated from CARLA. When applied to the KITTI dataset, the object detection method achieved an average classification accuracy of 96.72% and an average Intersection over Union (IoU) of 0.72, where the IoU metric compares predicted bounding boxes to those used for training.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude towards Dr. Spring Berman for giving me this amazing opportunity to learn and develop my skills within team at the Autonomous Collective Systems Laboratory. Dr. Berman made herself always available whenever I had any troubles in my research. Without her guidance and persistent help this thesis would not have been possible.

I would like to thank Dr. Kathryn Johnson for her valuable inputs into the Psychological aspects of the research work and trusting me with her project. I would also like to thank Dr. Hamid Marvi for supporting and encouraging me throughout my coursework and research.

I dedicate this thesis to my parents and my sister for their unconditional love, prayers and caring. You'll were always there for me. I am extremely grateful for the constant support and love I received directly or indirectly from all of my family. Finally, I would like to express my gratitude towards my friends Omik Save, Bhargav Ram, Sowparnika Koka, Akshay Reddy and many others for supporting and motivating me throughout this graduate degree.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Autonomous Vehicles (AVs) and Levels of Autonomy

Driving is an amalgam of continual risk assessment, environmental awareness, decision making and adapting to tremendously variable surroundings or weather conditions. Automotive Industry in alliance with Robotics and Control Systems fields have created ground-breaking changes in road transportation by introducing Autonomous Vehicles (AVs). Ideally, a vehicle that can guide and maneuver itself from a beginning point to a preset destination without any human interaction but with the help of various technologies and sensors like Adaptive cruise control, Lane detection and centering, GPS, steering control, Cameras and LiDARs etc. is called an Autonomous Vehicle or a Self-Driving Car . (Business Insider, 2018).

The advent of Automated Driving technology dates back to at least the 1920s. It was not until the 1980s that a significant break-through was registered in the field, when Carnegie Mellon University's Autonomous Land Vehicle(ALV) (Kanade *et al.*, 1986), funded by Defense Advanced Research Projects Agency (DARPA), demonstrated self-driving capabilities on a two lane road with obstacle avoidance in 1986 and later they went on to complete the first autonomous coast-to-coast drive across the United States, spanning 2797 miles (NavLab 5, 1995). After witnessing these milestones many companies like Google, Ford, Uber and Tesla have embraced this technology and trusted it to be the next "Big Thing".

The AVs are already functional and are a reality today. There are six Levels (Level 0 to Level 5) of Driving Automation (Refer to Table 1.1) described by the Society of

Figure 1.1: Famous Companies Working on Self Driving Cars (Self Driving Times, 2016)

Automotive Engineers (SAE)(SAE, 2018). The Self-Driving Cars seen on the roads today are at Level 1-3 Automation. One of the predominant reasons for being stuck at Level 3 is the lack of complete safety and a robust framework capable of making moral decisions during crash scenarios to minimize any kind of damage to life or property. The ultimate aim is to achieve Level 5 Automation wherein your car acts as your chauffeur and can perform driving tasks without any human supervision. Renowned companies and the big players in this market, Tesla and Google Waymo are pursuing unprecedented measures to achieve Level-4 Automation. As of October, 2020 Google Waymo has introduced its "Fully Driverless Service" to the general Public in Phoenix, Arizona, USA (Google Waymo, 2020) and Telsa has rolled out its "Full Self-Driving" update (Andrew Hawkins, 2020) which is extremely slow and cautious as declared by Telsa's CEO, Elon Musk.

Despite all this progress Self-Driving Cars currently are not expected to perform on par with human drivers due to its safety restrictions and lack of trust bestowed upon AVs by general public (Hengstler *et al.*, 2016; Dixit *et al.*, 2016). This trust can be built only if the public and passengers are completely convinced that an AV is safe and it makes better decisions than humans, especially during crashes.

Figure 1.2: Google Waymo (Left) and Telsa (Right) (Mike Brown, 2018)

Automation Level	**Title**	**Description**
Level 0	No Driving Automation	Completely Manual Control. Despite having a few systems for warnings, the vehicles is not Autonomous
Level 1	Driver Assistance	The Vehicle exhibits only one automated system for either steering or acceleration.
Level 2	Partial Automation	The Vehicle can control both steering and acceleration. Constant human supervision is necessary
Level 3	Conditional Automation	The Vehicle is aware of its environment with the help of its sensors and can make calculated decisions. However, the driver must be alert and ready to take control whenever needed
Level 4	High Automation	These vehicles do not need any human intervention. The only drawback here is, the vehicles can operate only under certain conditions
Level 5	Full Automation	The vehicle is completely automated and will function as effectively as a human driver or better

Table 1.1: SAE - Levels of Automation and Description, (SAE, 2018)

1.2 Advantages and Challenges of AVs

Irrespective of the proficiency of the driver, crashes are inevitable and Autonomous Vehicles are not an exclusion to this. From 2014 to 2018, 62 accidents were reported when an AV was on "Autopilot" of which only one was caused by the AV itself (Business Insider, 2018). Road accidents are alone the greatest cause of unusual deaths of healthy individuals aged below 54. (CDC, 2019) According to Annual Global Road Crash Statistics, approximately 1.35 million lives are lost in road crashes every year i.e, 3700 lives per day and additionally, 20-50 million are injured critically enough to require medical assistance (T Pietrasik, 2020). The need for Autonomous Vehicles has been inspired from the bitter fact that 94% of serious crashes are a result of human error or distraction (NHTSA, 2019). Including the potential to reduce these accidents, AVs possess the ability to minimize traffic, mitigate harmful emissions, promote ride sharing, ease parking struggle and assist the elderly and disabled through greater mobility, thus increasing their independence.

Despite its comforting prospects there are a few drawbacks. Any vehicle, irrespective of its capabilities, moving at a considerable speed is bound to crash in certain scenarios. Whilst travelling millions of miles collectively, AVs have been involved in a few crashes (Andrew Hawkins, 2020). The first fatal crash took place in 2016, on a highway, near Handan, China when the car couldn't identify the sweeper truck and crashed right into the back of it (Neal Boudette, 2016). Soon after this a similar crash took place in Florida resulting in the death of the driver again (Danny Yadron, Dan Tynan, 2016). In 2018, the death of Elaine Herzberg caused by an Uber Self-Driving Car in Tempe, Arizona (Greg Bensinger, Tim Higgins, 2018), was recorded as the first accident resulting in the death of a Pedestrian. The results of these crashes were placed under thorough examination. In addition to technical errors, security concerns

and other vulnerabilities, decision making capabilities of the vehicle were recognized to be one of the main barriers hindering AVs from becoming a reality. Ultimately, the decisions made by these vehicles will be judged by the ethical and moral standards of the society they drive in (Gerdes and Thornton, 2015). These vehicles must be able to make clear technical and moral decisions in dilemma situations to perform better than, or comparably to, human drivers and to enable this, the programmers of the AVs must make sure that the control algorithms cause actions that are approved legally and ethically.

The public's Trust in Autonomous Vehicles is another challenge that AVs have to overcome. The way these vehicles manage to smoothly handle the social interactions a human encounters in traffic on a daily basis will strongly impact their social acceptance (Gerdes and Thornton, 2015). The American Automobile Association asserts that 3 out of 4 U.S drivers are afraid to ride in Autonomous Vehicle (American Automobile Association, 2019) and about half of them felt less safe sharing the road with these vehicles. This trend has been changing gradually after witnessing the deployment of Partially Automated Vehicles and the current generation are embracing this relatively new technology of Self-Driving Cars. But for complete deployment of Autonomous vehicles the car has to function as effectively as a human being or better. For this purpose a lot of effort is being put into programming autonomous vehicles to exhibit ethical decision making.

1.3 Morals and Ethics

With improvements in research areas like big data, computation, sensing and Machine Learning over the past few decades, there has been decent growth in Vehicle Automation. This growth has given birth to another new subject of study known as Machine Ethics or Machine Morality (Wallach *et al.*, 2010; Goodall, 2014b). For

their acceptance by the society, AVs should exhibit socially approved stances when encountering moral or ethical dilemmas. Fagnant and Kockelman (2015); Gerdes and Thornton (2015) discuss and predict ethical challenges that might take place upon deploying AVs into the world.

Ideally, it makes sense to program AVs with driving styles that align with the best practices of good drivers (Basu *et al.*, 2017). The intent of the programmers is to directly code in such a way that provides the AV with commands on how to behave in diverse situations. The approach to do this has generally been based from two schools of thought from the ethical theories, Deontology and Utilitarianism.

Utilitarian ethics, introduced and developed by Jeremy Bentham, emphasizes "the greatest good for the greatest number" (Kuipers, 2016). It was asserted that to be moral or not was entirely dependent on the outcome that causes least damage or choosing the best of all possible outcomes. For this reason Utilitarian ethics was also referred to as the Consequentialist approach.

Deontology emphasizes on following the rules no matter what the outcome is. Deontology is the ethical concept that uses rules to distinguish between right and wrong. It was first put forward by a European philosopher Immanuel Kant. Deontology believes in following universal moral laws like Don't steal and Don't Cheat. Deontology is simple in its proposition. It describes that people should just follow the rules and fulfill their duties no matter what. Unlike Utilitarianism, which judges actions by their outcomes, deontology doesn't consider costs and implications of the outcome. This disregards the context and uncertainty because you only have to follow the rules.

So far it has been observed that the notion of implemented ethics for AVs either aligns with deontology, that preaches following rules irrespective of the result, or utilitarianism, where minimizing damage or optimizing a crash outcome on the basis

of severity is desirable. Limiting the scope to only these ethical theories will cause neglecting the other dominant human morals and values that might have a significant impact on the crash decisions.

Virtue Ethics, according to its proponents, is the ethical theory that should be utilized in Self driving cars (Thornton *et al.*, 2016). It is an approach that reflects the values and character of the person rather than the external rules or outcomes. Virtue ethics explores concepts like how virtues are acquired and applied to different scenarios one might face. This is another possible approach towards incorporating moral integrity into AV controllers.

Most accepted practices to estimate these moral priorities and values are questionnaires or surveys. The most widely used questionnaires for this are the Moral Foundations Theory (MFT) as specified in Graham *et al.* (2013) and Basic individual values proposed and updated by Schwartz (1992); Schwartz *et al.* (2012).

1.4 Trolleyology

There has been great progress in vehicle automation and autonomous vehicle technology, yet, little has been done towards researching optimal crashing responses for Automated Vehicles. Drivers in unsafe situations, where collision is inevitable, are forced to make quick decisions with little or no planning (Goodall, 2014a). The lack of consensus about the right crash decisions is an important factor that makes ethical programming complex for crash scenarios. The Trolley Problem is one of the most famous debates in the discussions of ethical dilemmas. This story to explore moral dilemmas was introduced by Philippa Foot in 1967 (Foot, 1967). The trolley problem story assumes a situation where a trolley is navigating around a corner and the driver notices that ahead on the track there are five men tied to the rail tracks.

The driver tries to stop the vehicle to avoid killing them but realizes that he has lost its control due to brake failure. The trolley is now headed towards 5 people tied to a track and the only thing under his control is the steering which would help him redirect the trolley to another track. However, on this track too there is one person tied to the tracks. (Figure 1.3)

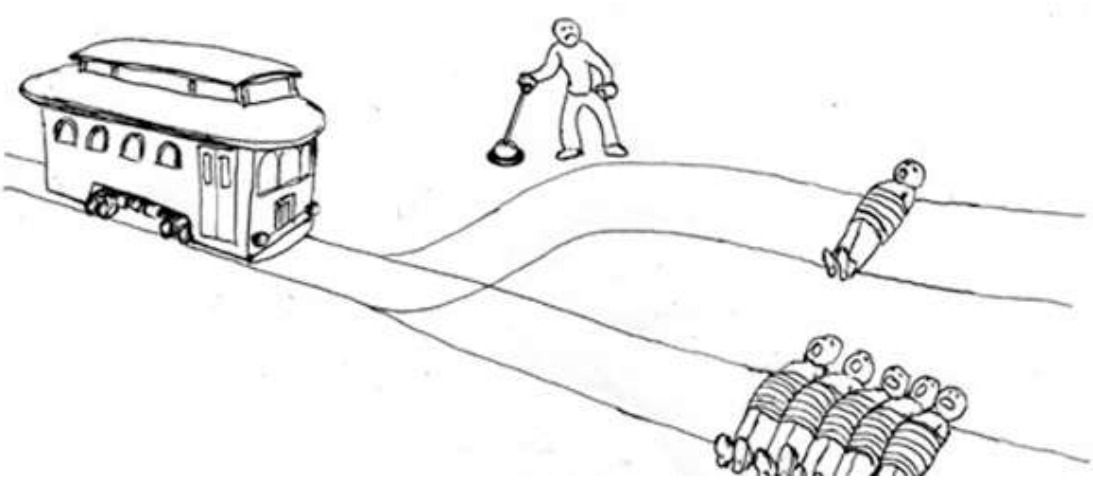


Figure 1.3: Trolley Problem Scenario 1 (Alyssa, 2016)

The whole motivation behind this story is to illustrate the outcomes of decisions that are consistent with different ethical theories and how one perceives ethics. Thomson (1985) in her paper questions if it is morally permissible for the driver to turn the trolley? If the driver follows a utilitarian approach he would change the track to kill one person instead of five; even if he chooses deontology he is supposed to change the track and cause the death of the person so he follows the rules by endangering only one as opposed to five. (Figure 1.4)

However there would be differences in opinion if this story can be tweaked a little to change it into a situation where the subject in this context is standing behind a

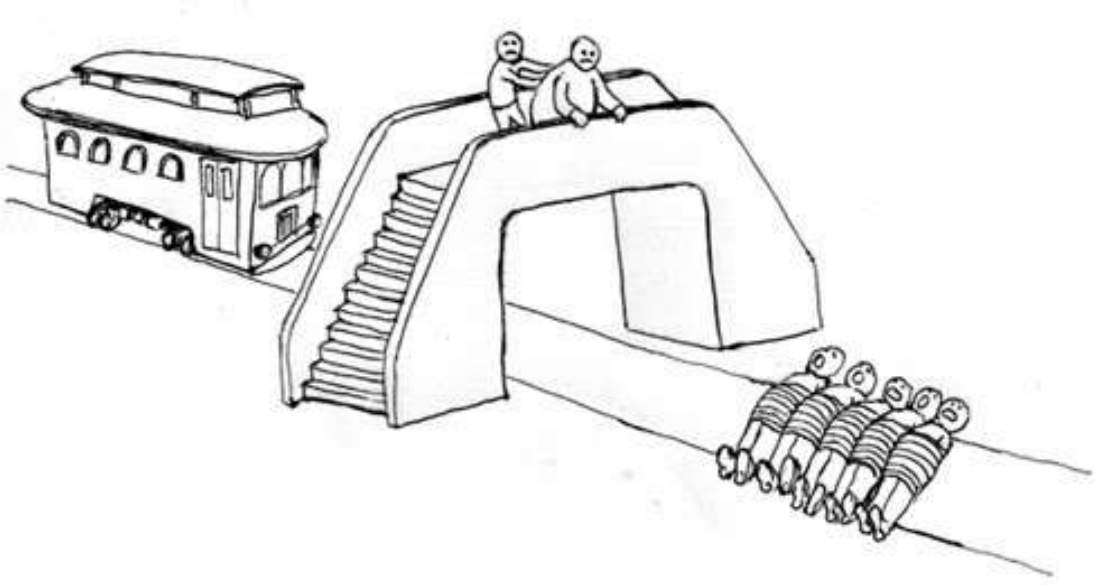


Figure 1.4: Modified Trolley Problem Scenario (Alyssa, 2016)

fat man and they are on an overpass. The subject notices a train on a slope sliding towards 5 people tied to the track. The only way for the subject to save the five people is to push the fat man on to the tracks to save the 5 members. According to the utilitarian approach this exactly seems like the outcome that the subject should choose. But for a supporter of deontology this is totally against his values as in this case he would be exploiting the person thus killing him.

The trolley problem has been a relevant debate even in the context of AVs. But how does the community come to an agreement on the particular ethical theory to use as a basis for programming the AV?. Simulating such scenarios to understand people's responses towards this problem seems like a viable solution.

1.5 Unreal Engine and CARLA Simulator

The Unreal Engine is a game engine created by Epic Games (Unreal Engine, 2004), first introduced in 1998 through the game "Unreal". The Unreal engine has been

modified several times since then and has established itself as one of the prominent game developing platforms. Programmed in C++, the Unreal engine is a highly flexible tool that supports diverse platforms. 'Blueprints' is the GUI system inside Unreal Engine that facilitates scripting without the need to write lines of code but by utilizing an intuitive node-based interface (Unreal Engine Documentation, 2014). These blueprints can be used to manipulate behaviors of agents or objects in the simulation. Its abilities can be expanded using the comprehensive 'Plugins' system. Since 2015 the Unreal Engine has been available for free to encourage developers from all Research fields.

CARLA- Car learning to act is an open source simulator for Automated Driving Research (Dosovitskiy *et al.*, 2017) developed by Intel Labs in collaboration with The Toyota Research Institute and the Computer Vision Center at Universitat Autonoma de Barcelona in Barcelona, Spain. CARLA exploits Unreal Engine's abilities to simulate dynamic worlds with convincing Physics of Materials, Animations, Virtual Reality support and outstanding rendering capabilities. As compared to other simulators like Airsim, CarSim, TORCS, Udacity Simulator and others, CARLA stands out as the most suitable and ever evolving simulator compatible with the scope of this thesis. Mainly, CARLA eases training, testing and validation of Autonomous Vehicles in urban traffic scenarios permitting us to make desired changes to the environments. To encourage future modifications and upgrades by the community, CARLA is built as an open-source layer over the Unreal Engine.

For the purposes of this thesis, CARLA (Version 0.9.10) was built on an Ubuntu distribution (18.04 LTS) of Linux. The following table lists the specifications of hardware used. However, a high performance Graphic card would be ideal for this application. Including the Driving Simulator, CARLA also provides digital assets like buildings, vehicles, humans and other features that are characteristics of urban

Operating System	Ubuntu 16.04 and above
RAM	8 GB
Memory	1TB (Need a minimum of 60-80 GB free disk space)
GPU	Nvidia GeForce GTX 470 or higher with latest NVIDIA drivers
Processor	Quad-core Intel or AMD, 2.5GHz or faster

Table 1.2: Recommended Hardware Specifications to Build CARLA on Unreal Engine (CARLA- Car Learning to Act, 2017)

environments to inspire imagination in creation of new environments or maps. The Simulator also offers a wide range of sensor suite including Cameras, LiDARs, GNSS (Global Navigation Satellite Systems), IMU (Inertial Measurement Unit) amongst others. Other features of CARLA include dynamic weather conditions, full control of all agents, ROS (Robot Operating System) bridge, map generation and customization and many more. The ROS bridge in particular is a very useful feature which can help study the behaviors and moral profiles extracted from the human driving data in a multi-agent ecosystem.

1.6 CHARTOPOLIS Test Bed

The 'CHARTOPOLIS' (Subramanyam, 2018) is a small scale traffic test bed established by the Autonomous Collective Systems Lab in ASU. Primarily, it was set up as a safe, controlled environment to study interactions between Autonomous vehicles and human drivers. Initially Pheeno Robots (Wilson *et al.*, 2016) were used for testing the lane detection and traffic light detection alogrithms. Later Go-CHARTs will be used to emulate AVs on this test bed. Go-CHART (Kannapiran and Berman, 2018) is 4-wheeled robot which replicates a scaled version of a standard sedan. It was designed and manufactured by the Autonomous Collective Systems Laboratory for its use on the CHARTOPOLIS test bed. This robot is fitted with some of the

most useful sensors that you can find on an AV. All the studies and tests will be conducted using these robots since they have much of the functionality of full-size AVs such as perception, localization and state estimation. For the purpose of this thesis, CHARTOPOLIS is referred to as the Physical domain and CARLA is referred to as the Virtual Domain.

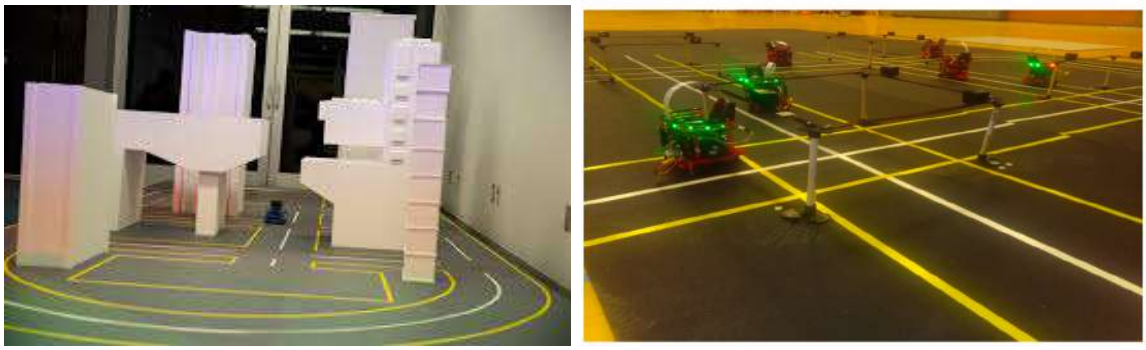


Figure 1.5: CHARTOPOLIS Test Bed at Polytechnic Campus, ASU (Left). Pheeno Robots Navigating an Earlier Version of CHARTOPOLIS (Right) (Subramanyam, 2018)

1.7 3D Object Detection

Perception of the environment is an important prerequisite for autonomous vehicles. With ever rising number of driverless vehicles on the road, human safety is an alarming concern. Despite facing issues with time complexity, computational capacity and robustness, use of computer vision, facilitated by machine learning algorithms, has led progress in this field. Identifying, Classifying objects and estimating the volume enclosed by these objects is a crucial capability for AVs and is necessary for implementing safety measures. Including the camera data, exploiting valuable information from high precision instruments like LiDAR has proven to be significantly effective in achieving 3D object detection and classification. While research

in computer vision has produced algorithms to analyze objects in a scene, a robust and reliable architecture is yet to be standardized. This work is a sincere effort towards developing an algorithm that harnesses point cloud information of LiDAR in conjunction with RGB images captured by cameras on an AV to determine the class and volume of objects in a scene through sensor fusion. In this approach, the use of PointNet (Qi *et al.*, 2016), to analyze LiDAR's point clouds and 'Exception' (Chollet, 2017), trained on 'ImageNet' (Deng *et al.*, 2009) to analyze camera RGB images, thus implementing transfer learning, was proposed. The world is moving towards training neural network models using synthetic datasets for application in simulators, to achieve higher accuracy by training a network on relevant data. Enforcing fusion of the two state-of-the-art tools, the network is trained and tested on a dataset which was generated using CARLA (Car Learning to Act – an open simulator for Urban Driving). ResNet (He *et al.*, 2015a), Exception(Chollet, 2017) and VGG(Simonyan and Zisserman, 2015) were compared for choosing the best CNN. The compatibility of Exception was determined and was decided to be the best CNN for this fusion network.

A modern autonomous vehicle is equipped with multiple high resolution and low noise cameras, LiDARs and other sensors (Kesten *et al.*, 2019). While object detection using cameras has been around for a significant period, it lacks robust abilities in determining the bounding volume occupied by the same object. LiDAR on the other hand, has significant volume detection capabilities but has poor results in semantic mapping. This thesis implements an algorithm that involves fusion of camera and LiDAR data to classify objects and determine their bounding volume. The data of interest includes precise outputs from the left stereo camera and corresponding LiDARs mounted on a vehicle that capture activities on road. Previous attempts can be broadly classified into image based detection and LiDAR based detection. The

former method (Hegde and Zadeh, 2016; Qian *et al.*, 2020; Li *et al.*, 2019) takes in RGB images from cameras and estimates a 3D bounding volumes of objects in the image along with classifying these objects. LiDAR based methods compute volume of objects through mapping density of point cloud data (Zhou and Tuzel, 2018; Himmelsbach *et al.*, 2008; Beltrán *et al.*, 2018) and determines the class of an object by projecting the point cloud on a plane. CNNs have been used frequently in both these methods. As discussed previously, camera based methods were fruitful in examining 2D bounds of an object, thus classifying the object correctly while LiDAR based methods were suitable in predicting the volume of the objects but performed relatively poorly in determining object class using point clouds (Prokhorov, 2010). To accurately classify and detect volumes of objects, a robust algorithm is needed that assimilates various features of data from the cameras and LiDARs on the vehicle. Inspiration for this algorithm has been derived from observing effectiveness of multi-data as input to neural networks (Bindhi and Gupta, 2018). Here, individual feature maps are prepared for each data type and are fused together to achieve classification and bounding volume via regression. The important feature map identified is Camera Image (RGB) using point by point analysis. Extracting semantic information like object class and bounding box information like length, width, height, center x, center y and center z to facilitate 3D Object Detection are key features of this research. Through layers of convolution networks, regions of interest are identified and are finally evaluated to assess the outputs of choice.

1.8 Contribution and Scope of this Thesis

Preferably, the performance of AVs should align with human driving styles that reflect best driving practices. Crashes are likely inevitable as long as AVs share roads with other human drivers, cyclists and pedestrians. This thesis has been inspired by

recent and ongoing studies and surveys conducted by Dr. Kathryn A Johnson from Arizona State University (Johnson, K.A., Berman, S., Chiou, E., Pavlic, T.P., Cohen, A.B., 2020) (in preperation) to analyze the moral foundations (Graham *et al.*, 2013) and basic individual values (Schwartz, 1992; Schwartz *et al.*, 2012) influencing driver's crash responses obtained from the driving data of the subjects. This thesis work includes construction of a virtual world in a driving simulator where Autonomous Vehicles can be tested in an urban traffic environment. This world is designed with various scenarios to understand how would drivers respond to scenarios involving a school zone or trolley problem like situations. The data from the sensor suite of the simulated vehicle was logged during sample driving trials in all the scenarios. In future studies, this data will be collected from human subjects with different sets of moral priorities and values (as determined by surveys) and will be used to fit parameters of AV controllers, in an effort to characterize and mimic the variability in driving behaviors across different types of drivers. In order to implement the resulting AV controllers, object detection algorithms must be developed for classifying objects from AV sensor data, in order for the AV to respond appropriately based on its surroundings. A Sensor Fusion Approach for 3D Object Detection to classify and plot the 3D bounding box of objects in a scene from RGB camera and LiDAR data has been developed using transfer learning techniques on ResNet and Xception models available in the Keras library. These models were tested to evaluate their performance in the KITTI dataset (Geiger *et al.*, 2013a).

Chapter 2

LITERATURE REVIEW AND RELATED WORK

2.1 The Trolley Problem and Other Scenarios

These are hypothetical Scenarios which help us explore the moral and ethical inclinations of humans. The trolley problem has been described in chapter 1.4 of this thesis. Here we explore work that has been carried out to study the trolley problems.

In relevance to the field of Automated Vehicles there a been a change of perspective in the way the trolley problems have been viewed (Goodall, 2014a). The trolley problems for this research area are designed to resemble scenarios that we might encounter on the roads at some point of time. These scenarios have been constructed in a way to make subjects choose between 2 choices which decide the outcomes of a crash scenario. These scenarios involved situations where the subject who imagines himself to be in an unavoidable crash scenario has to choose between saving his own life by running into one or more pedestrians in his way or sacrificing himself by swerving into a wall. Other scenarios were set up under the same context in which the subject was asked to choose between a group of people on one side and only one person on the other side of the road mimicking the original trolley problem. Other complexities and parameters were allotted to the pedestrians such as age, gender, profession, whether he/she was a law-abiding citizen or not and so on. A problem similar to the trolley problem, the helmet problem has also been in discussions. In this setup the subject imagines himself to be the driver of the AV. He notices a vehicle coming towards him in the same lane, so he tries to brake, but the vehicle's brakes have failed. He can steer it to right or left to minimize damage or save himself.

But the catch here is on the left there is a Motorist who is not wearing a helmet and on the right is a Motorist wearing his helmet. The dilemma here is whether the subject/driver of the AV swerves to right to kill the motorist who has followed the rules and wore a helmet to be safe because his chances of survival are better or should the AV collide with the one on the left even if he is not wearing a helmet. The paradox is that if we choose the one on the right counting on his survival chances due to the helmet, we would be inflicting damage upon a person who follows the law and has worn the helmet to keep himself safe. But this collision happened because he was wearing his helmet and the one without helmet has been spared.

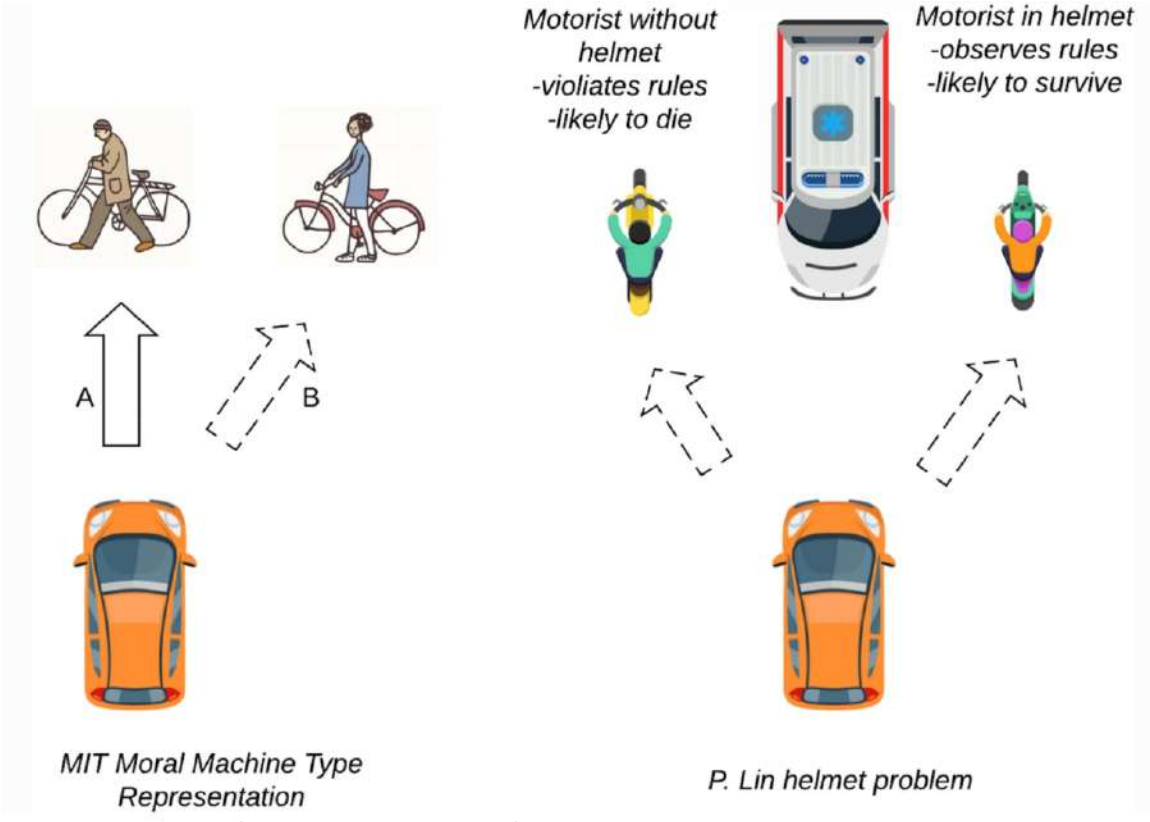


Figure 2.1: The Helmet Problem (Mihaly Heder, 2020)

Another important scenario that has been discussed widely in the context of moral

profiles of human drivers is the School Zone scenario (HANA, 2017; Michael Clamann, Nancy Pullen-Seufert, 2020). For the deployment of AVs, developers should make sure that the pedestrian detection algorithms are exceedingly accurate and should show capability in distinguishing between children, cyclists and adults (Michael Clamann, Nancy Pullen-Seufert, 2020). Letting AVs to be tested on the road at school zones can be extremely risky. The vehicles must realize when they enter or exit school zones and must abide by the respective speed regulations. Possible challenges in this scenario could be to pass by various adults and children to perform curbside pick-up and drop-off of the passengers. Communication between AVs and humans especially to determine the right-of-way would be complex to program. Navigation problems can also arise due to the terrain and infrastructure. Automated vehicles tests must involve school zone scenarios but they should not be allowed at the school zones if they do not exhibit the accuracy standards. Therefore for the purpose of testing school zone scenarios, CARLA was used to simulate such environments to understand and log the behavior of various human drivers when navigating through a school zone.

2.2 Human Driving Styles

With augmented deployment of AVs, the question that arises is how should the AVs drive. Apart from safety and reliability, providing great comfort and a satisfactory user experience will also play an important role in the acceptance of these vehicles (Kuderer *et al.*, 2015). Comfort and user experience can have multiple meanings here. Some people might prefer a peaceful and steady drive whereas others might prefer high speed drives (within the speed limits). This fact leads us to a conclusion that AVs must be available in different driving styles to serve people's preferences.

The three distinctly identified driving styles in this context are Power Driving, Benevolent Driving, Defensive Driving (Basu *et al.*, 2017). For the scope of this

thesis only Power and Benevolent Driving Styles are considered. Factors like clean driving record, mean distance to lead car, distance maintained during lane change and merges, braking distance, maximum and mean speeds at turns, speeds through school zones and crash decisions can help determine the moral profile of a driver and categorize them as a Benevolent or a Power Driver. A benevolent driver is someone who follows the rules and a power driver is someone who is a little careless when it comes to hard and fast rules. In the context of trolley problem dilemmas adapted from the philosophical studies, the person who sacrifices himself for saving others would be categorized as a benevolent driver and the person who is willing to run over the pedestrians to save his own life is classified as an power driver. This kind of classification can help diversify the AV experience and make them more compatible to the societal needs. Dr. Kathryn Johnson, in her work Johnson, K.A., Berman, S., Chiou, E., Pavlic, T.P., Cohen, A.B. (2020)(in preperation) has extracted important moral factors that can be leveraged to assess a person's driving behavior or moral profile in the context of human driving styles.

This thesis aligns with the research proposed by Dr.Kathryn Johnson. According to the proposed research, it was important to understand the moral profiles and priorities of participants who are taking the driving simulation tests. In comparison to the MFT survey which hypothesizes 5 moral domains of Harm/Care, Fairness/Reciprocity, Ingroup/Loyalty, Authority/Respect and Purity/Sanctity; Basic Individual Values proposed by Schwartz *et al.* (2012) seemed more suitable. For the proposed research, the 57-item Schwartz Value Survey (SVS) (Schwartz, 1992) will be used. From these 57 questions we will be able to assess an individual's inclination towards values such as self-direction, stimulation, hedonism, achievement, power, security, conformity, tradition, benevolence and universalism.

For the purpose of this research it has been determined that Power and Benev-

olence are the 2 most predominant values that influence a person's driving behavior during crash scenarios (Johnson, K.A., Berman, S., Chiou, E., Pavlic, T.P., Cohen, A.B., 2020) (in preperation). These questionnaires in conjunction with the driving simulation results will be used to fit models to design AV controllers which would presumably align with the moral, ethics and values of the good drivers with clean driving records.

2.3 Existing Crash Simulators

MIT Moral Machine (Awad *et al.*, 2018), is the most popular platform that records responses to dilemmas. It was developed by Iyad Rahwan and his team at Massachusetts Institute of Technology. The experiment consists of hypothetical dilemma situations like the ones discussed in Chapter 2.1. These scenarios have been derived from the trolley problem as well. The Moral Machine generates such dilemmas randomly and records the responses that subjects make between two hazardous outcomes. The website, Moral Machine (2016), gathered more than 40 million decisions in ten languages from millions of people in 233 countries and territories. More results and statistics about the experiments have been provided in Awad *et al.* (2018).

The results of this experiment have been made available at Moral Machine Results (2018) and the summary of it given in Figure 2.4. The figure explains the statistics of the responses of all the subjects towards each category in the Moral Machine Experiment.

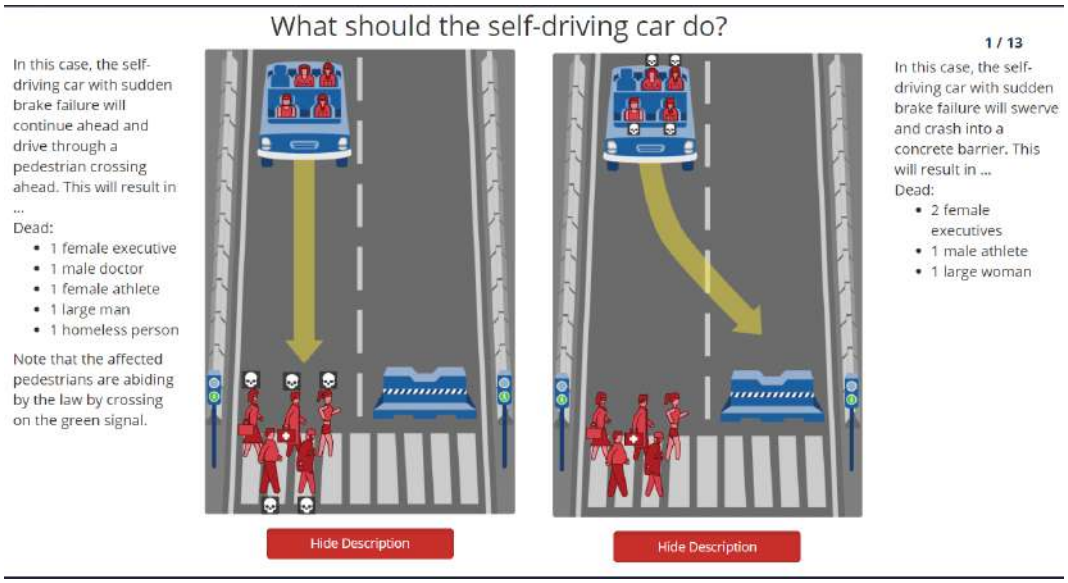


Figure 2.2: MIT Moral Machine Example Scenario 1 (from surveys in Moral Machine (2016))

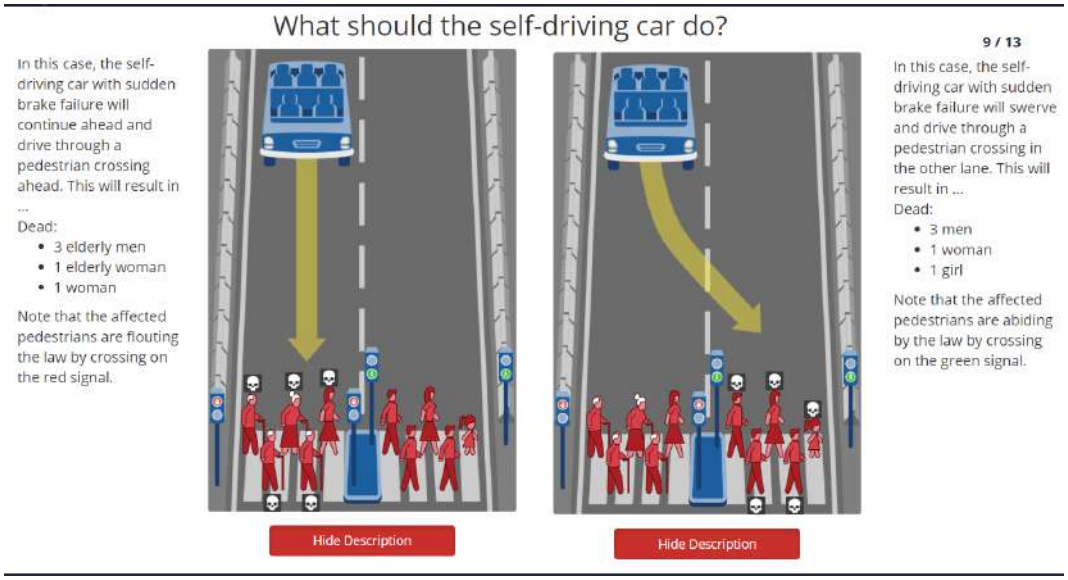


Figure 2.3: MIT Moral Machine Example Scenario 2 (from surveys in Moral Machine (2016))

TrolleyMod v1.0 (James Minton, Vahid Behzadan, 2018) is an open source data collection platform for ethical decision making in autonomous vehicles (See figure 2.5) developed by a team at the Kansas State University. This work has similar motivation

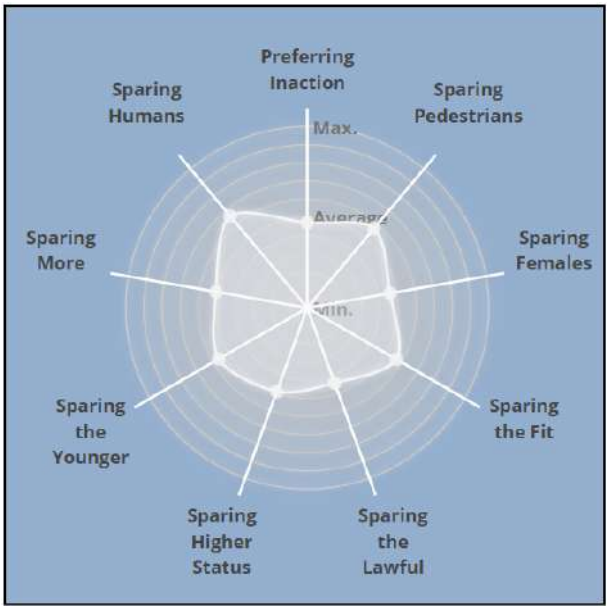


Figure 2.4: MIT Moral Machine Results - Statistics (Moral Machine Results (2018))

to this thesis but is limited to observing who the subject crashes in given dilemma situations. The system generates random trolley problem like scenarios obligating the user to crash into at least one of the two choices. The main issues with this simulator is that it uses outdated versions of CARLA and Unreal Engine for which support and updates are not provided as of now. This project has not been updated since December 2018. It is not very immersive or realistic because the participants are not given enough time to understand the environment or interface but are directly tested to log the responses.

2.4 Object Detection and Transfer Learning

Image based 3D object detection : An RGB image in its simplest form is a planar representation of a 3D space. Naturally, this image lacks vital depth information to estimate the volume enclosed by object in the scene. More importantly, without accurate volume, determining the coordinate center and orientation of objects

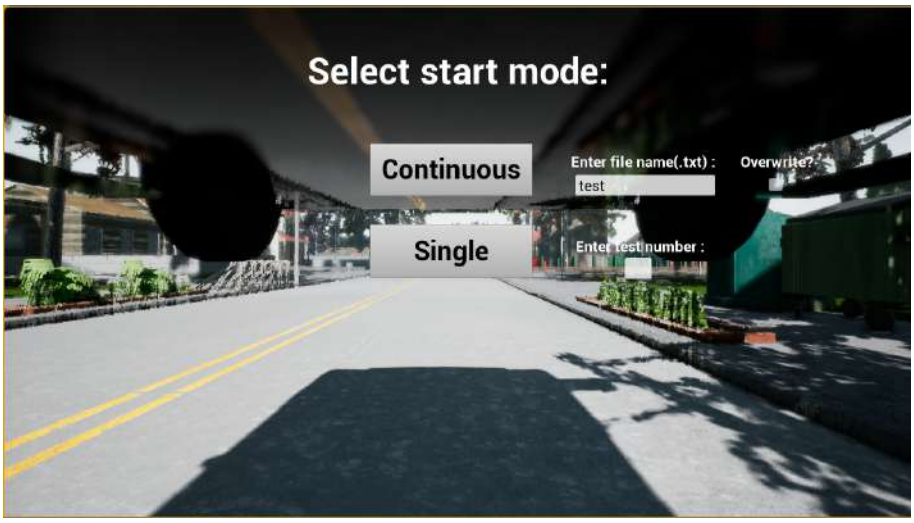


Figure 2.5: TrolleyMod v1.0 (James Minton, Vahid Behzadan, 2018)

is challenging as well as unreliable. However, a representation of depth can be achieved by generating point clouds with the help of inexpensive tools or using images of the same scene in different orientations. Sliding Shape (Song and Xiao, 2014) approach uses RGB images together with depth images to generate bounding boxes around objects.

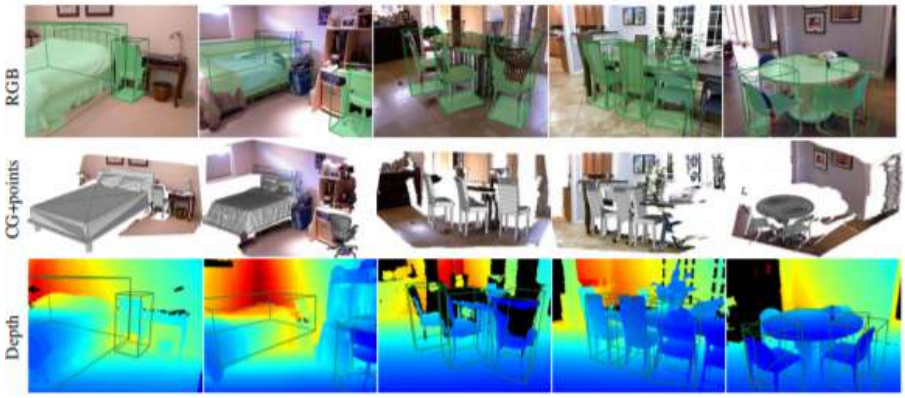


Figure 2.6: Results From Sliding Shape Approach (Song and Xiao, 2014)

A depth image is an RGB image where distance between the object and camera plane is encoded. This eliminates computationally intensive parameters like sensor noise, texture clutter and illumination. Because depth is encoded in an RGB image,

computational cost of analysing 3D point space is saved. Stereo-RCNN (Li *et al.*, 2019) approach takes stereo images as input and simultaneously detects and classifies objects in the images. This method takes advantage of information encoded in stereo images to estimate object dimensions by analyzing the Region of Interest (RoI)s. In monocular approach (Girshick, 2015; Chen *et al.*, 2015), a set of lenses are used to magnify distant objects and project them on 2D plane. Primarily, this method is used for detecting objects in 2D plane and is accurate in estimating the region encompassed by the object in the plane, but lacking depth information creates challenge in predicting the actual volume i.e. 3D bounding box of the object. Although this method is faster compared to various other complicated methods and can also exhibit real-time performance (Song and Chandraker, 2014), it is restricted by the performance of cameras. Since, cameras are the sole generators of data in this case, data is lost in projecting the 2D image in 3D space. Object detection algorithms use prior knowledge about the features of the object and try to identify similar objects. However, deficiencies in learning, e.g. new or unique object can challenge the understanding of the algorithm. Cameras are also known to be prone to environmental factors like temperature, humidity, light intensity and thus using data from only such cameras can produce corrupted data as well as inconsistent data involving same objects captured in different conditions. RGB images in general, lack information about depth (Chen *et al.*, 2016a) and thus are not suitable as a robust approach to 3D object detection and classification.

Recent works like (Ma *et al.*, 2019) show that RGB images can be projected in point cloud space and complicated feature detectors can be used to predict the location and orientation of objects. However, this approach is restricted by visibility of object as well as the data loss in projections over the process. In this context, we aspire to exploit the capabilities of LiDAR information to generate regions of interest in

Figure 2.7: Results from Monocular 3D Object Detection for Autonomous Vehicles (Chen *et al.*, 2016a)

determining the depth of the objects.

Point cloud based 3D object detection : LiDARs can be a huge resource in identifying objects by comparing density and intensity of points in the point cloud (Li, 2017). Since LiDARs generate huge point clouds on the order of 1.5 million points, it can be computationally expensive to monitor these clouds and determine object location and orientation. 2D image analysis has greatly advance since its inception and thus has a wide range of applications in autonomous driving. It is possible to project LiDAR data in 2D space (Su *et al.*, 2015) and the method is known as Point Cloud Projection. Most important advantage of this approach is that it reduces the computation space and thus decreases time complexity. However, front view projections of LiDAR data are prone to data loss (Li *et al.*, 2016) as the sparse data in volume may appear dense in the projection. This may trick the algorithm into inaccurately predicting the object class and thus is a trade-off in return of high speed of computation. Regions of interest (ROI) is a technique (Chen *et al.*, 2016b) used to discard the irrelevant point clouds and focus on only specific spaces in the

cloud for relevant information. However, semantic projection of LiDAR data does

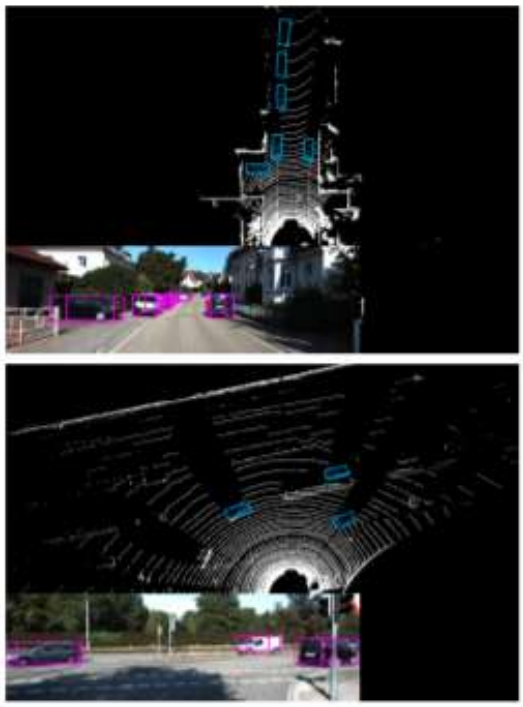


Figure 2.8: Results from Multi-View 3D Object Detection Network for Autonomous Driving (Chen *et al.*, 2016b)

not have comparable accuracy of a camera based RGB image analysis approach to classify and bound object in 2D space. These proposals are then used to evaluate the same object in the LiDAR Front View data for accurate depth analysis. Similar proposals are used to evaluate object location in the image to reduce computation time.

TensorFlow and Transfer Learning: Tensorflow (Abadi *et al.*, 2016) is an open source library for Machine Learning, Deep Neural Networks and complex computation using data-flow graphs. It is compatible with Python, C++ and Java. Tensorflow was first developed by Google Brain Team for internal research applications. Its version 1.0, released in February 2017, has seen exponential demand through github forks. It runs on diverse platforms like GPUs, TPUs and can also run on mo-

bile phones or micro controllers. Tensorflow is generally used to build a computation graph as a data structure for building the models and compiling them. Keras (Ketkar, 2017) acts as an interface for Tensorflow. It provides the building blocks to build Deep Neural Networks on the tensorflow framework. It is the library which contains classes and functions for Optimizers, Activation function and metrics. Both Tensorflow and Keras are provided through github. Tensorflow's applications include a wide range of research domains like data science, natural language processing, robotics etc.

Another attractive attribute of the Keras library is that it enables Transfer learning on the existing Deep Neural Network models which have been pre-trained on ImageNet (Dai *et al.*, 2007). ImageNet is a large dataset of more than 14 million hand annotated images mentioning the objects in them and with at least 1 million of them containing bounding box information of the images (Deng *et al.*, 2009). ImageNet is the largest collection of images available with more than 20000 classes. Transfer learning is a technique in which pre-trained models are frozen using tensorflow and are reused for applications in related but different contexts. Keras provides us with such pre-trained models like ResNet, VGG19, Inception, Xception, NasNET etc. The feature extraction capabilities of these pre-trained networks are used to develop intermediate model outputs for the data in our application thus facilitating transfer learning.

Chapter 3

METHODOLOGY AND APPROACH

3.1 Building the Simulator

The CARLA Simulator for this application needs to be built from source. This process has been clearly documented on the official CARLA website (CARLA- Car Learning to Act, 2017). Building it from source will enables us to open the simulator in the Unreal Engine as an Unreal Project as opposed to Standalone game mode when using the pre-compiled version.

Operating System	Ubuntu 18.04 LTS, 64-bit
RAM	16 GB
Memory	500GB HDD
GPU	Nvidia GeForce GTX 1050 Ti 4GB with NVIDIA drivers (390, proprietory)
Processor	Quad-core Intel i5-3550 3.3GHz

Table 3.1: Hardware Specifications of the Host Computer Used for this Application

Upon successful build the editor can be launched from its root directory. The user can choose an existing base project with dedicated blueprints or he/she can choose a blank project. Each map or virtual environment or game is called a 'Level'. Opening a project opens the main editor whose default layout is as follows

For panning, zooming and moving around the input is Keyboard keys WASD, arrow keys and Mouse. The Content Browser is place where all the assets related to the project can be created, edited, migrated or imported. The details pane is the

Figure 3.1: Deafault Layout for a Blank Project 1. Tab Bar, 2. Toolbar, 3. Modes, 4. Content Browser, 5. Viewport, 6. World Outliner, 7. Details (Unreal Engine Documentation, 2014)

interface which provides us a list of all editable properties and their current values. The world outliner shows information of all the actors or blueprints used in the current level.

CARLA can be built on this compiled version of Unreal Engine through git clone and make commands in the Terminal. This will download the latest assets and updates provided by CARLA. Once this process has been completed successfully, CARLA can be opened using the command '**make launch**' from the root directory through the Terminal. This will open up the CARLAUE4 Project in Unreal Engine. Press the Play button on the toolbar to simulate the level or the default map (Town03).

CARLA functions on 2 different modules the server and the client. The server refers to the simulator itself and the client refers to the PythonAPI module. Most of the processing like rendering of actors and sensors, lighting, logic, simulating

Figure 3.2: Screenshot of Town03 in CARLA Simulator

physics and shadows takes place on the server side. The client communicates using Python language. Almost everything relating to controlling, spawning and destroying actors and sensors, controlling the weather, logging sensor data etc. can be done through Python scripts. Both CARLA build-from-source and pre-compiled versions come with example scripts. Some of the most useful example scripts are manual_control_steeringwheel.py which enables one to use a steering wheel to control the vehicle in a pygame setup, spawn_npc.py to spawn vehicles and pedestrians.

3.2 RoadRunner for Customizing Maps

RoadRunner is a third-party software originally created by VectorZero but was later acquired by Mathworks. Hence to use RoadRunner one must have an active Mathworks subscription. RoadRunner is a comprehensive tool for creation of 3D scenes and Maps for testing AVs. Its in-built assets from the RoadRunner Asset

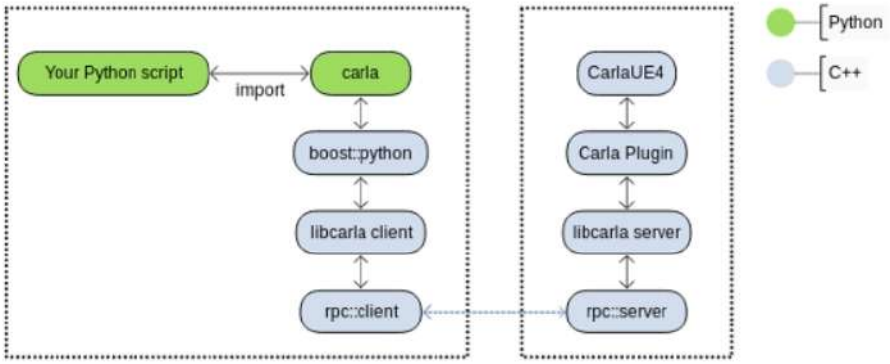


Figure 3.3: CARLA Server-Client Build System (CARLA- Car Learning to Act, 2017)

Library will enable us to build a map or a virtual world very similar to urban cities. Some of its most useful assets include buildings, traffic lights, various road styles, markings, speed signs etc. RoadRunner also enables us to export the created map into file formats compatible with Unreal Engine and CARLA. The work flow around RoadRunner is relatively intuitive and there are enough resources available online to guide you through the software.

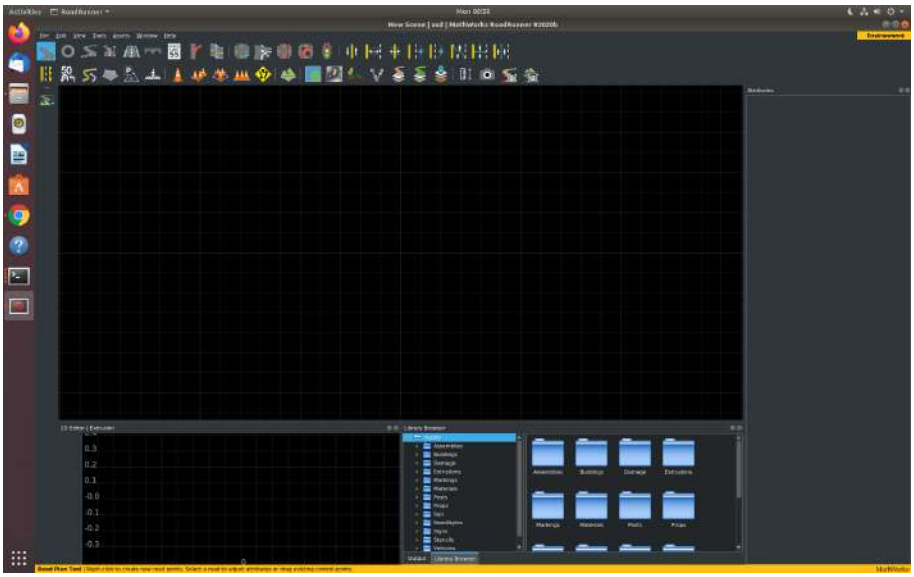


Figure 3.4: Screenshot of RoadRunner - Home

We can create road and intersections seamlessly by using the road plan tool.

There are options available to create hills and valleys too. Any number of lanes can be added, merged and modified using the lane chop, lane joining or lane marking tool. Road navigation and traffic flow is described by the Maneuver tool. This tool helps in specifying signal phases which describe the traffic flow across a junction. In the figure below the 2D editor on the left bottom shows the differnt phases for the 4-way Protected left case. We can also set time limit for each phase and this is what decides the time limit for change in traffic lights. Once this is done we can import traffic lights from the assets and place them around the corners.

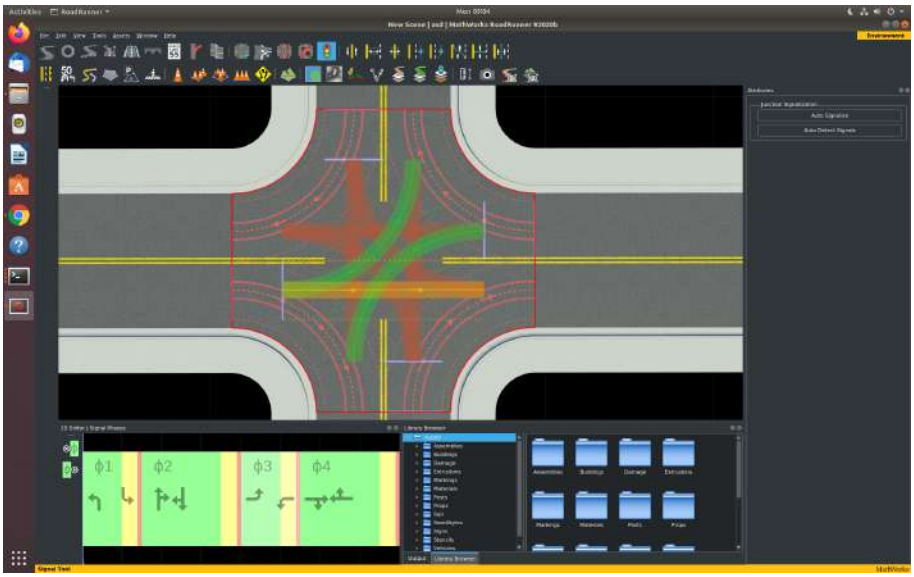


Figure 3.5: Screenshot of RoadRunner - Simple Junction and Maneuver Tool for Designing Signal Phases

Now from here we place the traffic lights from the Asset Browser on bottom right by dragging and dropping it in the workspace. All of this can also be done through the Auto-Signalize option from the attributes panel on the right. Now we select the Crosswalk tool and select the particular marking style from the assets and right click on the junction we want the markings for. This will create the cross walks for Pedestrians as shown below. A lot of other commands are available for creation of complex maps and this can be referred from the RoadRunner Documentation on the

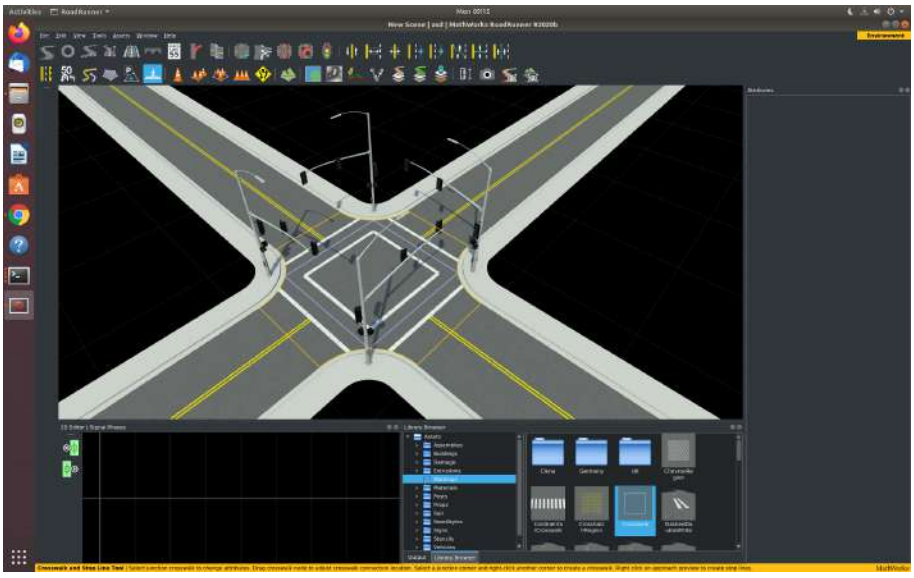


Figure 3.6: Screenshot of RoadRunner - Signal Tool and Crosswalk Tool

Mathworks website (Mathworks, 2019). After this the map can be previewed for lidar point cloud map or opendrive preview or scene export preview to check if everything is right. After this we can export the map created into .xodr (Opendrive), .fbx (filmbox) and .xml file for clean import into CARLA simualator. The Opendrive file is an editable text file that parses the information of the roads, signals, georeference latitudes, longitudes useful to convert GNSS sensor measurement to world coordinates and pedestrian navigation paths. If there are no errors the files will be exported to the set destination folder. Now we will have to download RoadRunner Plugins for CARLA and Unreal Engine to enable importing of these maps. We place the Plugin folders inside the Plugins folder in CARLA root and rebuild CARLA to compile these Plugins. Now we are ready to ingest a map into CARLA. The image below shows a test map in which a few buildings have been imported.

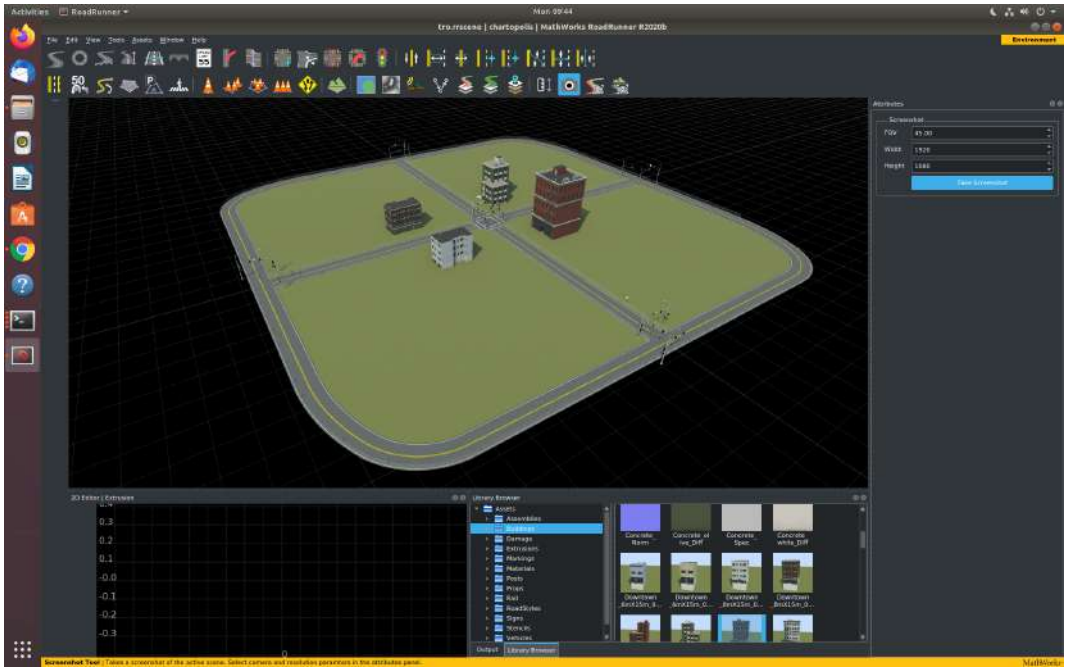


Figure 3.7: Screenshot of RoadRunner - Test Map

3.3 Map Ingestion into Simulator

Once we have the exported files from RoadRunner we place the exported OpenDrive and Filmbox files in the Import folder present CARLA root folder. If the filmbox file is being created in a different software like Maya (Autodesk) it has to follow a certain nomenclature for roads, sidewalks,crosswalks and grass as mentioned in CARLA- Car Learning to Act (2017).

3.3.1 Importing the Map

Importing this map is a simple process for the user but a lot of computation and processing happens behind the scenes. The names of the exported files or maps should not be changed throughout this process otherwise the import will fail. The next step is to open a Terminal in the root folder and run the command '**make import**'. This will import the map and rebuild it from the source. The map will now be available in the Content browser in the Unreal Engine under the map_package folder

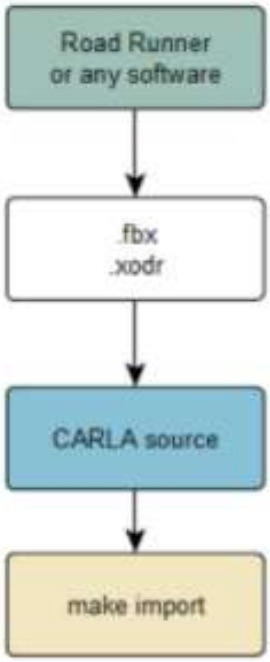


Figure 3.8: Ingesting the map - Using Source CARLA- Car Learning to Act (2017)

3.3.2 Pedestrian Navigation

The imported map does not necessarily have the information about where the pedestrians can walk. This is given by a binary file which has to be created separately and copied to the 'Nav' folder inside the Map Package that has been created from running make import. For the imported maps Pedestrian Navigation information has to be derived through RecastBuilder which is also available in the source code of CARLA. We have to now import .obj file of the map from either unreal or roadrunner. The Opendrive (.xodr) file and the object (.obj) files are copied to Carla/Util/Docker/dist folder and the command **'build.sh map_name** is run to let the RecastBuilder create the binary file for navigation which is then placed in the 'Nav' folder in the package. The map has to be rebuilt again following the import procedure mentioned above. We can now open the map by double clicking on the

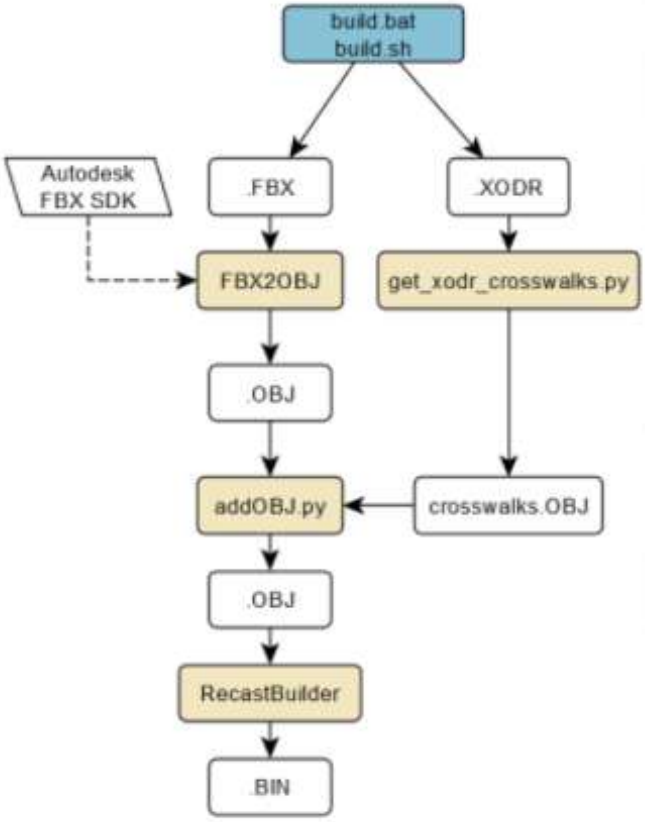


Figure 3.9: Creating Pedestrian Navigation CARLA- Car Learning to Act (2017)

map_name.umap file from the content browser.

Figure 3.10: Test Map

Figure 3.11: Custom Map 1

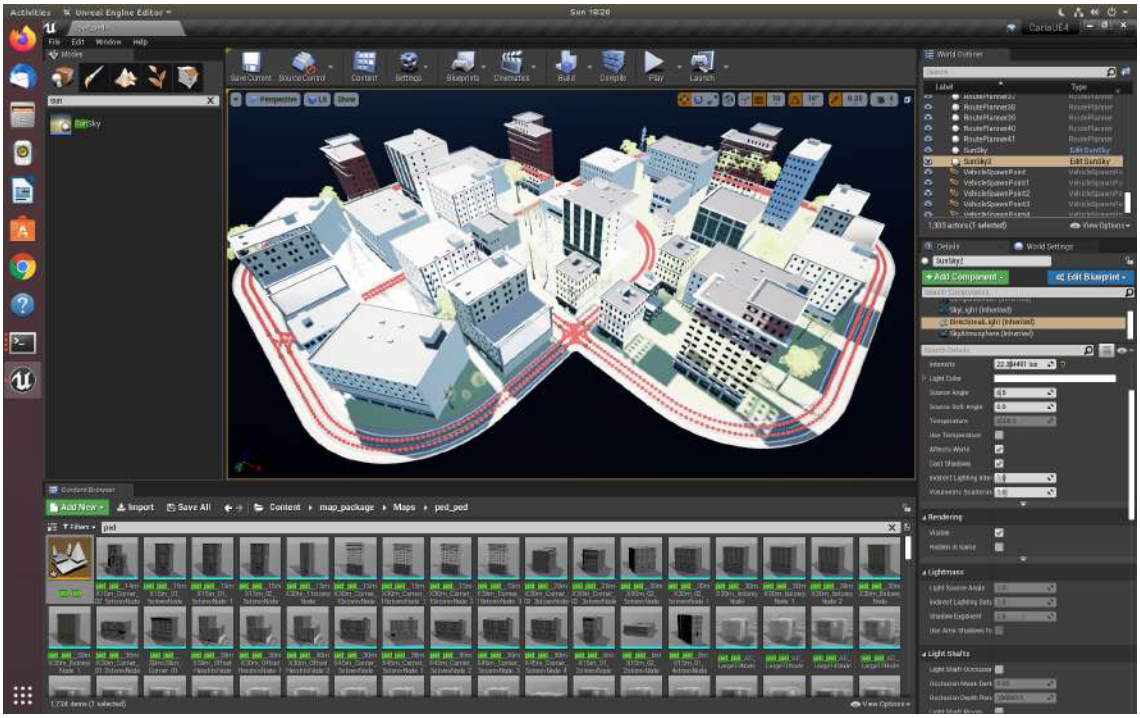


Figure 3.12: Custom Map 2

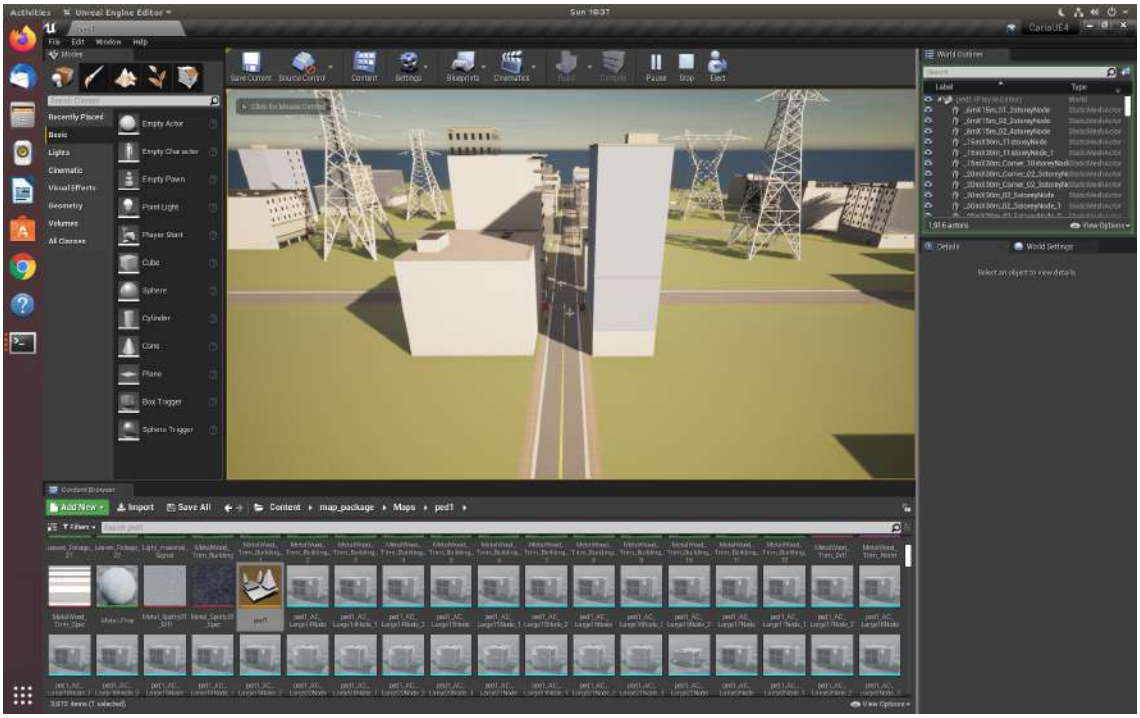


Figure 3.13: Custom Map 1 - Closer View

3.4 Creating Scenarios in CARLA

We use the existing skeletal meshes and animations to create blueprints and spawn these actors instantaneously when necessary. The Blueprints are mainly used to impart animations like walking, idle stance and running to the actors. Nodes like 'AI Move To', 'Spawn Actor', 'Destroy Actor', 'On Event Hit' etc. have been very useful to manipulate the simulation in a way as to cover many scenarios in the same map. As described in Chapters 1 and 2 we have chosen a total of 4 scenarios. One scenario in which the driver has to drive through the school zone and the other 3 involved strategically placing actors to recreate trolley problem like dilemmas. These dilemma scenarios were placed around the corner and the driver will be directed towards this scene and will encounter a sudden crash scenario which would give him very little time to respond.

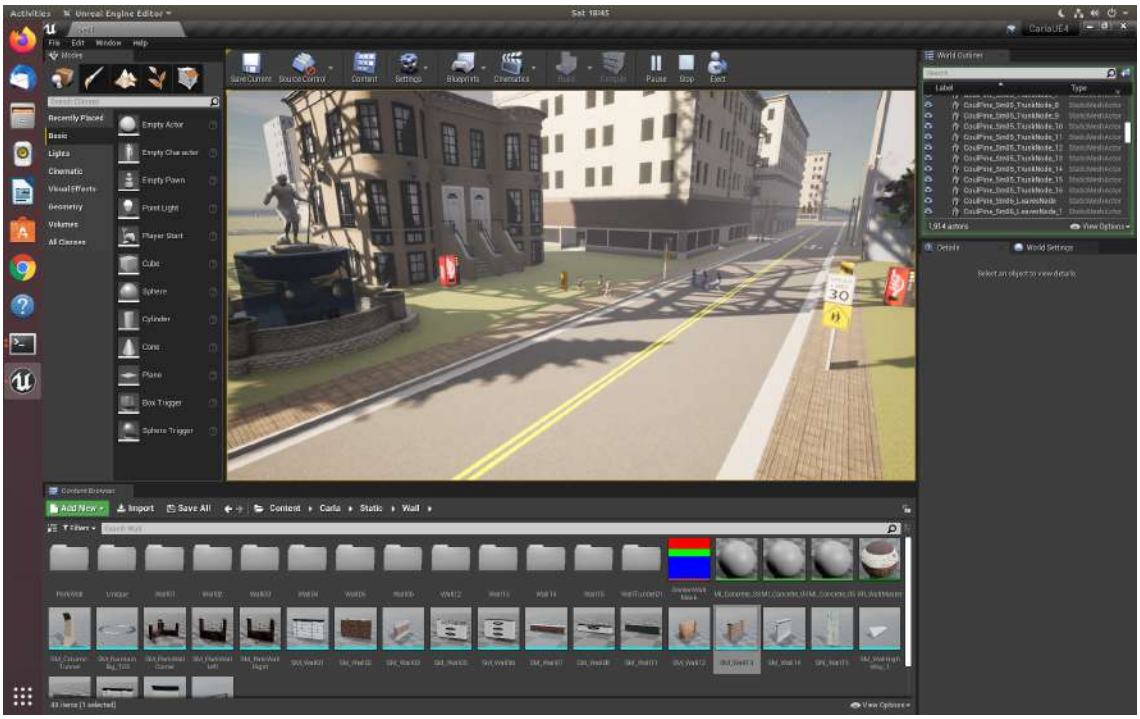


Figure 3.14: Scenario 1

Figure 3.15: Scenario 2

Figure 3.16: Scenario 3

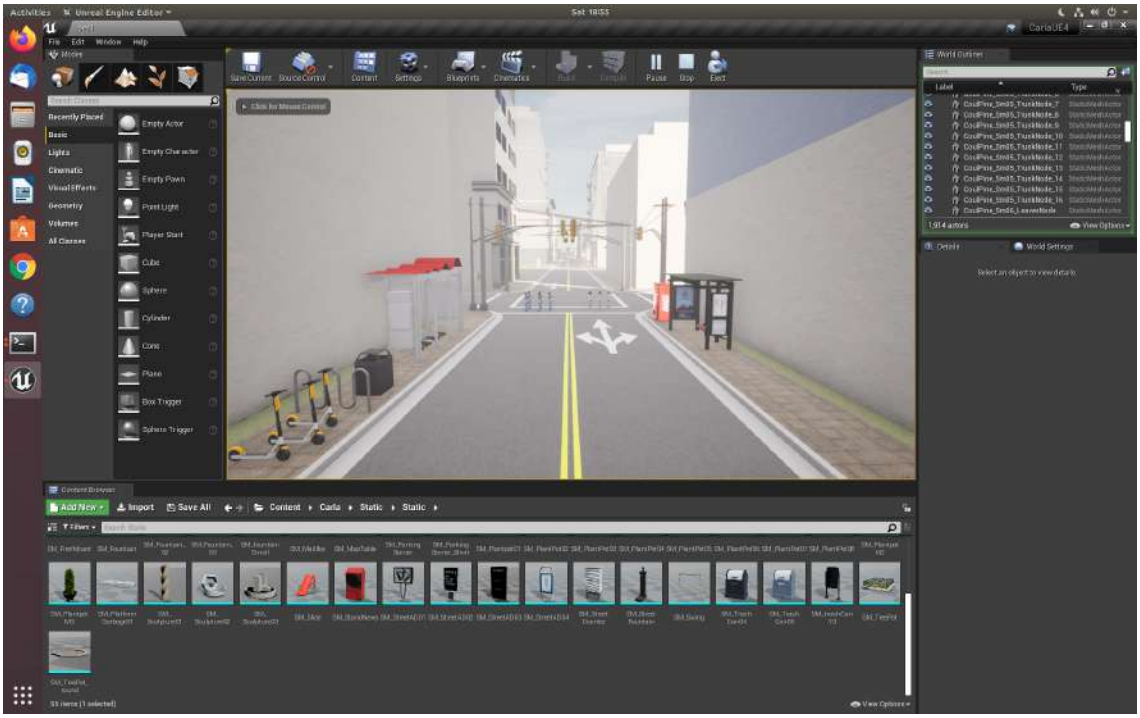


Figure 3.17: Scenario 4

3.5 Data Collection

Sensors CARLA has a wide range of in built sensors blueprints that can be attached to a vehicle before starting the simulation. Some of the main sensors are Cameras, LiDARs, RADARs, IMU, GNSS. All the sensor classes have a listen() funtion which enables us to record the data. The main challenge here is to synchronize all the sensors to output a data at fixed time steps so as to enable sensor fusion. Using python from the client side we can spawn actors and attach sensors to the actors/vehicles and record data from those sensors. We plan to collect these sensor's data from human participants with different Values Profiles who drive through the scenarios created in the simulator and then fit parameters of AV-implementable controllers to these data. This data will then be processed to convert the ourputs into latent variable which in the future will be used by SEM framework to implement Path Analysis

3.6 Object Detection Algorithms

In this section we dissect the architecture of 3D Fusion Network that classifies and predicts 3D bounding box regression by extracting information from RGB images captured by camera and raw point clouds captured by LiDAR on-board an autonomous vehicle.

3.6.1 PointNet

PointNet (Qi *et al.*, 2016) is a unified network that is invariant to N data points in a cloud. This allows the input to PointNet to be in a random order which is convenient compared to systematic voxels and grids. Thus, irrespective of order of points fed to the network, the analysis of the network remains unchanged. This also acknowledges that $N!$ permutations of N data points are accommodated in the network.

Points in a cloud are usually encoded with distance metric. While the size of point is not available, its distance from a defined origin, helps distinguish it from a subset of other points. This signifies that points in a cloud are not in practice, random, but hold meaningful information in the form of distance. A collection of neighbouring points can be used to define an object in space. PointNet also features analysis of group of local or neighbouring points used to identify an object. Additionally, it can analyze all combinations of K neighbouring points which are a subset of N.
Invariance of analysis due to transformation is an important consideration while choosing an algorithm. It is important that while the points undergo an affine transform with a combination of translation and rotation individually or together, the network does not vary its output. PointNet features this invariance under transfor-

mation and hence suits the application in this research.

3.6.2 Image Analysis

Processing of 2D RGB images is equivalently important for this research as it provides the ground truths of locations of object and class identification in image space. It has been established that object detection and classification in image space is robust and reliable. Thus choosing an algorithm that takes RGB images as an input and provide bounding box and class of object as output is important to estimate viability of our research. Key indices considered for choosing an image analysis algorithm were time-space complexity, ease of integration and its overall performance in fusion network accuracy. Xception, Inception, ResNet152 and VGG19 are discussed briefly in this context. Results obtained by using these architectures and comments about them have been discussed in the later sections.

ResNet: Deep networks have been utilized in many computer vision applications. However, training a deep network is difficult and challenging. The deeper the network is, the higher the probability of the model to be overfitted. But, ResNet (He *et al.*, 2015b) architecture, introduced by Microsoft, avoids this hassle by reiterating the layers of a deep network with a learning residual function as input. This reiteration stems from counter-intuitive phenomena of degradation problem where additional layers, when developed as identity mapping, reduces the errors of deep network equal to the errors of its corresponding shallow network.The ResNet building block can be represented as,

$$y = F(x, W_i) + x \tag{3.1}$$

where x and y are input and output vectors and $F(x, W_i)$ represents residual learning

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Figure 3.18: Comparison of Various CNNs

function. Ideally the addition of F and x happens using shortcut connections and element wise addition. This shortcut helps skip the many cluttered layers and speed up the process. The main function of this shortcut is to join linear functions in a deep network and skip the non-linear functions. It is observed that the performance of ResNet compared to traditional deep networks is very similar, except that ResNet outperforms other deep networks in space-time complexity.

The most common versions of ResNet are ResNet50, ResNet101 and ResNet152. The input to any ResNet network is an image of fixed size of 224×224. Our research features the ResNet152 version of the algorithm. Here, the 2 3×3 layers are replaced by 3 new layers. These layers are ordered as 1×1, 3×3 and 1×1. Reducing dimensions by introducing 1×1 layers significantly reduce errors. It is observed that the complexity is not altered by this modification and the efficiency is raised. Hence, the use of ResNet152 is justified.

VGG19: A refined version of convolution nets is featured in the Visual Geometry Group (VGG) (Simonyan and Zisserman, 2014) architecture. This architecture is known to provide state-of-the-art accuracy on a wide range of datasets. Minimal

pre-processing is required which is subtracting mean RGB value from each pixel of an image. The processed image is passed through a set of convolution layers of size 3×3 and 1×1. While VGG is a predecessor of ResNet, the occasional dimensional reduction by 1×1 layers helps it reduce errors and maintain high efficiency. This architecture proves that higher orders of first convolution layer featured in (Simonyan and Zisserman, 2014) i.e. 11×11 and 7×7 do not necessarily increase accuracy is tantamount to high computation costs. Using a combination of 3×3 and 1×1 convolution layers achieves similar or higher accuracy at reduced complexity helps VGG stand amongst competitors.

The input to a VGG network is an image of fixed size of 224×224. To achieve this, the network reshapes an image to maintain the dimension constraint. Furthermore, random flipping and RGB colour shift is performed to enable data augmentation and raise efficiency of training. VGG is tested on classifying images in dataset distributed across 1000 categories and resulted in high accuracy. Use of VGG allows us to validate our fusion network with a traditional element. VGGs are advanced than the complex conventional deep networks because it features reduced bottlenecks but at the same time, it holds its deep network intact as compared to ResNet that completely eradicates non-linearity from the model. This allows us to study the effect of non-linear elements in a mode on autonomous driving applications and hence, the use of VGG to create a variant of our fusion network is justified.

Inception and Xception: The Inception (Szegedy *et al.*, 2014, 2015) micro-architecture, formerly known as the GoogLeNet is a multi-level feature extraction Network. Depending upon the version rolled out by Google it is currently called Inception V3 and it is a model that is available through Keras. It computes 1×1, 3×3 and 5×5 convolutions in the network. Before they are sent to the next layer the outputs from these convolutions are horizontally stacked. The developers of

this network were more focused on cutting down the computational costs and thus expanded 'wider' rather than 'deeper'. The Inception Network proposed in Szegedy *et al.* (2014) was remodeled into Inception V3 mentioned in Szegedy *et al.* (2015) to improve accuracy on the ImageNet.

Xception (Chollet, 2017), also referred to as Extreme Inception is an extension to the Inception network which involves a flow where the convolutions are separated for each deep layer. It was developed by the creator of Keras, Francois Chollet. As the number of classes in the dataset increase, the performance of Xception increases significantly in comparison with Inception V3.

3.6.3 Fusion Methodology

. The network features 3 important parts namely:

CNN which is pre-trained is used to extract features from input images. VGG19, Inception, ResNet and Xception were tested on the algorithms to compare whose performance was the best, the criteria being speed, accuracy, optimization and efficiency. The CNN was chosen to implement transfer learning on the network used for this application. These Nets are pre-trained on ImageNet Deng *et al.* (2009) and thus we use techniques of Transfer learning to train the model using these pre-trained nets directly from 'keras'. **Fusion architecture** performs the task of combining features extracted from individual networks and regresses the 3D bounding box and determines the class of the objects.

LiDAR representation: Point cloud is a collection of photons collected by the hardware at various instances that when scattered represents a 3D space. Analysing a point cloud is generally intensive because of the random nature of the points. Researchers have overcome this by aligning the points into grids or collection of images. However, this increases unnecessary computation costs while not increasing efficiency

significantly. The biggest issue with utilizing point clouds for deep learning is that they are unordered datasets. PointNets (Qi *et al.*, 2016) are invariant to the order and transformations of point clouds as they are built using symmetric functions. PointNet accepts the raw LiDAR file without the need to align the cloud and thus retains valuable information otherwise lost in transformations. The input to PointNet requires minimal preprocessing which is mostly determining an affine transformation matrix and multiplying it to all points in the cloud. This matrix calculated from sensor calibration file corresponding to a point cloud of a scene. By transforming, we aim to achieve points scattered in random orientation to a collection in sensor coordinates. This serves as a datum to process all point clouds knowing that all data is processed in a uniform geometric system and thus errors due to orientation is minimized.

Point Cloud Analyzer is used to skim through raw LiDAR bin files to extract a random sample of 2048 points in relevance to the scene. A 3D Lidar essentially collects millions of points per frame from the scene. Processing all these points is unnecessary and computationally costly. To extract points relevant to the scene camera calibration and camera's field of view are used to narrow down the domain to points that fall in the scope of the camera. PointNet (Qi *et al.*, 2016) is the network that analyzes points in a cloud individually. This helps the network encode the criteria for selecting particular points that describe a shape or a feature. A random sample of 2048 such points is derived from the output of data preprocessing.

As shown in the figure 3.19, Features extracted from LiDAR and camera's RGB images are stacked and are passed through multi-layer perceptron to determine the class and regress the bounding box of the object

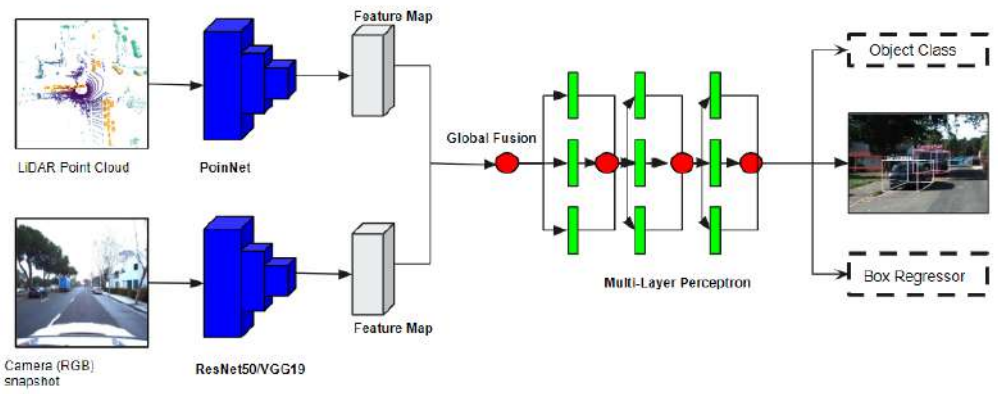


Figure 3.19: Proposed Network for 3D Object Detection and Classification

3.6.4 Datasets Used

As can be referred from above this thesis emphasizes on testing the Autonomous Driving styles in both virtual and physical domains. Both the domains have their inconsistencies like the robots used (Go-Charts) would not resemble vehicle dynamics of a real world car but to investigate this concept it is essential to test it in a physical environment. In order to conduct tests in a wide variety of scenarios it has to be tested in CARLA too. Although transfer learning has been a novel discovery it has proven its efficiency only the 2D Object Detection domain. Using the same model for 3D Object detection in both physical and virtual domains has given very poor results. Hence we train models with the same algorithms on 2 different datasets one from each of the physical and virtual domains.

Experiments were performed on the KITTI Vision Benchmark SuiteGeiger *et al.* (2013b) for obtaining models for the real world. KITTI is an opensource dataset that aims to enhance computer vision applications in autonomous driving. The dataset features 7481 training scenes and 7518 test scenes where the package contains the same number of multi-camera captured images and LiDAR files. While using the dataset, we only use images captured by left-end camera and corresponding LiDAR

point cloud. Alongside the raw dataset, sensor calibration files are provided that gives the location of the sensors (camera and LiDAR) with respect to the vehicle at all instants. Using this calibration file, we convert LiDAR point cloud data randomly distributed in global frame to sensor frame by multiplying all points with an affine transform. Using the location of camera, we also transform the corresponding RGB image. We observed that the effect of this transformation on the image is no more than a minor rotation. No manipulation on the illumination, noise and overall quality of the raw data is performed. This prepares the raw data for training while retaining most of its original features.

As a proof of concept, Data from CARLA (Version 0.8.4) has been generated inspired by the works from Brekke *et al.* (2019). The data has been saved in the KITTI Format so as to enable using the same algorithms for both the datasets to train the Neural Networks. Inspired from the works presented in Brekke *et al.* (2019) a training dataset similar to the KITTI dataset was created manually from CARLA. The size of this dataset is same as that of KITTI with 7481 images and corresponding point cloud and labels. This data was collected alternately from 2 different maps to avoid resemblance. Further data was collected in dynamic weather conditions.

3.6.5 Data Preprocessing

Both the datasets have been broken down into 3 parts for Training, Validation and Testing in a 18:1:1 ratio. Training had 6750 images, Testing set had 365 images and Validation set had 366 images. The coordinates of the 2048 interesting points randomly selected from the camera's field of view for every image makes one input to the neural network of size (7481,2048,3). The ground truth is given from an array of classes which converts and encodes the three classes "Cars" , "Pedestrians" and "Vans" from categorical variables to binary variables. For instance if the object in

Figure 3.20: Example images from data generated through CARLA 0.8.4

the image is a car the class is stored as [1.0, 0.0, 0.0], the value of 1.0 referring to the car. The shape of the classes input is (7481, 3) The information about the corners of the bounding box are stored in an array of size (7481, 8, 3) meaning 8 corners with 3 values for x,y,z coordinates. The RGB images were first converted and concatenated to make an array of all the images. This was preprocessed by the respective CNNs that were being used to implement Transfer Learning. The intermediate output obtained extracting the output of the model was used as an input to the model being trained. This intermediate output contained the feature maps required for training. These arrays are fed as input to the neural network and the model is compiled to enable 3D Object Detection.

3.6.6 Parameters and Training

All the metrics, optimizers and loss functions were directly used from the available functions in the Keras Library. The initial network architecture was adapted from the PointNet structure. The batch normalization layers and a few max pooling layers were

removed from the PointNet architecture. Upon experimentation it was found that these layers have been hindering the accuracy of the bounding box for our application. Above this architecture, a multi layer perceptron with 4 fully-connected hidden layers of 1024, 512, 256 and 128 units was established, with Rectified linear unit (ReLU) as the activation function, to serve our purpose of classifying the object and regressing a bounding box. The above derived model has 3,988,251 trainable parameters. Initially the model was trained for 200 epochs with the adam optimizer at a decaying learning rate of 0.01 and batch size of 32 with mean squared error loss for bounding boxes and categorical cross entropy for classification. This has produced terrible results in estimation and classification of the object. Parameter tuning was done slowly changing all the parameters one by one. Firstly, the categorical cross entropy was replaced by categorical hinge function to minimize the loss. This showed satisfactory performance but there was more scope for betterment. After this mean absolute error was used to optimize loss for the bounding box prediction. This too did not show better performance. By mixing and matching the loss functions for training the network the optimum settings for loss functions were found when the model performed well on using the huber loss function for box losses and mean squared error function for the classification losses. The number of epochs was gradually increased to test the performance of the model. The number of epochs and batch size which gave best results were 550 epochs and 64 images respectively.

3.6.7 Evaluation Metrics

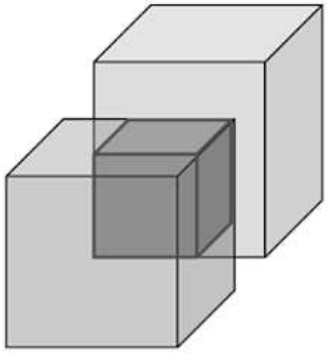
To evaluate the performance of the model, two important metrics were chosen namely, 1. IoU and 2. Class Probability.

IoU: Intersection over Union(IoU) is a metric used to compare predictions with

ground truth. Mathematically it can be represented as,

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \tag{3.2}$$

where A is the box volume of prediction and B is the box volume of ground truth.



$$\mathbf{3D\ IOU} = \frac{\boldsymbol{Intersection(obj_1, obj_2)}}{\boldsymbol{Union(obj_1, obj_2)}}$$

Figure 3.21: IoU (Donghyeop Shin, 2019)

The threshold for IoU is set at 0.5 and progresses to 1 with step size of 0.05. The closer the value to 1, the more accurate is the prediction with respect to ground truth. Generally the IoU of a True Positive is above the threshold value and if it is below the threshold it is a False Positive.

Class prediction probability: The objects in the dataset are distributed over 8 major categories including car, van, pedestrian and miscellaneous. When the model detects an object, there is a possibility that the model may classify the object in more than one categories. Hence, it is important to add a class prediction constraint such that, 1. The class with highest probability is chosen as the output and 2. the chosen probability is higher than 0.51 or 51%.

Precision and Recall: Recall describes the ratio of number of True Positive

predictions to the total number of images or ground truths. Precision is defined as the ratio of True Positive predictions to total number predictions

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{Numberofgroundtruths}$$
$$Precision = \frac{TP}{TP+FP} = \frac{TP}{Numberofpredictions}$$

Here TP - True Positive; FN - False Negative; FP - False Positive

Chapter 4

RESULTS AND DISCUSSIONS

A comprehensive framework for implementing controllers on Autonomous Vehicles in both the physical and virtual domains has been proposed. This framework enables us to collect driving data from multiple sensors like cameras, LiDARs, IMU, GNSS, obstacle detection sensor, lane invasion sensor, collision sensor. In future, this data will be used to characterize various driving behaviors from participants with different driving styles and motivations. The moral values and motivations of these participants will be determined by the Schwartz (Schwartz *et al.*, 2012) questionnaire and Moral Foundation Theory (Graham *et al.*, 2013) surveys and crash responses. Figure 4.1 summarizes the framework.

A sample visualization of the data logged through sensors from the simulated vehicle in CARLA, extracted during a test run can be seen from Figure 4.2.

In Figure 4.2, X and Y axes correspond to the position of the ego vehicle and hence the plot show the trajectory of the vehicle across the map. Each blue triangle refers to a collision and the image and lidar data corresponding to the frame of collision have been shown.

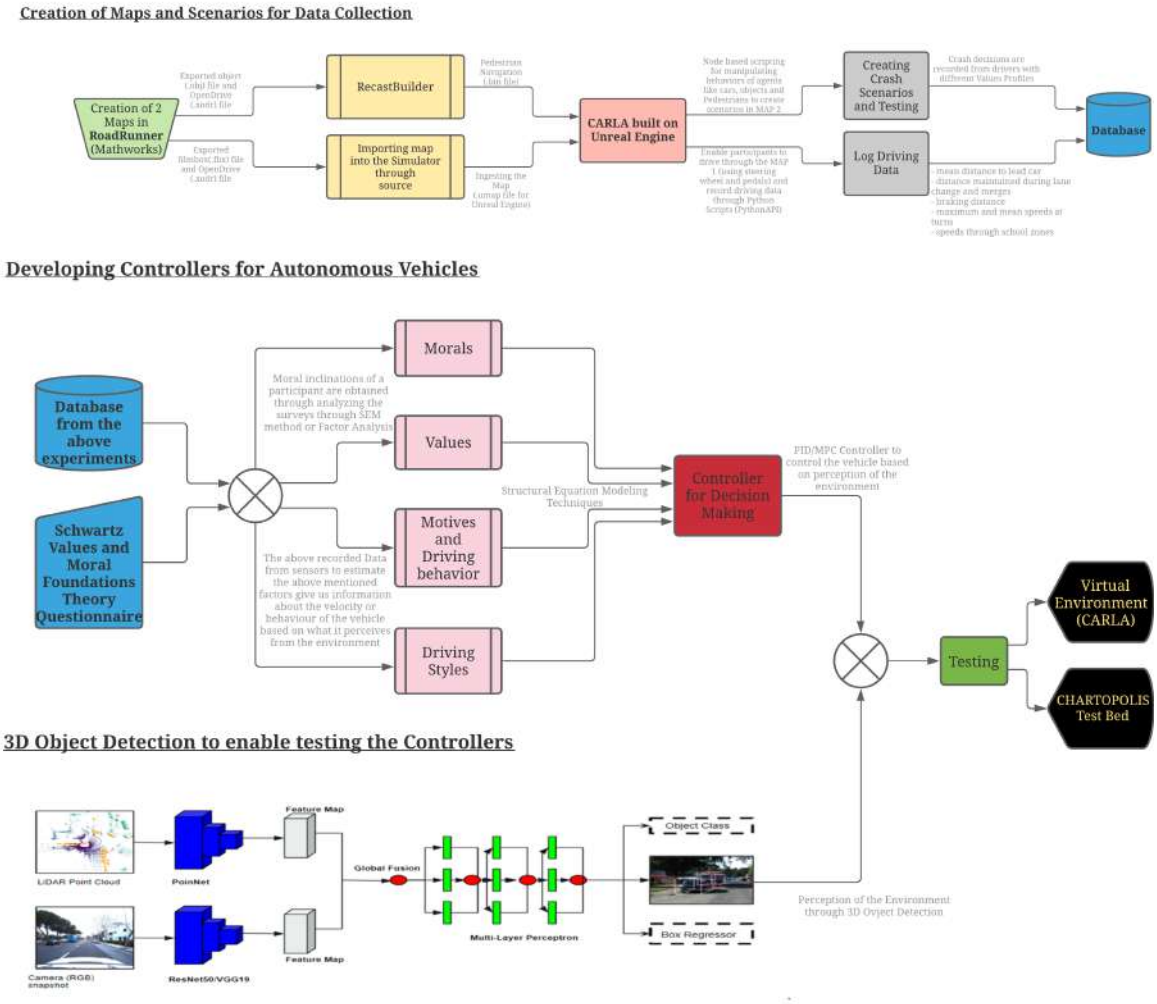


Figure 4.1: The Framework

As can be referred from the graphs in Figure 4.3 Batch normalization layers in the neural network cause manipulations of the point clouds and thus decreases the accuracy of the model and fails to optimize or minimize the loss. Therefore all the Batch normalization layers have been voided. The loss function for bounding box was huber loss and for classification it was mean squared error. The batch size was 32 and the it was trained for 500 epochs.

Both ResNet (Refer figure 4.4) and Xception (Refer figure 4.5) have been trained on same parameters as above except they were tested on both the loss function of

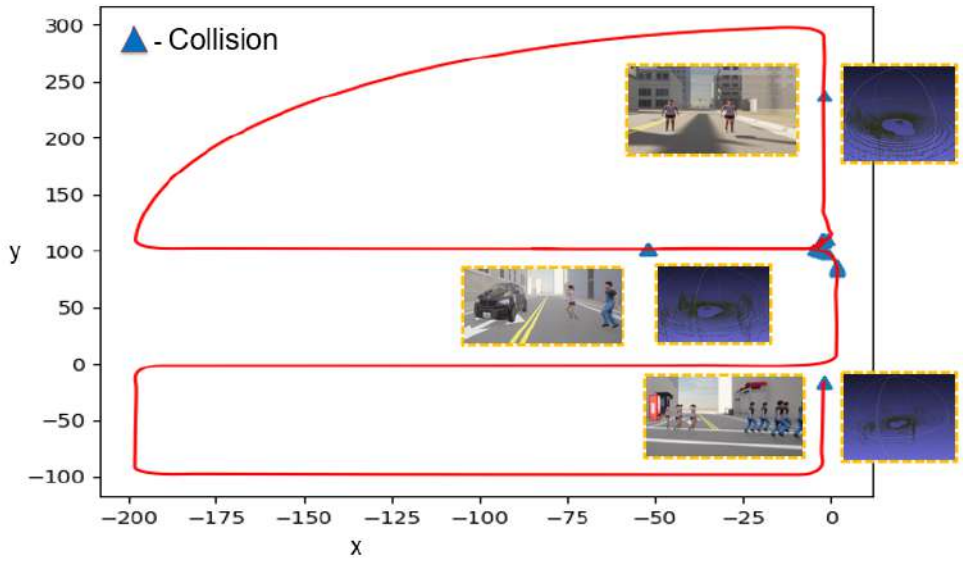


Figure 4.2: Sample Data Extracted from a Test Run

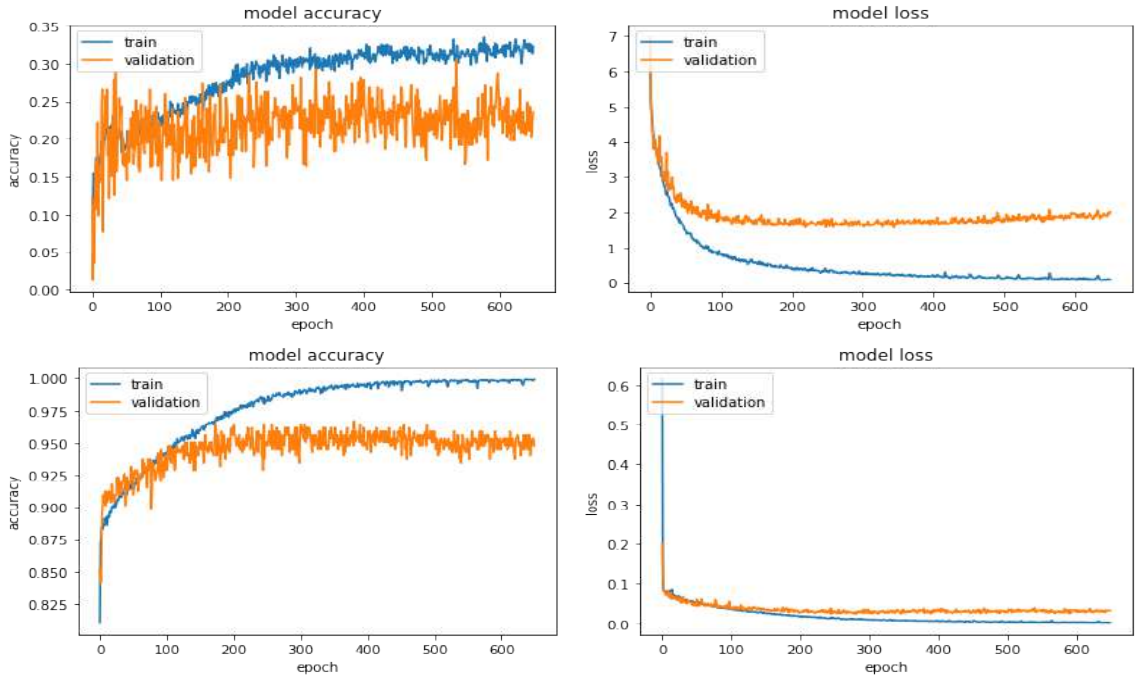


Figure 4.3: Results with Batch Normalization: Bounding Box Accuracy (top left); Bounding Box Loss (top right); Classification Accuracy (bottom left); Classification Loss (bottom right)

categorical hinge and mean squared error for the classification loss. Mean squared error seemed to yield better results.

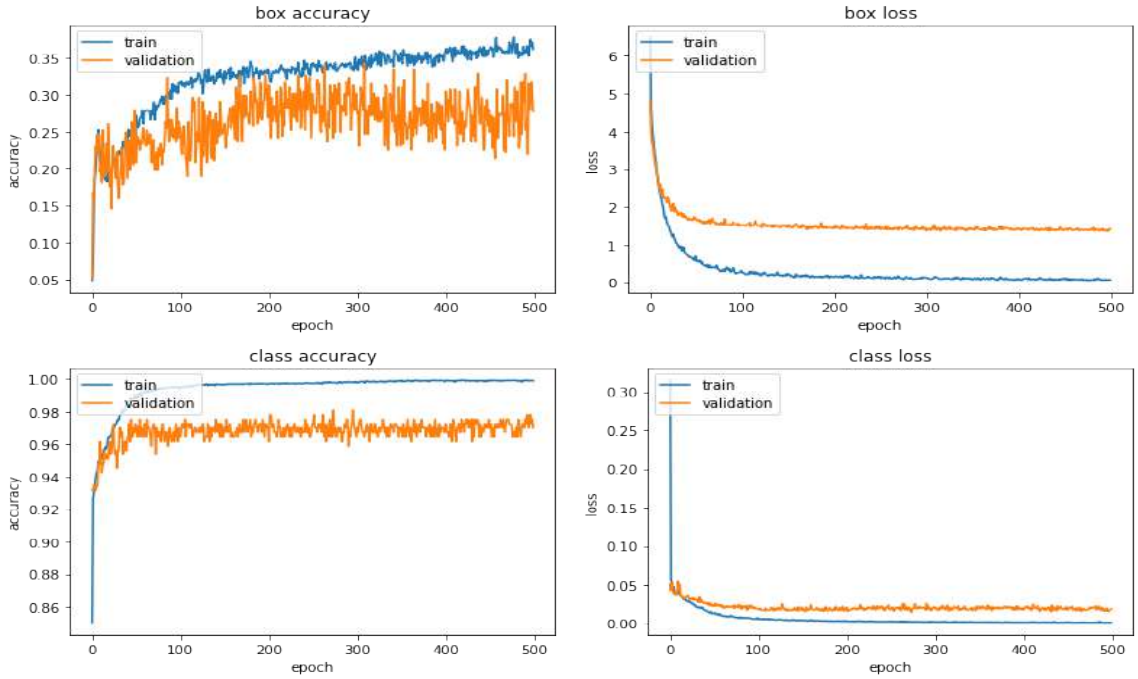


Figure 4.4: Results for ResNet Network: Bounding Box Accuracy (top left); Bounding Box Loss (top right); Classification Accuracy (bottom left); Classification Loss (bottom right)

From the plots and Table 4.1, **Xception** is the best choice for training our Object Detection model.

The metrics for the models trained using different pre-trained CNNs is shown in Table 4.1. With the Average IoU and classification accuracy as the criteria both ResNet152 and Xception have obtained decent results. Xception outperformed ResNet152 in classification accuracy.

	VGG19	ResNet152	InceptionV3	Xception
IoU	0.61	0.70	0.69	**0.72**
Class Accuracy	94.08%	95.62%	95.38%	**96.72**%
Precision	0.9459	0.9527	0.9402	0.9671
Recall	0.9480	0.9453	0.9445	0.9644

Table 4.1: Evaluation Metrics Tested on the KITTI dataset

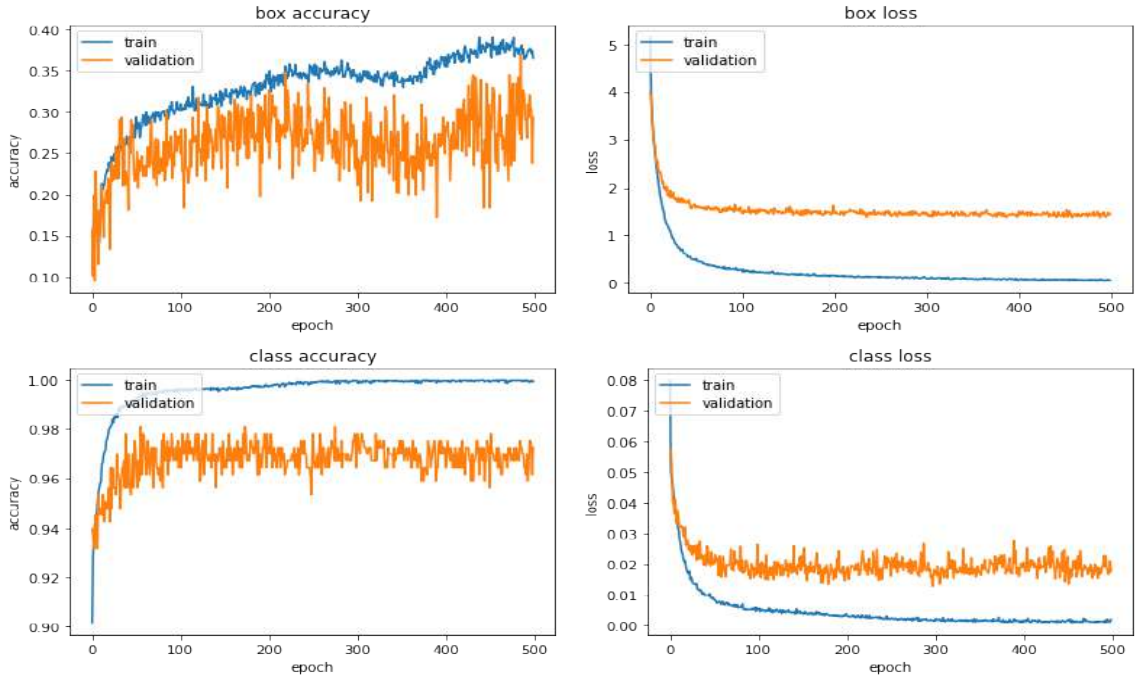


Figure 4.5: Results for Xception Network: Bounding Box Accuracy (top left); Bounding Box Loss (top right); Classification Accuracy (bottom left); Classification Loss (bottom right)

The class will be determined by the color of the bounding box

Car - Red ; **Pedestrian - Yellow** ; **Van - Green**

Figure 4.6: Test Results of the KITTI Vision Benchmark Suite Dataset

Figure 4.7: Test Results of the Virtual Dataset

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

A comprehensive framework for collecting data on human driving styles and automatically classifying objects in order to implement human-like AV driving behaviors on a physical tesbed has been developed in the CARLA driving simulator.

Modeled a city in the simulator that is similar to the test bed in context.

Three crash scenarios and a school zone scenario have been created and animated to give the participants a more realistic experience.

All possible data from sensors like cameras, lidars, GNSS, IMU, collision detection, obstacle detection and lane invasion sensors is logged during simulator driving trials for post analysis of the crash decisions made by the participants.

To facilitate these tests Object Detection algorithms and models have been tested and Xception network pre-trained on ImageNet has proven to be very effective in finding the bounding boxes of objects in an image using a Sensor Fusion architecture trained through transfer learning

5.2 Future Work

Create more scenarios in CARLA and run driving simulation trials with a large set of people with different Values Profiles.

Designing and Implementing human-like driving styles, extracted from data logged from CARLA, on Go-CHART robotic cars once the CHARTOPOLIS test bed is ready.

The current dataset is saturated and does not produce great results overall because

the CARLA 0.8.4 version has very few vehicle models and this caused repetition in the features and the model was over-fitted with the same features. A larger virtual dataset with many instances per class (a wide range of vehicles) and one which is larger in size is needed so as to achieve a decent model trained on meaningful features.

Implement Real-Time Object detection on both the physical test bed and CARLA

REFERENCES

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning", in "12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)", pp. 265–283 (2016).

Alyssa, "The trolley problem", Retrieved from `https://sites.google.com/site/has233aw/the-trolley-problem` (2016).

American Automobile Association, *Three in Four Americans Remain Afraid of Fully Self-Driving Vehicles*, Retrieved from `https://newsroom.aaa.com/2019/03/americans-fear-self-driving-cars-survey/` (2019).

Andrew Hawkins, *Tesla's Full Self-Driving software is starting to roll out to select customers*, Retrieved from `https://www.theverge.com/2020/10/21/21527577/tesla-full-self-driving-autopilot-beta-software-update` (2020).

Awad, E., S. Dsouza, R. Kim, J. Schulz, J. Henrich, A. Shariff, J.-F. Bonnefon and I. Rahwan, "The moral machine experiment", Nature **563**, 7729, 59–64 (2018).

Basu, C., Q. Yang, D. Hungerman, M. Sinahal and A. D. Draqan, "Do you want your autonomous car to drive like you?", in "2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI", pp. 417–425 (2017).

Beltrán, J., C. Guindel, F. M. Moreno, D. Cruzado, F. García and A. De La Escalera, "Birdnet: A 3d object detection framework from lidar information", in "2018 21st International Conference on Intelligent Transportation Systems (ITSC)", pp. 3517–3523 (2018).

Bindhi, M. and A. Gupta, *Deep Sensor Fusion for 3D Bounding Box Estimation and Recognition of Objects*, Retrieved from `http://cs230.stanford.edu/files`$_w$*inter*$_2$018/*projects*/6939556.*pdf*(2018).

Brekke, Å., F. Vatsendvik and F. Lindseth, "Multimodal 3d object detection from simulated pretraining", arXiv preprint arXiv:1905.07754 (2019).

Business Insider, "Many self-driving car accidents have been caused by humans", Retrieved from `https://www.businessinsider.com/self-driving-car-accidents-caused-by-humans-2018-9` (2018).

CARLA- Car Learning to Act, *CARLA Documentation*, Retrieved from `https://carla.readthedocs.io/en/latest/` (2017).

CDC, *Centers for Disease Control and Prevention: Road Traffic Injuries and Deaths*, Retrieved from `https://www.cdc.gov/injury/features/global-road-safety/index.html` (2019).

Chen, X., K. Kundu, Z. Zhang, H. Ma, S. Fidler and R. Urtasun, "Monocular 3d object detection for autonomous driving", in "The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2016a).

Chen, X., K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler and R. Urtasun, "3d object proposals for accurate object class detection", in "Advances in Neural Information Processing Systems 28", edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama and R. Garnett, pp. 424–432 (Curran Associates, Inc., 2015).

Chen, X., H. Ma, J. Wan, B. Li and T. Xia, "Multi-view 3d object detection network for autonomous driving", (2016b).

Chollet, F., "Xception: Deep learning with depthwise separable convolutions", (2017).

Dai, W., Q. Yang, G.-R. Xue and Y. Yu, "Boosting for transfer learning", in "Proceedings of the 24th international conference on Machine learning", pp. 193–200 (2007).

Danny Yadron, Dan Tynan, *Tesla Driver Dies in first Fatal crash while using Autopilot mode*, Retrieved from `https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk` (2016).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in "2009 IEEE conference on computer vision and pattern recognition", pp. 248–255 (Ieee, 2009).

Dixit, V. V., S. Chand and D. J. Nair, "Autonomous vehicles: disengagements, accidents and reaction times", PLoS one **11**, 12, e0168054 (2016).

Donghyeop Shin, I. K., "Deep neural network-based scene graph generation for 3d simulated indoor environments", KIPS Transactions on Software and Data Engineering (2019).

Dosovitskiy, A., G. Ros, F. Codevilla, A. Lopez and V. Koltun, "CARLA: An open urban driving simulator", in "Proceedings of the 1st Annual Conference on Robot Learning", pp. 1–16 (2017).

Fagnant, D. J. and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations", Transportation Research Part A: Policy and Practice **77**, 167–181 (2015).

Foot, P., "The problem of abortion and the doctrine of the double effect", Oxford Review **5**, 5–15 (1967).

Geiger, A., P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The kitti dataset", International Journal of Robotics Research (IJRR) (2013a).

Geiger, A., P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The kitti dataset", International Journal of Robotics Research (IJRR) (2013b).

Gerdes, J. C. and S. M. Thornton, *Implementable Ethics for Autonomous Vehicles*, pp. 87–102 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2015), Retrieved from `https://doi.org/10.1007/978-3-662-45854-9`$_5$.

Girshick, R., "Fast r-cnn", 2015 IEEE International Conference on Computer Vision (ICCV), December 2015, pp.1440-1448 (2015).

Goodall, N. J., "Ethical decision making during automated vehicle crashes", Transportation Research Record **2424**, 1, 58–65 (2014a).

Goodall, N. J., "Machine ethics and automated vehicles", in "Road vehicle automation", pp. 93–102 (Springer, 2014b).

Google Waymo, *Waymo is opening its fully driverless service to the general public in Phoenix*, Retrieved from `https://blog.waymo.com/2020/10/waymo-is-opening-its-fully-driverless.html` (2020).

Graham, J., J. Haidt, S. Koleva, M. Motyl, R. Iyer, S. P. Wojcik and P. H. Ditto, "Moral foundations theory: The pragmatic validity of moral pluralism", in "Advances in experimental social psychology", vol. 47, pp. 55–130 (Elsevier, 2013).

Greg Bensinger, Tim Higgins, *Video Shows Moments Before Uber Robot Car Rammed Into Pedestrian*, Retrieved from `https://www.wsj.com/articles/video-shows-final-seconds-before-fatal-uber-self-driving-car-crash-1521673182` (2018).

HANA, S. A., "Intention to comply with the school zone speed limit: Scenario-based study", Journal of the Eastern Asia Society for Transportation Studies **12**, 1965–1973 (2017).

He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", CoRR **abs/1512.03385**, Retrieved from `http://arxiv.org/abs/1512.03385` (2015a).

He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", (2015b).

Hegde, V. and R. Zadeh, "Fusionnet: 3d object classification using multiple data representations", (2016).

Hengstler, M., E. Enkel and S. Duelli, "Applied artificial intelligence and trust—the case of autonomous vehicles and medical assistance devices", Technological Forecasting and Social Change **105**, 105–120 (2016).

Himmelsbach, M., A. Mueller, T. Lüttel and H.-J. Wünsche, "Lidar-based 3d object perception", in "Proceedings of 1st international workshop on cognition for technical systems", vol. 1 (2008).

James Minton, Vahid Behzadan, *TrolleyMod v1. 0: An Open-Source Simulation and Data-Collection Platform for Ethical Decision Making in Autonomous Vehicles*, Retrieved from `https://github.com/zminton/TrolleyMod` (2018).

Johnson, K.A., Berman, S., Chiou, E., Pavlic, T.P., Cohen, A.B., *Toward virtuous vehicles: Identifying the moral profile of good drivers as a basis for ethical decision-making in self-driving cars*, in preperation, 2020 (2020).

Kanade, T., C. Thorpe and W. Whittaker, "Autonomous land vehicle project at cmu", in "Proceedings of the 1986 ACM Fourteenth Annual Conference on Computer Science", CSC '86, p. 71–80 (Association for Computing Machinery, New York, NY, USA, 1986), Retrieved from `https://doi.org/10.1145/324634.325197`.

Kannapiran, S. and S. Berman, *Go-CHART: A miniature remotely accessible self-driving car robot* (2018).

Kesten, R., M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang and V. Shet, "Lyft level 5 av dataset 2019", `https://level5.lyft.com/dataset/` (2019).

Ketkar, N., "Introduction to keras", in "Deep learning with Python", pp. 97–111 (Springer, 2017).

Kuderer, M., S. Gulati and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration", in "2015 IEEE International Conference on Robotics and Automation (ICRA)", pp. 2641–2646 (IEEE, 2015).

Kuipers, B., "Human-like morality and ethics for robots", in "AAAI Workshop: AI, Ethics, and Society", (2016).

Li, B., "3d fully convolutional network for vehicle detection in point cloud", in "2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", (2017).

Li, B., T. Zhang and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network", (2016).

Li, P., X. Chen and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)", (2019).

Ma, X., Z. Wang, H. Li, P. Zhang, X. Fan and W. Ouyang, "Accurate monocular object detection via color-embedded 3d reconstruction for autonomous driving", (2019).

Mathworks, *RoadRunner*, Retrieved from https://www.mathworks.com/help/roadrunner/index.html (2019).

Michael Clamann, Nancy Pullen-Seufert, *Considerations for Deploying Automated Driving Systems Around Schools*, Retrieved from http://pedbikeinfo.org/cms/downloads/PBIC$_W$*hitePaper*$_A$*DS*%20*Near*%20*Schools.pdf*(2020).

Mihaly Heder, *The epistemic opacity of autonomous systems and the ethical consequences. AI Soc*, Retrieved from https://doi.org/10.1007/s00146-020-01024-9 (2020).

Mike Brown, *Waymo vs Tesla*, Retrieved from https://www.inverse.com/article/50456- waymo-vs-tesla-who-will-win-the-self-driving-car-race (2018).

Moral Machine, *What should the self-driving car do?*, Retrieved from https://www.moralmachine.net (2016).

Moral Machine Results, *MIT Moral Machine Statistics based on Countries*, Retrieved from http://moralmachineresults.scalablecoop.org/ (2018).

NavLab 5, *PANS: A Portable Navigation Platform*, Retrieved from https://www.cs.cmu.edu/ tjochem/nhaa/navlab5$_d$*etails.html*(1995).

Neal Boudette, *Autopilot cited in Death of Chinese Driver*, Retrieved from https://www.nytimes.com/2016/09/15/business/fatal-tesla-crash-in-china-involved-autopilot-government-tv-says.html (2016).

NHTSA, *Automated Vehicles for Safety*, Retrieved from https://www.nhtsa.gov/technology -innovation/automated-vehicles-safetytopic-road-self-driving (2019).

Prokhorov, D., "A convolutional learning system for object classification in 3-d lidar data", IEEE Transactions on Neural Networks (Volume: 21 , Issue: 5) (2010).

Qi, C. R., H. Su, K. Mo and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation", (2016).

Qian, R., D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger and W.-L. Chao, "End-to-end pseudo-lidar for image-based 3d object detection", (2020).

SAE, *Taxonomy and Definitions for Terms Related to Shared Mobility and Enabling Technologies*, Retrieved from https://doi.org/10.4271/J3163$_2$01809(2018).

Schwartz, S. H., "Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries", Advances in experimental social psychology **25**, 1, 1–65 (1992).

Schwartz, S. H., J. Cieciuch, M. Vecchione, E. Davidov, R. Fischer, C. Beierlein, A. Ramos, M. Verkasalo, J.-E. Lönnqvist, K. Demirutku *et al.*, "Refining the theory of basic individual values.", Journal of personality and social psychology **103**, 4, 663 (2012).

Self Driving Times, *The Interesting Estimation of 33 Corporations Working on Self Driving Cars*, Retrieved from `https://www.selfdrivingtimes.com/posts/the-interesting-estimation-of-33-corporations-work` (2016).

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", (2014).

Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in "International Conference on Learning Representations", (2015).

Song, S. and M. Chandraker, "Robust scale estimation in real-time monocular sfm for autonomous driving", in "The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2014).

Song, S. and J. Xiao, "Sliding shapes for 3d object detection in depth images", Proceedings of the 13th European Conference on Computer Vision (ECCV2014) (2014).

Su, H., S. Maji, E. Kalogerakis and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition", (2015).

Subramanyam, R., *CHARTOPOLIS: A Self Driving Car Test Bed* (2018).

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions", (2014).

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision", (2015).

T Pietrasik, *World Health Organization: Road Traffic Injuries*, Retrieved from `https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries` (2020).

Thomson, J. J., "The trolley problem", The Yale Law Journal **94**, 6, 1395–1415 (1985).

Thornton, S. M., S. Pan, S. M. Erlien and J. C. Gerdes, "Incorporating ethical considerations into automated vehicle control", IEEE Transactions on Intelligent Transportation Systems **18**, 6, 1429–1439 (2016).

Unreal Engine, *Epic Games: Unreal Engine*, Retrieved from `https://www.unrealengine.com` (2004).

Unreal Engine Documentation, *Unreal Engine Documentation: Blueprints*, Retrieved from `https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html` (2014).

Wallach, W., S. Franklin and C. Allen, "A conceptual and computational model of moral decision making in human and artificial agents", Topics in cognitive science **2**, 3, 454–485 (2010).

Wilson, S., R. Gameros, M. Sheely, M. Lin, K. Dover, R. Gevorkyan, M. Haberland, A. Bertozzi and S. Berman, "Pheeno, a versatile swarm robotic research and education platform", IEEE Robotics and Automation Letters **1**, 2, 884–891, Retrieved from `10.1109/LRA.2016.2524987` (2016).

Zhou, Y. and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2018).