Car Driving Without Cameras

By

JONES MABEA AGWATA

Department of Computer Science
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of MASTER OF SCIENCE in the Faculty of Engineering.

SEPTEMBER 2018

Word count: ten thousand and four

Executive Summary

The need for robust object detection in 3D point clouds has greatly increased with the ongoing push for autonomous vehicles (AVs). Most of these systems use Light Detection and Ranging (LiDAR), cameras or a combination of both in order to perform object detection. LiDAR presents objects as point clouds in a 3D space thus offering critical shape information of objects in view. However, this representation is sparse. As a result, LiDAR-based detection performs poorly as compared to multimodal methods that have helped overcome this. Nonetheless, multimodal methods are often complex to set up and synchronise with the cost of components running into thousands of pounds. In light of this, reducing the number of sensors while still maintaining or even improving the accuracy has become a topic of interest by industry players who are developing AVs. With the cost of LiDAR declining following recent improvements in solid-state LiDAR technology, dropping cameras in favour of LiDAR has become more plausible. As compared to cameras that suffer drawbacks such as visibility issues in extreme weather, LiDAR is extremely robust and accurate in various conditions.

This project will be 65% type I (Software Development) and 35% type II (Theoretical).

Dedication and acknowledgements

Here goes the dedication.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

Accelerated by recent advancements in technology, the prospect of Autonomous Vehicles (AVs) driving in public roads is becoming more and more a reality. As this is an emerging field, there are numerous variations of implementations by different companies. Arguably, a key characteristic of these implementations is a large number of perception sensors including cameras, radars and Light detection ranging sensors(LiDAR). This is necessary for mapping the environment around the vehicle in order to safely navigate. In addition to the high cost of these sensors, fusing their input and running object detection models requires powerful processing units such as GPUs. This process consumes a lot of power and generates a lot of heat.

In an effort to reduce the cost, companies are exploring different ways to reduce the number of sensors while still achieving a high level of navigational accuracy and safety. Extensive research has been undertaken to assess the performance of these sensors in different weather contexts such as in rain, fog and snow. However, the performance of these sensors in different area contexts, that is urban or non-urban, has not been exhaustively explored. Urban environments tend to have more dynamic objects and scenarios as compared to the static ones in most non-urban areas. As such, it can be argued that depending on the area context, the sensor configuration could be modified.

You probably need to present a succinct critique comment for each of the top 4 pieces of related work (which 'coincidentally' are limitations you try to address in your aims). Why do we need to test with different datasets? different models? Why is this important scientifically and also for companies developing lidar-onlu systems?

In order to investigate this, state of the art object detection models will be considered. Firstly, VoxelNet [42]], a LiDAR only model that uses point clouds as input. Secondly, Aggregated View Object Detection(AVOD) [20], a multimodal model that fuses image and point cloud data. Both model implementations were available on GitHub and were modified in order to align with the

aims of this project.

1.1 Aims and Objectives

Following the motivations in the presented discussion , the performance of LiDAR only and multimodal(LiDAR and Camera) models in different contexts will be investigated with the aim of reducing the number of sensors in AVs. To achieve this aim, the following objectives will need to be fulfilled:

1. Detect and characterise the context of images and point clouds.

2. Evaluate the performance of both models in different contexts.

3. Validate performance of VoxelNet(single sensor) on a custom dataset.

1.2 Deliverables

The deliverables are:

- **Image and LiDAR Context Classifier**. Available as Jupyter interactive notebooks including pre-trained models.

- **Custom VoxelNet Model** Modified VoxelNet model including interactive notebooks for training, testing and validating the model.

- **Custom AVOD Model** Modified AVOD model including interactive notebooks for training, testing and validating the model.

- **Validation Dataset** Point Cloud dataset obtained from the University of Bristol Smart Internet Lab working on connected and AVs. Tools to convert the dataset into a trainable input for VoxelNet will be provided as well as some annotated sample frames.

- **Evaluation report.** The following topics will be discussed.

    1. A review of related research and implementations tackling object detection in AVs.
    2. Economic, ethical and legal analysis of the implementation and its potential impact on the development of AVs.
    3. Evaluation of context classifiers and object detection models in different contexts.
    4. Validation of VoxelNet using the Smart Internet Lab dataset.

1.3 Report structure

This report will consist of five main chapters.

- Chapter 2 discusses the different components of AVs, current implementations in the industry, a background on the research that has been undertaken in the field of object detection.
- Chapter 3 details the project execution and the methods undertaken to achieve the objectives.
- In Chapter 4, the results following evaluation of the methods will be discussed and analysed.
- Finally, a concluding chapter discusses the major findings, whether the objectives were achieved and a justification of their implications.

CHAPTER 2

BACKGROUND

The idea of autonomous vehicles(AVs) is not new. As early as 2005, DARPA had invested heavily in the creation of unmanned trucks and organised for the Urban Challenge [6] to allow for different teams to showcase their unmanned vehicles. However, due to the challenges such as low computational power and underdeveloped AI and ML systems, the resulting implementations were not practical and had a high fault rate of 1 fault in 100 miles compared to the human fault rate of around 1 in 100 million miles. Nonetheless, from this challenge, it was clear that the prospect of AVs was plausible and indeed possible.

Currently, AVs are divided into five levels of autonomy as defined be the Society of Automotive Engineers(SAE):

SAE level	Description
0	No AV control systems. An example is blind spot indicators.
1	Basic driver assistance built into vehicle design. An example is the cruise control function.
2	Basic AV control systems. Driver required to monitor the environment and take back control if need be. A good example is the Tesla Autpilot.
3	AVs can safely navigate and drive within mapped environment. Driver required to monitor the environment and take back control if need be.
4	Highly autonomous control capable of handling most conditions but the driver has the option to take control. Renault ISymbioz is an example of a level 4 AV.
5	Completely autonomous with zero human involvment.

2.1 Components of an AV

- **LiDAR** - LiDAR provides highly detailed 3D information about the environment around the vehicle and objects in it. LiDAR operates by sending out pulses of lasers and recording

the reflections of the pulses from objects. By comparing this with the time taken for the lasers to be reflected(time of flight) and their direction, the distance of these objects can be calculated and mapped in a point cloud.

LiDAR units require complex optical systems that are expensive to build. As a result, they are the most expensive sensors in AVs with the top end such as Velodyne HDL-64E shown in figure 21 costing more than 50,000$. In a bid to reduce the cost of LiDAR units, different companies are exploring different design methods that are cheaper but still able to offer the same performance as the top end LiDAR units such as solid state LiDAR[1].

Figure 21: Velodyne LiDAR family. From left: Velodyne HDL-64E , HDL-32E , VLP-16

- **Cameras** - Cameras mounted on the vehicle are used for classification and identification of various objects on the road. Cameras can also be used to create 3D maps of the surrounding environment.By combining two cameras, a stereo image can be captured that provides depth information. Alternatively, by combining a camera and IR Laser sensor for depth estimation, RGB-D [17] images are obtained and mapped in a point cloud.

- **Position Estimators** - Position estimators are a group of sensors used for navigation of the vehicle. These include GPS systems, odometers and gryometers.

[1]See appendix **??**

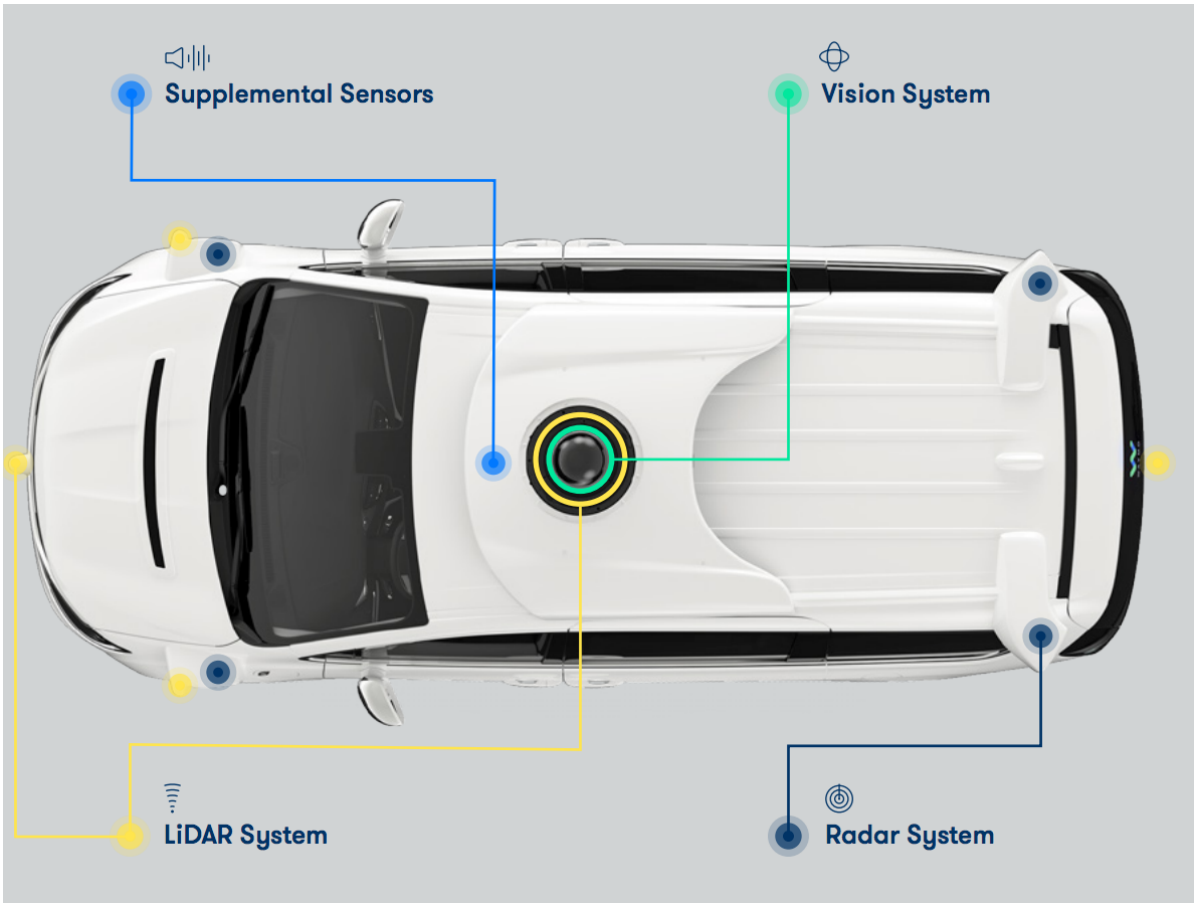

Figure 22: Components of Waymo's Self Driving Car [39]

- **Distance Sensors** - Distance sensors such as radars and sonars are important for gauging the distance of objects on the road. Radars are the most commonly used distance sensors and they work by transmitting radio waves and recording the reflected radio waves from objects. As compared to cameras and LiDARs, radars work well in a variety of low visibility scenarios such as poor weather. However, the reflectivity of these radio waves depends on the nature of objects, their size, absorbtion characteristics and the transmitting power. As such, it is may not be effective for detecting objects with low absorbtion characteristics such as pedestrians and animals.

- **Processing Unit** - In order to process all the data from sensors on the vehicle, AVs require powerful processing units to to do so in real time. Most ML/AI algorithms used for detecting and identifying objects from LiDAR and camera data demand large amounts of processing power. This is achieved through the use of CPUs, GPUs, Field Programmable Gate Arrays(FPGA)[5], Application Specific Integrated Circuits(ASICs)[37] or combinations with each other.

AVs have three main modes of operation namely,

- **Perception** - This is the first step which involves processing the input from the sensors. In this mode tasks such as object detection and tracking, lane detection, traffic sign detection and recognition are performed.

- **Planning** - This is the next step after detection and recognition tasks are performed . In this stage route and trajectory planning algorithms are run to plan how the vehicle should navigate in the immediate environment as well as a route to a target location. These algorithms are required to handle complex situations to ensure safety of the passengers and other road users.

- **Control** - This stage involves the execution of plans created in the planning stage. This stage is crucial as the actuators involved in steering and movement have to be able to be able to accurately follow the plans. This involves calculation of energy and forces. At this stage the trajectories and movement of other road users and objects have to be calculated in order to anticipate and avoid any accidents.

2.2 Industrial Approaches

In order to be viable, AVs need to meet a few constraints as discussed in [24].

- **Performance**
  At the moment there are no clear regulations as to how fast the perception, planning and control pipeline should be. However according to research by Brown et al. [4], humans take around 600 ms to respond and brake when expecting an interruption, however this figure shoots to 850 ms when an unexpected situation arises. In addition, Newell et al [28], established that the fastest human response time is between 100-150ms. These figures can be used as a baseline while developing a pipeline. In this pipeline, there are two factors to consider, namely the frame rate (frequency of the data from sensors) and the processing latency (time taken to process data). As such, an AV system should be able to react within 100ms, that is, faster than the human response time.

- **Storage**
  AVs should be able to store maps of different areas with fine granularity for accuracy during localisation. As a result, the storage of these maps can run into tens of Terabytes. Despite recent advances in cloud technology and the emergence of 5G connectivity that is significantly fast. Downloading these maps would take significant time and would also render the car unusable in case of no internet connectivity. As such, the vehicles should have enough storage to store these maps locally.

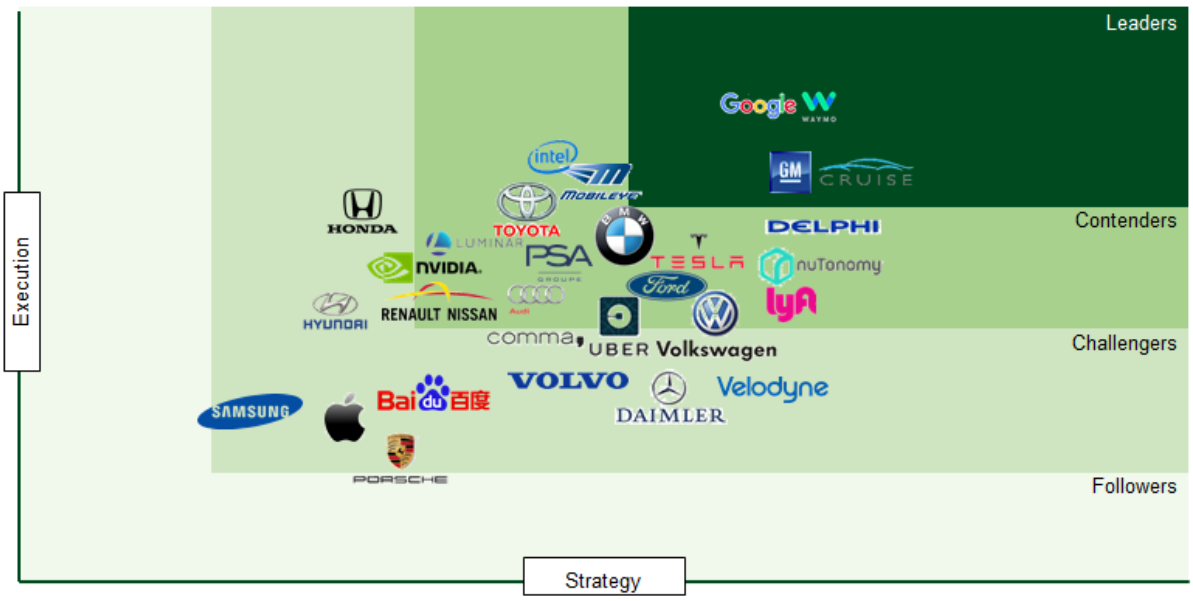

Figure 23: AV Leaderboard. Navigant Research[33]

- **Power**
  Most of the major industry players have moved to electric-vehicles for their AV systems. Depending on the equipment and sensor configurations used in these systems, the power usage can range from 500 watts to 1.5 kilowatts. Given that the AVs have limited battery capacity, heavy power consumption by these systems can lead to poor driving ranges hence making the cars less viable. As such the configuration of these systems has to be carefully considered to ensure a reasonable driving range.

- **Thermal**
  Processing components in the AV systems such as CPUs and GPUs require a significant amount of energy to cool. This is necessary to ensure that they operate within their recommended thermal operating range. Failure to do so could result in the failure of these systems. As such, additional cooling systems have to be installed in the vehicle in order to ensure this.

Following the discussion above, the next subsections will review how different industry players are implementing their AV systems with special focus to the issue of perception

2.2.1 Camera

Mobileye[27], is one of the top contenders in the development of camera only AVs. It was recently acquired by Intel and believes that AVs should be able to accurately and safely navigate using cameras only given the fact that humans are able to do so with vision only. Previously, MobilEye was a supplier of vision systems for Advanced Driver Assistance Systems and had a partnership

with Tesla to supply their vision systems prior being acquired by Intel. Given their extensive background in developing these vision systems, the partnership with Intel aims to develop a complete autonomous driving package.[18]

2.2.2 Camera and Radar

Another major contender that is using the camera and radar configuration is Tesla. Elon Musk, the founder, believes that LiDAR is not necessary for AV perception. However, just recently, a Tesla vehicle was involved in a crash while in 'autopilot' mode that resulted in the loss of life. A major attributing factor to this accident was caused by the fact that bright sunlight blinded the camera thus causing the vehicle not to detect a white truck that was ahead of it. The radar was also unable to detect it thus leading to a collision. LiDAR on the other hand, would have been able to detect the truck despite the poor visibility. This illustrates how unreliable this configuration can be.

2.2.3 LiDAR, Camera and Radar

This configuration is used by most of the leading industry players such as Waymo and GM Cruise. This configuration has proven to be robust and accurate. However, in order to process the amount of fused data, they require large amounts of processing power which in turn may lead to increased power usage.

2.3 Object Detection

2.3.1 Object Detection using Classic Computer Vision Methods

Classic computer vision methods have advanced over the years from simple algorithms such as SIFT[25] and SURF[2] which use descriptors to detect interesting points in images for object detection to much higher level representations such as the Histogram of Gradient(HOG)[11] or Haar Features[38] that were used in classifiers. However, these methods have been unreliably slow and not as accurate. In addition these features are only able to model 2D data. 3D methods such as Intrinsic Shape Signatures(ISS)[], NARF[] and Uniform Sampling have since been developed but are lacking in terms speed. Furthermore, these feature detectors have to be hand-crafted manually which is a tedious process. As a result, they are rarely used in real-time AV applications due to their speed and accuracy.

2.3.2 Object Detection using Deep Learning

The emergence and development of Deep Neural Networks such as Convolutional Neural Networks (CNNs)[21] has offset the reliance of classical methods of object detection in images by allowing for features to be automatically learnt by the network without the need for manual

feature extraction. This is due to the fact that these networks have numerous number of deep layers that are able to capture these features.

Monocular and Stereo Vision

Monocular vision is derived from a single camera. This representation is 2D and lacks depth information. Mono3D by Chen et al [8] is a state of the art object detection model that uses monocular images where they were fed through a CNN and the resulting 2D object proposals were extruded into 3D bounding boxes by performing segmentation.

Stereo vision involves calibrating two cameras intrinsically and extrinsically by matching similar points in images from both cameras. Using the resulting stereo image, it is possible to estimate the depth of objects in the image. However the accuracy of this depends on a few factors such as the resolution, camera quality, lens distortion and the calibration algorithm. 3DOP [9]

LiDAR+Camera

By fusing input from a monocular image and point clouds, it is possible to extract depth information from the images. Chen et al further reiterated their monocular approach into MV3D[10] a multiview 3D object detection network. In their work, they were able to fuse LiDAR and camera input to create a bird's eye view of the surrounding environment and further perform object detection of features.

LiDAR

Point cloud object detection is particularly challenging as established 3D object detection methods for images cannot be applied to point clouds. Point clouds are a set of geometric points in a euclidean space. They are unordered in nature and this presents a particular problem to deep learning methods as they need to be invariant to permutations of the input set.

Depending on how they manipulate the point clouds, current point cloud based architectures can be split into three groups.

1. **Direct Manipulation**

   .

   Pointnet[30] developed by Qi et al was able to overcome this challenge on small sets of point clouds. By using point clouds as direct input into Recurrent Neural Networks[26] They were able to create a global point cloud signature that could classify and segment 3D objects in the point cloud. They further improved this model into Pointnet++[31] which recursively applied PointNet on nested partitions of the point clouds using a hierarchical neural network. In doing so they were able to capture the local structures of the point clouds as a result of their metric nature and thus achieve better performance than PointNet. For both of these publications, the source code was released for public use. Consequently, they have formed a fundamental foundation for further development of point cloud object detection methods.

2. **Voxel Based**
   By dividing the pointclouds into 3D voxel grids, points within these voxels can be sampled to perform feature extraction. VoxelNet [42] is a state of the art architecture that uses this method to extract these features through Voxel Feature Extraction layer. The extracted features are then passed through a RPN to generate detextions.

3. **2D-Based**
   This involves projecting the pointcloud into a 2D image plane coordinate system which can then be trained on existing image CNN models such as VGG-16. VeloFCN [] was one of the first methods to implement this approach using a fully convolutional network. However, VeloFCN did not include bird eye and front view representations.

Network Model	**Mode**	**Code**	**Inference Time**
VoxelNet	*LiDAR*	Unnoficial	***100ms***
AVOD	*LiDAR+Image*	Yes	***100ms***
MV3D	*LiDAR+Image*	Unnoficial	360ms
MONO3D	*Mono Image*	Yes	4.2s
3DOP	*Stereo Image*	Yes	3s

Table 21: Modes of Operation of Common Architectures

CHAPTER 3

IMPLEMENTATION

This chapter details the execution of the project.

3.1 Datasets

3.1.1 Context Classification

1. **CityScapes** - Cityscapes is a large-scale dataset used to develop pixel and instance level semantic segmentation models. It is composed of a scenes from 50 different cities captured using stereo cameras. In 5000 scenes, the images have been annotated on a pixel-level. 20000 scenes have also been coarsely annotated. In addition to this, a benchmark suite to evaluate these models is available.

2. **KITTI Object Detection**- The KITTI dataset was collected with the aim of encouraging the development of computer vision and robotic algorithms for AVs.The data was captured by a car was fitted with a number of sensors including a high resolution greyscale and colour cameras, Velodyne HDL-64 LiDAR and a GPS/IMU inertial navigation system. The vehicle was driven around a city, rural areas and highways. This dataset is available as part of a benchmark suite for various AV object detection models.

3.1.2 Lidar Performance Evaluation

1. **University of Bristol Smart Internet Lab** - This dataset was captured from a Velodyne VLP-16 LiDAR on stationary vehicle positioned at the Millennium Square in Bristol as part of the connected and autonomous vehicles project. Available as LiDAR network capture files(.pcap).

3.2 Can We Automatically Detect The Context in Driving Scenes?

Determining the context from images and point clouds where no prior information about their location nor context is provided is quite difficult. This is often the case in a lab setting where the data is obtained from public databases.Similarly, images and point clouds from the KITTI dataset did not have any such information. Consequently, creating a context detector that can be used in a lab setting would be very useful.

To begin with, I visually classified 3715 images from the KITTI dataset to be used as training and test data for the context detector. The context of the image scenes were determined using the following criteria. Urban regions were characterised by:

1. High density of road users often including pedestrians, cyclists and vehicles.
2. Presence of large multi-story buildings.
3. Presence of multiple transport facilities such as trams.

Non-urban regions were characterised by:

1. Low/medium density of road users mostly vehicles.
2. Few or no buildings.
3. Long stretches of motorways surrounded by vegetation.

Following this, the classified data was split on a 80:20 ratio to be used as the training and test set respectively.

Figure 31: Urban scene

Figure 32: Non-Urban scene

3.2.1 Image Context Detection

Images provide a rich representation of a scene that can be exploited through image segmentation. Various image segmentation models have been developed to break down images into coherent regions.Semantic segmentation involves classifying these regions into various classes on a pixel level. Classic computer vision techniques for semantic segmentation utilised Texton Forests[35] and Random Forest classifiers [34], however, deep learning techniques have overtaken them in terms of performance. Developed by Google, Deeplab-V3 is one of the best performing open-source models on the Cityscapes semantic segmentation benchmark table.

Classifying Semantic Histograms

Using the Deeplab-V3 network, I performed semantic segmentation on the classified images. For each image, a semantic histogram containing the number of pixels in each semantic class as seen in figure 33 was calculated. By matching these histograms with the context label of the image, I was able to train a Support Vector Machine(SVM) to classify an image. Linear and radial basis functions were tested on the SVM to evaluate their performance.

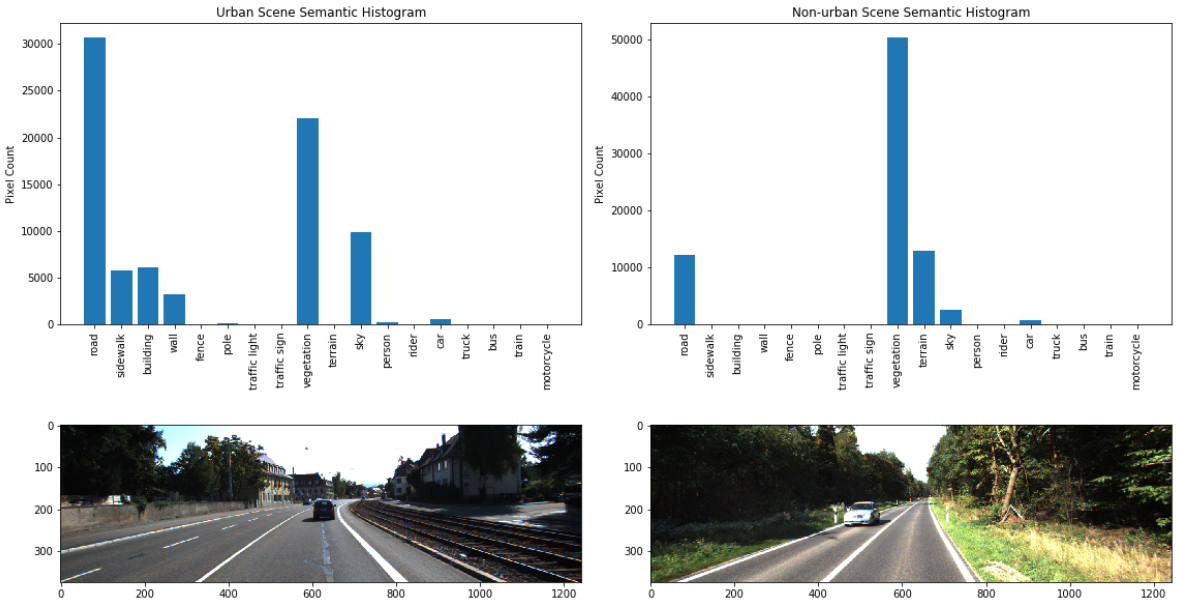

Figure 33: Semantic Histograms

3.2.2 PointCloud Context Detection

As compared to point clouds from RGB-D or stereo cameras, point clouds from LiDAR lack rich features. This is due to the sparse nature of the point clouds and as a result, the resulting representation tends to have a large number of 'holes'. Despite the availability of point clouds segmentation models, there are no datasets with annotations of outdoor scenes such as Cityscapes. As such using semantic histograms to characterise scene contexts in this case would not be suitable. A different approach was therefore used by first detecting the key points in the cloud, extracting and classifying features from them and later on matching these features in other point clouds.

1. **Keypoint extraction**
   Similar to classic image processing, key points are salient points that are not affected by any form of transformation or distortion. As such, they are distinctive and repeatable in

Figure 34: Ambiguous scene

different points-of-view. Currently, only a few 3D keypoint detectors have been proposed. These include Intrinsic Shape Signatures(ISS)[], NARF[] and Uniform Sampling[] of point clouds in voxels(similar to VoxelNet's).

2. **Feature description**
   Signature of Histogram of Orientations

3. **Feature Matching**

3.2.3 Implementation Issues

Despite being able to visually classify a large number of images, some images were difficult to classify as they exhibited characteristics from both contexts. Furthermore, due to the fact that the images was not contiguous , I was unable to infer their context by examining preceding image frames. As such, this affected the performance of both classifiers.

3.3 Do Some Object Detection Methods work better in Different Contexts?

Having been able to successfully classify and create a dataset of urban and non-urban images, the next step was evaluating how a LiDAR based and image+LiDAR based object detection model would perform in different contexts. For this task,the state-of-the-art VoxelNet(LiDAR only) and AVOD(LiDAR+image) proved to be the most suitable candidates. This was influenced by the fact that they both have similar inference times, and both were open-source with the only exception being that the VoxelNet implementation was unnoficial[1]. Both of these models were written in Python and run on the TensorFlow Deep Learning Framework.

3.3.1 VoxelNet

VoxelNet is an end to end point cloud based 3D object detection network that was recently published by leading researchers at Apple Inc. This network was one of the top performing LiDAR only models on the KITTI benchmark. However, the code was never released and only unofficial implementations are currently available online.

[1]Had similar evaluation results as the unreleased original on KITTI benchmark.

3.3.1.1 Architecture Overview

VoxelNet is composed of three fundamental blocks.

1. **Voxel Feature Encoding Layer**

2. **Convolutional Middle Layers**

3. **Region Proposal Network**

3.3.1.2 Implementation Details

3.3.2 AVOD

3.3.2.1 Architecture Overview

3.3.2.2 Implementation Details

3.3.3 Is VoxelNet Performance Valid for Different Datasets?

To validate the VoxelNet model, I used the Smart Internet Lab dataset that was obtained from a LiDAR positioned on a stationary car.

Preprocessing

Before using the data on VoxelNet, the data had to be preprocessed to the necessary input format. VoxelNet only required four input fields, **XYZ values** and **reflectance** of the points, whereas the dataset was in the form of network capture files(.pcap). To convert them into this format, I exported the capture filed into CSV format for each of the frames. This was done in VeloView using the CSV export function. The result was a CSV file with 13 fields as seen in figure 35

Points_m_XYZ:0	Points_m_XYZ:1	Points_m_XYZ:2	X	Y	Z	intensity	laser_id	azimuth	distance_m	adjustedtime	timestamp	vertical_angle
0.01759156584739685	5.304835319519043	-1.421434164047241	0.01759156507648251	5.304835470033373	-1.421434195703044	1.0	0.0	19.0	5.492	3433080897.0	3433080897.0	-15.0
0.07194533199071884	20.6107349395752	0.3597639203071594	0.07194532898218986	20.61073481953913	0.3597639062981624	20.0	1.0	20.0	20.614	3433080900.0	3433080900.0	1.0
0.02194887027144432	5.988438129425049	-1.382549166679382	0.02194887062991644	5.98843819471877	-1.382549179997394	1.0	2.0	21.0	6.146	3433080902.0	3433080902.0	-13.0
0.02645202539861202	6.88900899887085	-1.339097499847412	0.02645202482367224	6.889008789131341	-1.339097529552591	1.0	4.0	22.0	7.018	3433080907.0	3433080907.0	-11.0
0.3046736121177673	75.89746856689453	6.640221595764416	0.3046736085365985	75.89747013826725	6.640221728458581	63.0	5.0	23.0	76.188	3433080909.0	3433080909.0	5.0
0.033701688051223 75	8.045639038085938	-1.274315118789673	0.033701689426747 4	8.045638637731509	-1.274315152217721	1.0	6.0	24.0	8.146	3433080911.0	3433080911.0	-9.0
0.3095385730266571	70.94056701660156	8.710489273071289	0.3095385870564042	70.94056833290217	8.71048945053951	63.0	7.0	25.0	71.474	3433080913.0	3433080913.0	7.0
0.04397721216082573	9.691121101379395	-1.189932227134705	0.04397721326242784	9.691120843322894	-1.18993226900786	1.0	8.0	26.0	9.764	3433080916.0	3433080916.0	-7.0
0.3213215470314026	70.80862426757812	11.2150993347168	0.3213215462835961	70.80862345723331	11.21509966766423	63.0	9.0	26.0	71.69200000000001	3433080918.0	3433080918.0	9.0
0.05691538378596306	12.07773017883301	-1.056676268577576	0.05691538501887421	12.07773041569957	-1.056676225072608	3.0	10.0	27.0	12.124	3433080920.0	3433080920.0	-5.0
0.08184240758419037	16.16960144042969	-0.8474237918853 76	0.08184241059056 33	16.16960230547166	-0.8474238034857464	2.0	12.0	29.0	16.192	3433080925.0	3433080925.0	-3.0

Figure 35: Smart Internet Lab Frame CSV

Out of the 13 fields, I extracted the XYZ values and intensity(reflectance) of the points and saved them as Numpy binary files. This resulted in a smaller file sizes(from around 4MB to 70KB) that are also quicker to read than normal CSV files.

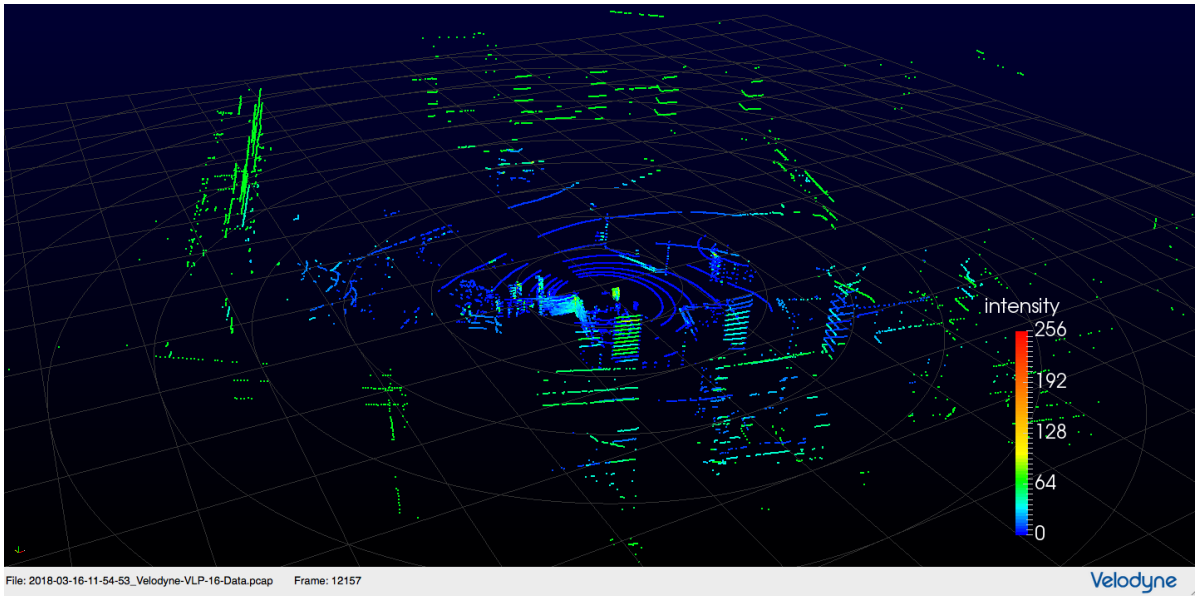

Figure 36: Veloview Frame of Smart Internet Lab data

Finally, I modified VoxelNet's configuration file to include the specifications for the VLP-16 LiDAR sensor. This was essential as the network was initially trained on KITTI point clouds that were obtained from a Velodyne HDL-64E with different specifications as seen in table 31. These specifications are necessary for manipulating the point clouds and therefore directly affect the performance of the network. The modified entries were the angular and vertical resolution, maximum and minimum XYZ values and the height of the VLP-16 LiDAR sensor on the vehicle. Given that there was no image data for visualisation, the image size in the configuration file was set to 0 to prevent unnecessary tensor memory allocation for images.

LiDAR	Hor FOV	Ver FOV	Range	Angular Resolution	Points/second	Channels
HDL-64E	360°	26.9°	120m	~0.4°	~2.2 Million	64
VLP-16	360°	± 15°	100m	0.1°	600,000	16

Table 31: Velodyne HDL-64E and VLP-16 Specifications.

Annotating Ground Truth Labels

The SIL dataset did not contain any ground truth labels for the point clouds as compared to the KITTI dataset. As a result, there was no metric to assess the performance of the network. To prevent this, objects from the dataset needed to be annotated to obtain some ground truth bounding boxes.

L-CAS Annotation Tool

Developed by the Lincoln Centre for Autonomous Systems Research, this tool provides a semi-automatic labelling function that clusters and highlight regions of interest that may contain

objects. The user can then label them depending on the object class such as car, cyclist, pedestrian. The result is then stored in a file containing the objects and their positions in the point cloud. To use this tool, the .pcap capture files were converted into .pcd format using the ROS velodyne point cloud package[2].

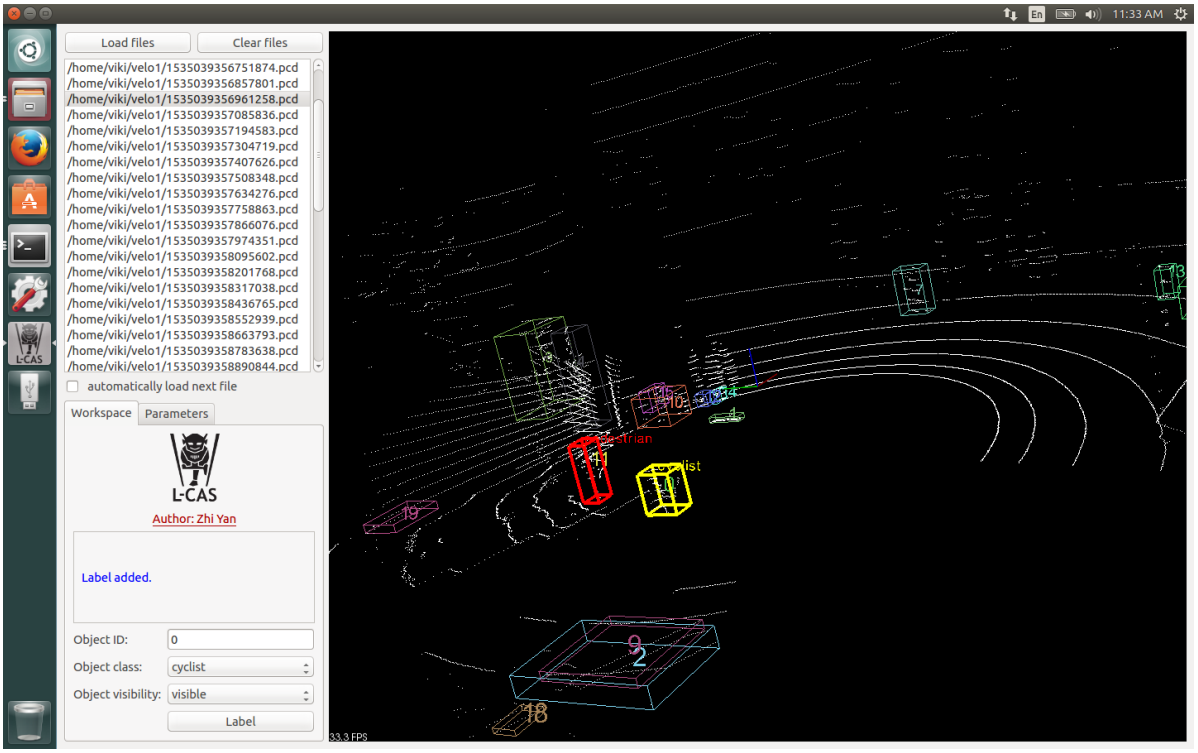

Figure 37: L-CAS Annotation Tool

Following this, the converted input was fed through VoxelNet and the resulting predictions evaluated.

[2]See listing A.1

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Can We Automatically Detect Scene Contexts?

4.1.1 Metrics

Precision, recall and f1-score from the predictions of the detectors were the main performance metrics.

4.1.2 Results

	Context Detection using PointCloud Feature Matching				Context Detection using Image Segmentation			
Context	**precision**	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**	**support**
non-urban	**0.52**	0.45	0.48	206	0.81	**0.9**	**0.85**	193
urban	0.51	**0.58**	**0.55**	206	**0.9**	0.81	**0.85**	218
avg / total	0.51	0.51	0.51	412	0.86	0.85	0.85	411

Table 41: Comparison of context detection methods

4.2 Do Object Detection Models Perform Better in Different Contexts?

Metrics

The models were evaluated using a similar method as the KITTI evaluation protocol. The average precision(AP) of the intersection over union(IoU) of the bounding boxes was used as the main metric. IoU is calculated as:

$IoU = \frac{Overlap area}{Union area}$

Where the overlap area is the region that is shared between the predicted bounding box and the ground truth bounding box and the Union area is the total area of the predicted bounding box and ground truth box. This was calculated for 2D bounding boxes, 3D bounding boxes and Bird-Eye View bounding boxes. Only IoU values ≥0.7 were considered for the car class and ≥0.5 for the pedestrian class in calculating the AP.

4.2.0.1 GPU

For uniformity purposes, all the experiments were performed on single NVIDIA P100 GPU with 16GB High Bandwidth Memory. During inference, GPU statistics were collected at an interval of 1 second using NVIDIA System Management Interface(nvidia-smi). The following metrics were collected.

- **Power Draw** - Watts
- **Memory Used** - MB
- **Temperature** - °C
- **Clock Speed** - MHz

4.2.1 Results

Baseline

	Class	Car			Pedestrian		
Model	**Difficulty**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
AVOD	***2D Bounding Box***	**86.987999**	**77.10569**	**68.012459**	56.214417	51.969517	46.578247
	BEV Bounding Box	**86.00412**	**74.355942**	**65.62397**	**50.94368**	**49.857227**	**44.704525**
	3D Bounding Box	**75.447769**	**63.742085**	**54.334251**	**48.757492**	**44.84359**	**43.014023**
VoxelNet	***2D Bounding Box***	65.951279	56.683437	55.972511	**66.309807**	**60.556213**	**53.479141**
	BEV Bounding Box	81.242409	70.672707	65.347595	50.308723	46.726196	41.24551
	3D Bounding Box	40.000149	34.892159	31.376112	34.161884	30.576441	28.982103

Table 42: Baseline

4.2.1.1 Running Each Model Exclusively

Each model was run on a single context dataset exclusively on the GPU.

	Context	Urban			Non-Urban		
Model	**Difficulty**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
AVOD	***2D Bounding Box***	**86.987999**	**77.10569**	**68.012459**	**89.186172**	**79.754562**	**78.550514**
	BEV Bounding Box	86.00412	74.355942	65.62397	87.11274	**76.859818**	**75.720955**
	3D Bounding Box	**75.447769**	**63.742085**	**54.334251**	**75.430656**	**64.199951**	**62.902302**
VoxelNet	***2D Bounding Box***	69.597939	65.98201	59.284454	77.520409	67.733231	62.281651
	BEV Bounding Box	**86.34037**	**76.187859**	**68.103294**	**88.635025**	75.361214	69.502548
	3D Bounding Box	73.632263	58.473991	50.744587	68.620865	49.455608	45.809917

Table 43: Car AP

	Context	Urban			Non-Urban		
Model	**Difficulty**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
AVOD	***2D Bounding Box***	**78.626633**	**77.407684**	**70.029228**	76.096977	70.946465	71.205276
	BEV Bounding Box	**80.880447**	**80.393105**	**79.296593**	77.806831	77.654358	71.792923
	3D Bounding Box	**80.649719**	**80.142876**	**79.056778**	77.538368	71.868484	71.564285
VoxelNet	***2D Bounding Box***	72.129387	65.245384	65.508232	**84.218407**	**76.52832**	**76.593803**
	BEV Bounding Box	74.766678	73.944069	74.138588	**89.830276**	**80.201164**	**80.114815**
	3D Bounding Box	71.177628	64.247559	64.18145	**87.206612**	**77.964493**	**78.036652**

Table 44: Pedestrian AP

Context	**VoxelNet**			**AVOD**		
	min	**max**	**mean**	**min**	**max**	**mean**
Urban	0.113	3.813	0.129	**0.095**	**2.457**	**0.112**
Non-urban	0.113	**2.224**	0.127	**0.096**	2.506	**0.113**

Table 45: Inference Time on Single NVIDIA P100 GPU

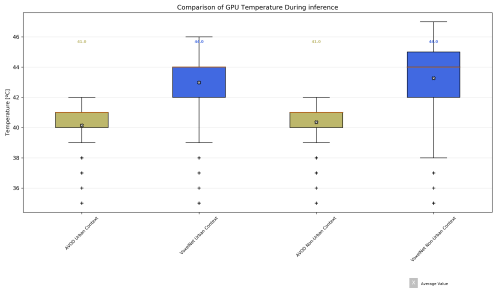

Figure 41: Temperature

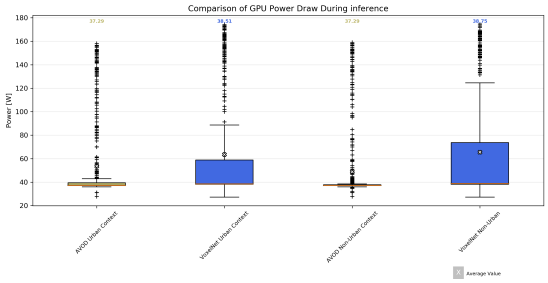

Figure 42: Power

Context	**Urban**		**Non-urban**	
Model	**Max**	**Mean**	**Max**	**Mean**
AVOD	14832	14762.815	14320	14241.497
VoxelNet	14832	14031.075	14832	13808.981

Table 46: Memory

4.2.1.2 Running Both Models Simultaneously

To investigate whether both models are capable of running on a single GPU, both models were simultaneously ran on a single NVIDIA P100 GPU. In terms of power, running both models saw no notable difference in the mean power drawn as was the the spread of values was similar as can be seen in figure 44. With regard to temperature, the mean temperature was higher by about 2°C .However, the spread was relatively similar.

Memory-wise, the both models consumed a mean of 14.5GB and 14.1GB when running on the urban and non-urban datasets respectively.

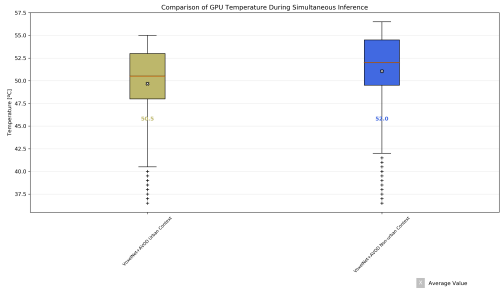

Figure 43: Temperature

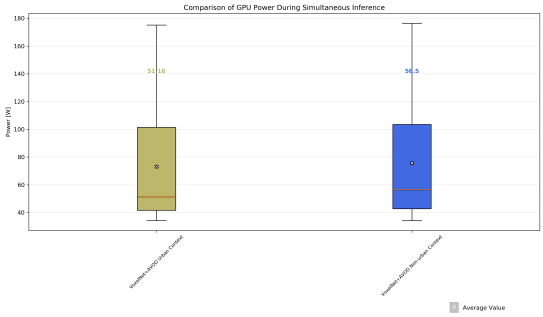

Figure 44: Power

Table 47: Memory Usage During Simultaneous Inference

Context	Min	Max	Mean
Urban	4612.5	14832	14474.9115
Non-urban	**4608.5**	**14320**	**14078.13272**

4.3 Is VoxelNet Performance Valid For Different Datasets?

Metrics

For this evaluation, AP was used as the main performance metric. However, as there was no corresponding image data for the point clouds, only the 3D bounding boxes and Bird-Eye View bounding boxes were used in calculating the AP. Given that the data was collected from a stationary vehicle in an urban area with mostly pedestrians the dataset was evaluated on the pedestrian class only. IoU values ≥ 0.5 were used in calculating the AP.

4.4 Results

CHAPTER 5

FUTURE WORK

CHAPTER 6

CONCLUSION

APPENDIX A

LiDAR	Hor FOV	Ver FOV	Range	Angular Resolution	Points/second	Channels
VLS-128	360°	+15°to -25°	300m	0.11°	~9.6 Million	128
HDL-64E	360°	26.9°	120m	~0.4°	~2.2 Million	64
HDL-32E	360°	+10°to -30°	80m-100m	0.1°	~1.39 Million	32
VLP-32C	360°	+15°to -25°	200m	0.1°	~1.2 Million	32
VLP-16	360°	± 15°	100m	0.1°	600,000	16

Table A1: Velodyne LiDAR Family

```
#Code to read .pcap files and convert them to .pcd

#Load existing pcap file and create a sensor publisher
$ roslaunch velodyne_pointcloud VLP-16-points.launch pcap:=(path to .pcap file)

#Receiver to  convert sensor messages to .pcd (In separate terminal window)
$ rosrun pcl_ros pointcloud_to_pcd input:=/velodyne_points _prefix:=(path to save it)

```

Listing A.1: .pcap -> .pcd

BIBLIOGRAPHY

[1] M. ABADI, 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 OSDI, vol. 16, 2016, pp. 265–283.

[2] H. BAY, T. TUYTELAARS, AND L. VAN GOOL, *Surf: Speeded up robust features*, in European conference on computer vision, Springer, 2006, pp. 404–417.

[3] J.-F. BONNEFON, A. SHARIFF, AND I. RAHWAN, *Autonomous vehicles need experimental ethics: Are we ready for utilitarian cars?*, arXiv preprint arXiv:1510.03346, (2015).

[4] I. BROWN, *Human factors: the journal of the human factors and ergonomics society*, Driver Fatigue, 36 (1994), pp. 298–314.

[5] S. D. BROWN, R. J. FRANCIS, J. ROSE, AND Z. G. VRANESIC, *Field-programmable gate arrays*, vol. 180, Springer Science & Business Media, 2012.

[6] M. BUEHLER, K. IAGNEMMA, AND S. SINGH, *The DARPA urban challenge: autonomous vehicles in city traffic*, vol. 56, springer, 2009.

[7] L.-C. CHEN, G. PAPANDREOU, I. KOKKINOS, K. MURPHY, AND A. L. YUILLE, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, IEEE transactions on pattern analysis and machine intelligence, 40 (2018), pp. 834–848.

[8] X. CHEN, K. KUNDU, Z. ZHANG, H. MA, S. FIDLER, AND R. URTASUN, *Monocular 3d object detection for autonomous driving*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2147–2156.

[9] X. CHEN, K. KUNDU, Y. ZHU, H. MA, S. FIDLER, AND R. URTASUN, *3d object proposals using stereo imagery for accurate object class detection*, IEEE transactions on pattern analysis and machine intelligence, 40 (2018), pp. 1259–1272.

[10] X. CHEN, H. MA, J. WAN, B. LI, AND T. XIA, *Multi-view 3d object detection network for autonomous driving*, in IEEE CVPR, vol. 1, 2017, p. 3.

[11] N. DALAL AND B. TRIGGS, *Histograms of oriented gradients for human detection*, in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, IEEE, 2005, pp. 886–893.

[12] D. J. FAGNANT AND K. KOCKELMAN, *Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations*, Transportation Research Part A: Policy and Practice, 77 (2015), pp. 167–181.

[13] T. M. GASSER, *Fundamental and special legal questions for autonomous vehicles*, in Autonomous Driving, Springer, 2016, pp. 523–551.

[14] A. GEIGER, P. LENZ, C. STILLER, AND R. URTASUN, *Vision meets robotics: The KITTI dataset*, International Journal of Robotics Research (IJRR), (2013).

[15] A. GEIGER, P. LENZ, AND R. URTASUN, *Are we ready for autonomous driving? the kitti vision benchmark suite*, in Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[16] R. GIRSHICK, *Fast r-cnn*, in International Conference on Computer Vision (ICCV), 2015.

[17] P. HENRY, M. KRAININ, E. HERBST, X. REN, AND D. FOX, *Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments*, in In the 12th International Symposium on Experimental Robotics (ISER, Citeseer, 2010.

[18] INTEL, 2018.
https://www.intel.co.uk/content/www/uk/en/automotive/go-automated-driving.html.

[19] J. KU, A. HARAKEH, AND S. L. WASLANDER, *In defense of classical image processing: Fast depth completion on the cpu*, arXiv preprint arXiv:1802.00036, (2018).

[20] J. KU, M. MOZIFIAN, J. LEE, A. HARAKEH, AND S. WASLANDER, *Joint 3d proposal generation and object detection from view aggregation*, arXiv preprint arXiv:1712.02294, (2017).

[21] S. LAWRENCE, C. L. GILES, A. C. TSOI, AND A. D. BACK, *Face recognition: A convolutional neural-network approach*, IEEE transactions on neural networks, 8 (1997), pp. 98–113.

[22] B. LI, *3d fully convolutional network for vehicle detection in point cloud*, arXiv preprint arXiv:1611.08069, (2016).

[23] B. LI, T. ZHANG, AND T. XIA, *Vehicle detection from 3d lidar using fully convolutional network*, arXiv preprint arXiv:1608.07916, (2016).

[24] S.-C. LIN, Y. ZHANG, C.-H. HSU, M. SKACH, M. E. HAQUE, L. TANG, AND J. MARS, *The architectural implications of autonomous driving: Constraints and acceleration*, in Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ACM, 2018, pp. 751–766.

[25] D. G. LOWE, *Object recognition from local scale-invariant features*, in Computer vision, 1999. The proceedings of the seventh IEEE international conference on, vol. 2, Ieee, 1999, pp. 1150–1157.

[26] L. MEDSKER AND L. JAIN, *Recurrent neural networks*, Design and Applications, 5 (2001).

[27] MOBILEYE, *Enabling autonomous - mobileeye*, 2018. https://www.mobileye.com/future-of-mobility/mobileye-enabling-autonomous/.

[28] A. NEWELL AND S. K. CARD, *The prospects for psychological science in human-computer interaction*, Human-computer interaction, 1 (1985), pp. 209–242.

[29] D. A. POMERLEAU, *Alvinn: An autonomous land vehicle in a neural network*, in Advances in neural information processing systems, 1989, pp. 305–313.

[30] C. R. QI, H. SU, K. MO, AND L. J. GUIBAS, *net: Deep learning on point sets for 3d classification and segmentation*, Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 1 (2017), p. 4.

[31] C. R. QI, L. YI, H. SU, AND L. J. GUIBAS, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, in Advances in Neural Information Processing Systems, 2017, pp. 5105–5114.

[32] S. REN, K. HE, R. GIRSHICK, AND J. SUN, *Faster r-cnn: Towards real-time object detection with region proposal networks*, in Advances in neural information processing systems, 2015, pp. 91–99.

[33] N. RESEARCH, *Navigant research leaderboard: Automated driving vehicles*, Jan 2018. https://www.navigantresearch.com/research/navigant-research-leaderboard-automated-driving-vehicles.

[34] J. SHOTTON, A. FITZGIBBON, M. COOK, T. SHARP, M. FINOCCHIO, R. MOORE, A. KIPMAN, AND A. BLAKE, *Real-time human pose recognition in parts from single depth images*, in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, Ieee, 2011, pp. 1297–1304.

[35] J. SHOTTON, M. JOHNSON, AND R. CIPOLLA, *Semantic texton forests for image categorization and segmentation*, in Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.

[36] M. SIMON, S. MILZ, K. AMENDE, AND H.-M. GROSS, *Complex-YOLO: Real-time 3D Object Detection on Point Clouds*, ArXiv e-prints, (2018).

[37] M. J. S. SMITH, *Application-specific integrated circuits*, vol. 7, Addison-Wesley Reading, MA, 1997.

[38] P. VIOLA AND M. JONES, *Rapid object detection using a boosted cascade of simple features*, in Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, IEEE, 2001, pp. I–I.

[39] WAYMO, *Waymo safety report*, 2018.
https://waymo.com/safety/.

[40] F. YANG, W. CHOI, AND Y. LIN, *Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2129–2137.

[41] M. D. ZEILER AND R. FERGUS, *Visualizing and understanding convolutional networks*, in European conference on computer vision, Springer, 2014, pp. 818–833.

[42] Y. ZHOU AND O. TUZEL, *Voxelnet: End-to-end learning for point cloud based 3d object detection*, arXiv preprint arXiv:1711.06396, (2017).