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 [16]. LiDAR presents objects as point clouds in a 3D space thus offering critical shape information of objects in view. However, this representation is sparse and suffers from drawbacks such as occlusion. As a result, LiDAR-based detection performs poorly as compared to multimodal methods that have helped overcome this.[32]. However, multimodal methods are often complex to set up and synchronise [16] with the cost of components running into thousands of pounds.

The aim of this project is to replicate and improve on the novel VoxelNet by Zhou et al [32]. Doing so will be of added value as the implementation is currently proprietary to Apple Inc. If successful, my implementation will be made open source and can be used for further research in the field of object detection from LiDAR point clouds. In addition, through the evaluation and analysis, I hope to propose ways in which my implementation can be used in making cheaper AVs through the use of Solid-state LiDAR without having to use cameras at all.

In light of this, twos main objectives emerge. The first is to develop a region proposal network for object detection from LiDAR point clouds. This will be implementred using the TensorFlow [**?**] machine learning framework. The second will be a detailed evaluation report containing research into this topic area as well as an analysis and comparison of the performance of my implementation.

This project will be 65% type I (Software Development) and 35% type II (Theoretical). Software development of the deep learning framework will involve obtaining AV videos and point cloud data from public datasets such as KITTI [11]. This data will then be used in the development and training of the RPN. On completion of development, the model will be evaluated against the KITTI benchmark [12].

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

Table **Page**

List of Figures

CHAPTER 1

INTRODUCTION

In the coming decade there is expected to be proliferation of new technologies that have been aided by recent developments in Artificial Intelligence(AI) and Machine Learning(ML). AI and ML have become topics of increasing interest in academic fields and industries globally. As a result, collaborations between these two fields has become increasingly common. With increase in computational power and the development of newer AI algorithms robotic technologies have greatly advanced and are now able to perform complex tasks that were once too difficult, dangerous or even impossible for humans to perform. Consequently, industries have benefited from the accuracy and efficiency provided by these robots.

An area that has developed a wide range of interest currently is the topic of autonomous vehicles. The idea of autonomous vehicles(AVs) is not new and as early as 2005, DARPA had invested heavily in the creation of unmanned trucks and organised for the Urban Challenge [7] 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 there are a number of companies that are developing AVs . These companies have been able to develop vehicles that are capable of certain levels of automation. These levels are defined as:

- **Level 0** - No autonomy.
- **Level 1** - Basic driver assistance built into vehicle design.
- **Level 2** - Partially autonomous but driver expected to monitor environment at all times.

- **Level 3** - Conditionally autonomous with the driver not required to monitor the environment but is required to take back control if need be.
- **Level 4** - Highly autonomous with the vehicle capable of handling most conditions but the driver has the option to take control.
- **level 5** - Completely autonomous with the vehicle capable of handling all conditions.

1.1 Why Autonomous Cars

1.1.1 Safety

A major argument for self driving cars is the improved safety that they will provide on the road. According to a report released by the United States Department of Transport, National Highway Traffic Safety Administration(NHTSA), around 6,296,000 crashes occured in the year 2015 with 35,092 people losing their lives in these crashes. Of these crashes, around 94% of them were as a result of human error. Worldwide, it is estimated that there were 1.2 million deaths in 2013 due to road crashes. In light of this, self driving cars could greatly reduce the number of road crashes. According to the Eno Centre of Transportation, if 90% of the cars on the road were autonomous, there would be a reduction of 4,220,000 road crashes, this would save 21,700 lives. This is based on the fact that a large number of road crashes are a as a result of human error and therefore autonomous vehicles will be able to significantly reduce this number. However, this estimate is dependent on how well the AV system is designed to be able to handle complex, dynamic driving situations.

1.1.2 Environmental Impact

According to the Environmental Protection Agency, more than a quarter of greenhouse gases are from the transportation sector. A major contributor of this is traffic congestion due to various factors such as traffic destabilizing shockwave propagation and road accidents. Autonomous vehicles are able to mitigate this as they are able to gauge and calculate the motion vectors of different objects around them. By using traffic smoothing algorithms and smarter implementations such as the slot mechanism developed by MIT, traffic congestion can be greatly reduced and as a result lower fuel consumption. Furthermore, most companies working with AVs are moving towards electrical vehicles and thus reducing the impact of fossil fuels on the environment.

1.2 Aims and Objectives

This project aims to evaluate and develop a novel solution to aid perception in AVs by developing an end to end region proposal network capable of accurate object detection from LiDAR point clouds.

To achieve this aim, the following objectives will be undertaken:

1. Detailed analysis of state-of-the-art LiDAR-Based object detection deep neural networks.
2. Implement voxel feature encoding layer for grouping point clouds
3. Implement region proposal network for object detection from voxels.
4. Test and evaluate the implemented neural network against results of state-of-the-art point cloud object detection methods.

1.3 Deliverables

The deliverables are split into technical and analytical.

- **End to end point cloud object detection RPN.** This will be a software implementation of the system that will be publicly available through a Github repository.
- **Evaluation report.** In this report, the following topics will be discussed.
    1. A review of related research and implementations tackling object detection using LiDAR cloud points.
    2. Performance analysis of system and analysis criteria.
    3. A comparison between the implemented system and other state-of-the-art detection systems, potentially through a public benchmark.
    4. The ethical and safety implications of the system and its viability in a real world setting.
    5. Economic analysis of the LiDAR-based system and its potential impact on the development of AVs.
    6. Validation of DNN performance with other public datasets containing data from AVs.

1.4 Added Value

This project will implement and open source a sophisticated cloud point detection technique for use in autonomous vehicles. The implementation will be able to detect classes of objects such as cyclists, pedestrians and other vehicles on a real-time basis. In doing so, the project will democratize access to proprietary technology by Apple [32] to be used and developed further by future researchers working with object detection in point clouds. Following the detailed evaluation, I intend to propose ways through which this project can be implemented to reduce the cost of AVs in order to make them more viable. If successful, this project will provide a possible framework for the main stream adoption of AVs.

1.5 Research scope

The focus of this project is mainly with regard to computer vision, deep learning and robotics. Computer vision is the task of obtaining, processing, analysing and contextualising visual information to produce numerical information that can be understood and manipulated by computers. This is necessary in order to process and analyse the LiDAR data. Deep learning is a broad term used to describe methods that utilise the use of deep neural networks that have a large number of layers. DNNs have become a heavily researched an invested area due to their ability to capture complex underlying models from data. This will be crucial for detecting objects from point clouds. This project will combine existing research using computer vision and deep learning such as [24][32] to develop a Region Proposal Network capable of accurately detecting objects in point clouds.

1.6 Report structure

Chapter 2

Literature Review

2.1 Components of a Self Driving Car

Most self-driving cars consist of 4 main components:

- **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.

  $$distance = \frac{time \times \text{speed of light}}{2}$$

  To achieve a high level of accuracy, a LiDAR unit has to send out a large enough number of lasers in different directions fast enough to create an accurate point cloud representation of the environment around it. To do so, LiDAR units have multiple channels(emitter/receiver pairs) angled vertically that emit hundreds of thousands of laser pulses per second.

  LiDAR units require complex optical systems that are expensive to build. As a result, they are the most expensive sensor in AVs with the top end such as Velodyne HDL-64E shown in figure 2.1 costing around 75,000$. More channels allow for a more accurate representation of the surrounding environment which is necessary for safer navigation of AVs. However it is worth noting that the cost of LiDAR units increases proportionally with the number of channels. As such, different companies have are exploring different design methods that are cheaper but still able to offer the same performance as the top end LiDAR units. This has resulted in the development of different types of LiDAR units as shown below.

  - **Mechanical Mirror**
  - **Solid State**
  - **Optical Phase Array**
  - **Microelectromechanical systems (MEMS)**
  - **3D Flash**

Figure 2.1: Velodyne LiDAR family. From left: Velodyne HDL-64E , HDL-32E , PUCK

- **Cameras** - Cameras mounted on the vehicle are used for classification and identification of various objects on the road. This is important for recognising traffic rules from traffic signs or road markings as well as determining the nature of 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 [14] 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.
- **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

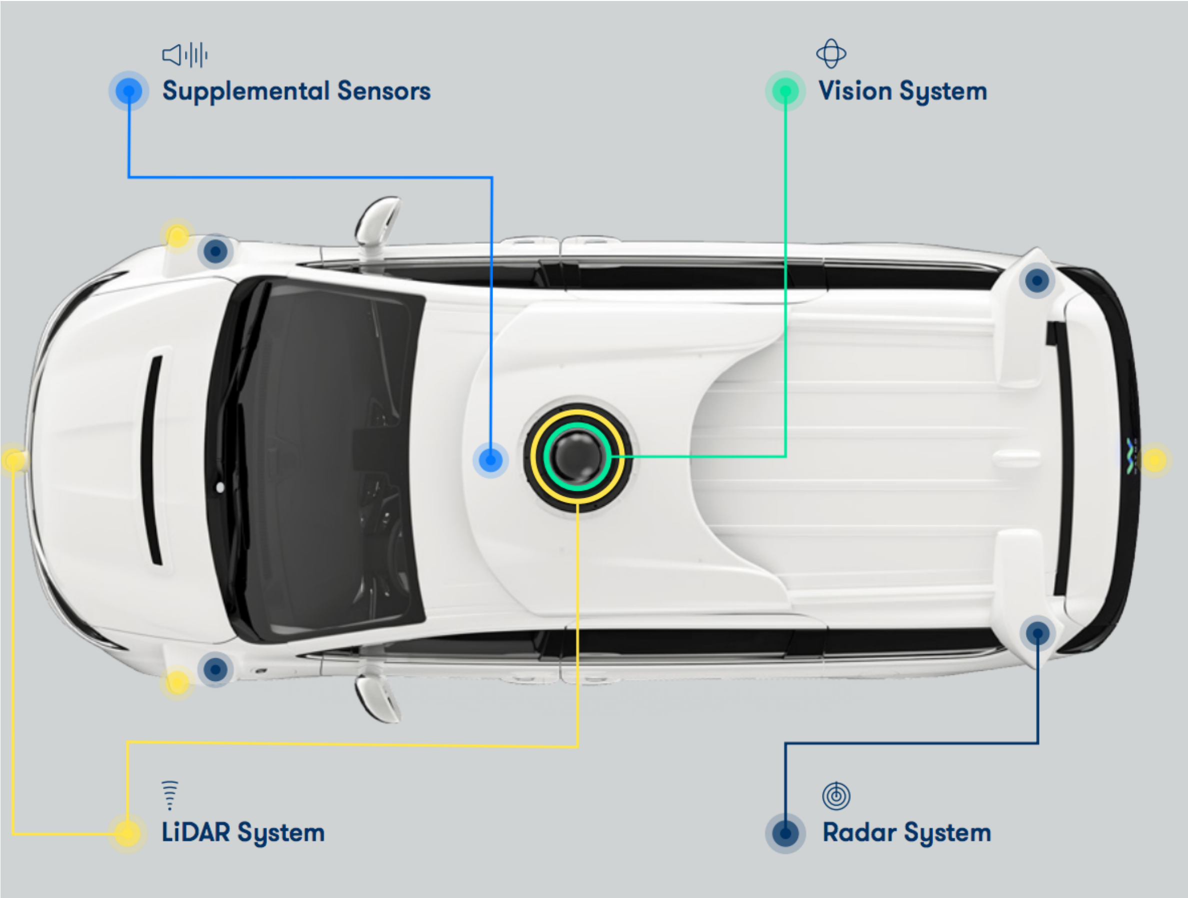

Figure 2.2: Components of Waymo's Self Driving Car [30]

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 the sensors in the vehicle, AVs require powerful processing units in order to be able to process all this data in real time. Most of the 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)[6], Application Specific Integrated Circuits(ASICs)[28] or combinations with each other.

2.1.1 Legal, Ethical and Economic Considerations

The classic ethical dilemma for self driving cars poses a scenario whereby AVs are presented with a situation whereby a fatal accident is inevitable. For example, the AV either has to crash into a group of people in order to save the life of a passenger or to crash itself and sacrifice the life of the passenger. This dilemma highlights important legal, ethical and economic considerations to be considered by companies involved in the production of AVs and their corresponding systems.

According to [10], there are no public laws that cover the use of independent autonomous vehicles in public spaces. In his article, he cites the fundamental issue as "understanding and accepting the effect of AVs as an independent action by a machine". Due to the lack of public laws on their use, he proposes extending fundamental rights such as the right to life into the framework for creating laws that cover emerging technologies that have an effect on the public, that are otherwise not accounted for in traditional laws.

Bearing this in mind, it is important to note that despite improvements in traffic safety over the years, 94% are still caused by human beings with a large majority of them being fatal with the current human error rate being 1 in 100 million miles. This creates a realistic baseline that can then be extended in evaluating the performance of AV systems. As such, if the risk of automation is lesser than the risk of human vehicle control then the AV would be beneficial. Consequently, the AVs are not to be considered as perfect systems.

With regard to economic considerations, car manufucturers involved in the production of AVs have to ensure that their AVs are able to handle numerous scenarios even if they are rare or considered statistically impossible. They should also be required to take legal liability in case any of their system components are defective and result in failures or accidents. This is important for customer trust without which they will not be able to convince customers to shift to AVs. In addition, for the adoption of AVs to be widespread, they have to be reasonably priced and efficient. With the current prices of the the various components and their power consumption, the adoption of AVs at the moment is not viable.

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 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 the 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 an effort to achieve the forementioned requirements, different companies have used different approaches in order to do so. However, a few constraints have to be considered as outlined in [19]

- **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. [5], 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 [22], 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.

- **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 to 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, is one of the leading industry players in the development of AV. 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. Intel in it's part has been developing an a

2.2.2 Camera and Radar

A major industry player that is using the camera and radar configuration is Tesla. Elon Musk, the founder, believes that LiDAR is not necessary for AV perception. However, following a recent accident whereby a passenger was killed in a crash while the vehicle was on 'autopilot', his statement was critiqued as LiDAR could have detected the presence of the woman. As mentioned in the discussion of components, Radar does not perform particularly well with organic objects due to their low reflectivity. Furthermore, due to the fact that it was night time, the cameras had a low visibility range and therefore this system configuration was not robust enough.

2.2.3 LiDAR, Camera and Radar

Waymo, GM Cruise and Uber all use a combination of LiDAR, camera and radar. This combination 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.

2.2.4 Accelerator Architectures

Accelerator architectures can be described as the systems responsible for processing the data from the sensor units. These systems are used to run ML/AI algorithms that are used for the perception, planning and control pipeline.

2.2.4.1 CPUs

With the

2.2.4.2 GPUs

GPUs are the most common CPUs, GPUs, FPGAs, ASICs all offer differe

2.3 Related Research

Following the discussions in the previous sections it is clear that systems to be used in AV have to fulfill certain requirements, namely:

- **Robust**
- **Reproducible**
- **Validatable**
- **Viable**

As the aim of the project is to develop an object detection region proposal network for LiDAR object detection, this section will review research in the area of point cloud object detection. These papers will be analysed against the list of requirements provided above.

2.3.1 Classic Computer Vision

With regard to perception, classic computer vision methods have advanced over the years from simple algorithms such as SIFT[20] and SURF[3] which use descriptors to detect interesting point in images for object detection to much higher level representations such as the Histogram of Gradient(HOG)[8] or Haar Features[29] that were uses in classifiers. However, these methods have been unreliably slow and not as accurate. In addition these features are only able to model 2D data and therefore when applied to 3D data such as point clouds they are unusable. In addition, these feature detectors have to be handcrafted manually which is a tedious process. As a result, such classical methods can not be used in AVs as they do not fulfill any of the requirements above.

2.3.2 Deep Neural Networks

The emergence and development of Deep Neural Networks such as Convolutional Neural Networks(CNNs)[17] 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 Vision

Stereo Vision

RGB-D

LiDAR/Camera

LiDAR

Deep neural networks overcame the problem of having to handcraft features to be used in a classifier. Their ability to automatically learn features has been greatly influential However, this has only been applicable to dense 2D data from cameras and not to sparse 3D data from LiDAR and RGB-D sensors that create cloud points. Cloud points are a geometric representation of unordered set of points. This representation is 3D in nature and as such contains important depth information that can be used to measure the size and shape of objects. Initial attempts to work with this data required hand crafted methods to first represent this information on a 2D plane before running the computer vision algorithms on these representations.

2.4 Deep Learning Framework

A recurring theme that is central to the operation of AVs is object detection. Object detection is crucial for safe operation of AVs as it forms the first step before any planning and control. To achieve real-time results, deep learning techniques are being developed for this task. As mentioned in the previous chapter one of the main deliverables will be an end to end RPN model for detecting objects in point clouds. In order to develop this I will make use of a deep learning framework. At the moment, there are various distributions of deep learning frameworks that are available for different programming languages. A few popular ones include:

Framework	Programming Language(s)
Tensorflow	Python, C++ , R
PyTorch	Python
Caffe	C, C++, Python, MATLAB
MXNet	Python, C++, R, Julia
Microsoft Cognitive Toolkit/CNTK	Python, C++

Table 2.1: Popular deep learning frameworks

I intend to use Tensorflow on Python to implement the RPN. As a RPN is a form of CNN, the next sub section will detail the process of developing a CNN on TensorFlow.

2.4.1 TensorFlow CNN

```
def cnn_model_fn(features, labels, mode):
"""Model function for CNN."""
# Input Layer
input_layer = tf.reshape(features["x"], [-1, 28, 28, 1])

# Convolutional Layer #1
conv1 = tf.layers.conv2d(
```

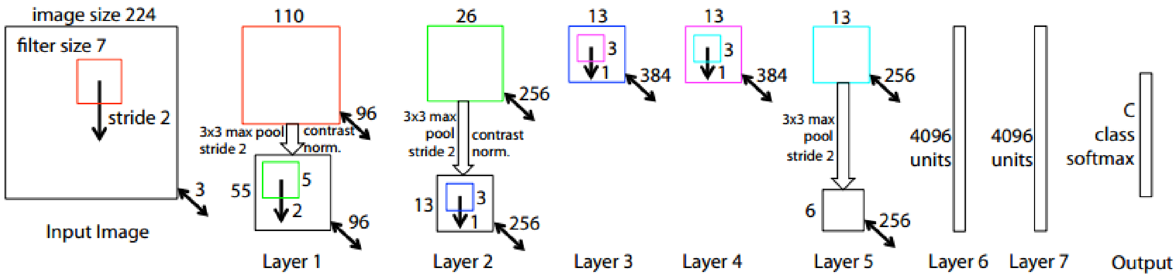

Figure 2.3: CNN Visualisation [31]

```
inputs=input_layer,
filters=32,
kernel_size=[5, 5],
padding="same",
activation=tf.nn.relu)

# Pooling Layer #1
pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], strides=2)

# Convolutional Layer #2 and Pooling Layer #2
conv2 = tf.layers.conv2d(
inputs=pool1,
filters=64,
kernel_size=[5, 5],
padding="same",
activation=tf.nn.relu)
pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=[2, 2], strides=2)

# Dense Layer
pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 64])
dense = tf.layers.dense(inputs=pool2_flat, units=1024, activation=tf.nn.relu)
dropout = tf.layers.dropout(
inputs=dense, rate=0.4, training=mode == tf.estimator.ModeKeys.TRAIN)

# Logits Layer
logits = tf.layers.dense(inputs=dropout, units=10)

predictions = {
# Generate predictions (for PREDICT and EVAL mode)
"classes": tf.argmax(input=logits, axis=1),
# Add `softmax_tensor` to the graph. It is used for PREDICT and by the
# `logging_hook`.
"probabilities": tf.nn.softmax(logits, name="softmax_tensor")
}

if mode == tf.estimator.ModeKeys.PREDICT:
```

```python
return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)

# Calculate Loss (for both TRAIN and EVAL modes)
loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)

# Configure the Training Op (for TRAIN mode)
if mode == tf.estimator.ModeKeys.TRAIN:
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
train_op = optimizer.minimize(
loss=loss,
global_step=tf.train.get_global_step())
return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)

# Add evaluation metrics (for EVAL mode)
eval_metric_ops = {
"accuracy": tf.metrics.accuracy(
labels=labels, predictions=predictions["classes"])}
return tf.estimator.EstimatorSpec(
mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)
```

Listing 2.1: Simple CNN implementation using Tensorflow in Python [2]

1. Convolution layer
2. Pooling Layer
3. Dense Layer
4.
5.

CHAPTER 3

PROJECT OUTLINE

3.1 Description of tasks

1. **Project environment setup**

    a) Download KITTI Dataset

    b) Split into training and test data

    c) Install Tensorflow and necessary libraries

    d) Review related implementations

2. **DNN implementation**

    a) Transform data into valid input

    b) Develop Voxel Feature Encoding Layer

    c) Develop CNN for object detection

    d) Integrate VFE and CNN layers
    Milestone: Prototype RPN model

    e) Add functionality for object classes

    f) RPN Training
    Milestone -Publish source code

    Deliverable: Robust and accurate object detection RPN.

3. **Testing and debugging**

    a) Initial testing

   b) Initial debugging
   c) Test added functionality
   d) Debug added functionality

4. **Parameter tuning**
   a) Compile parameters to test
   b) Run model with parameters
   c) Compile results
   Milestone - Obtain best performing parameters

5. **Evaluation**
   a) Obtain results of related implementations
   b) Run benchmark tests
   c) Compare results

6. **Validation with different datasets**
   a) Download extra datasets
   b) Split data into training and test data
   c) Transform data into valid input
   d) Evaluate model performance.

   Deliverable - Performance analysis of model against different datasets.

7. **Report**
   a) Write Up system design
   b) Compile Evaluation results
   c) Compile validation results
   d) Analyse results
   e) Editing and Formatting

   Deliverable - Final thesis document.

8. **Poster**
   a) Poster design
   b) Editing and formatting

   Deliverable - Final poster

3.2 Timeline and Risk Anaysis

Risk	Likelihood	Severity	Prevention	Contingency
Software Library incompatibility	4	2	Ensure necessary libraries and dependencies are installed beforehand	Use of ready instances on cloud platforms.
Loss of data	1	3	Frequent backups to GitHub and other local machines.	Datasets can be redownloaded from public websites.
Overestimated technical capability	3	2	Review related implementations and literature before beginning project	Large support community for Python and Tensorflow online and within university department.
Illness or injury	2	3	Avoid extreme sports and activities	Ensure adequate time has been allocated in timeline to account for unexpected disruptions.
Long GPU queue on Blue Crystal4	4	3	Queue work during low peak times	Use cloud instances for training

Table 3.1: Risk Analysis Table

teamgantt
Created with Free Edition

Printed: 04/20/2018

Research Outline	start	end
Environment setup	**28/05/18**	**06/06/18**
Download KITTI Dataset	28/05	28/05
Split into training and test data	28/05	29/05
Install Tensorflow and necessa...	28/05	30/05
Review related implementations	28/05	06/06
Deep Neural Network	**07/06/18**	**07/09/18**
Transform data into valid input	07/06	15/06
Develop Voxel Feature Encodi...	18/06	13/07
Develop CNN for object detecti...	18/06	27/07
Integrate VFE and CNN layers	09/07	03/08
Prototype CNN Model	06/08	06/08
Add functionality for object cla...	06/08	10/08
CNN training	02/07	24/08
Publish source code	07/09	07/09
Parameter Tuning	**06/08/18**	**24/08/18**
Compile parameters to test	06/08	08/08
Run model with parameters	09/08	17/08
Compile results	20/08	23/08
Obtain best performing parame...	24/08	24/08
Testing and Debugging	**30/07/18**	**10/08/18**
Initial tests	30/07	03/08
Debug initial model	30/07	30/07
Added functionality tests	06/08	10/08
Debug model with added functi...	06/08	10/08
Evaluation	**28/05/18**	**24/08/18**
Obtain results of other related ...	28/05	31/05
Compare result of papers agai...	24/08	24/08
Validation	**27/08/18**	**31/08/18**
Download extra datasets	27/08	27/08
Split datasets	27/08	27/08
Transform data into valid input	27/08	28/08
Evaluate performance	27/08	31/08
Visualisation	**13/08/18**	**17/08/18**
Design user interface	13/08	14/08
Overlay detected objects on c...	13/08	17/08
Report	**02/07/18**	**07/09/18**
Compilation of evaluation resul...	02/07	31/08
Write up system design	07/08	31/08
Compile validation results	27/08	31/08
Conclusions	13/08	31/08
Editing and formatting	27/08	06/09
Submission	07/09	07/09
Poster	**16/07/18**	**10/08/18**
Introduction	16/07	09/08
Results discussion	01/08	09/08
Editing and formatting	10/08	10/08
Submission	10/08	10/08

Chapter 4

Conclusion

APPENDIX

A

APPENDIX A

Begins an appendix

Bibliography

[1] 2018.

[2] 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.

[3] H. Bay, T. Tuytelaars, and L. Van Gool, *Surf: Speeded up robust features*, in European conference on computer vision, Springer, 2006, pp. 404–417.

[4] 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).

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

[6] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-programmable gate arrays*, vol. 180, Springer Science & Business Media, 2012.

[7] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*, vol. 56, springer, 2009.

[8] 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.

[9] 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.

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

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, *Vision meets robotics: The KITTI dataset*, International Journal of Robotics Research (IJRR), (2013).

[12] 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.

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

[14] 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.

[15] 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).

[16] 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).

[17] 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.

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

[19] 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.

[20] 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.

[21] MobilEye, *Enabling autonomous - mobileeye*, 2018.

[22] A. Newell and S. K. Card, *The prospects for psychological science in human-computer interaction*, Human-computer interaction, 1 (1985), pp. 209–242.

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

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 1 (2017), p. 4.

[25] 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.

[26] 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.

[27] M. Simon, S. Milz, K. Amende, and H.-M. Gross, *Complex-YOLO: Real-time 3D Object Detection on Point Clouds*, ArXiv e-prints, (2018).

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

[29] 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.

[30] Waymo, *Waymo safety report*, 2018.

[31] M. D. Zeiler and R. Fergus, *Visualizing and understanding convolutional networks*, in European conference on computer vision, Springer, 2014, pp. 818–833.

[32] Y. Zhou and O. Tuzel, *Voxelnet: End-to-end learning for point cloud based 3d object detection*, arXiv preprint arXiv:1711.06396, (2017).