Object detection and sensor data processing for off-road autonomous vehicles

Timothy Foster
timothyef@gmail.com

Object detection and sensor data processing

for off-road autonomous vehicles

By

Timothy Foster

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

April 2021

Name: Timothy Foster

Date of Degree: April 30, 2021

Institution: Mississippi State University

Major Field: Computer Engineering

Major Professor: John E. Ball

Title of Study: Object detection and sensor data processing for off-road autonomous vehicles

Pages of Study: 42

Candidate for Degree of Master of Science

Autonomous vehicles require intelligent systems to perceive and navigate unstructured environments. The scope of this project is to improve and develop algorithms and methods to support autonomy in the off-road problem space. This work explores computer vision architectures to support real-time object detection. Furthermore, this project explores multimodal deep fusion and sensor processing for off-road object detection. The networks are compared to and based off of the SqueezeSeg architecture. The MAVS simulator was utilized for data collection and semantic ground truth. The results indicate improvements from the SqueezeSeg performance metrics.

Key words: object detection, SqueezeSeg, lidar, camera, sensor fusion, computer vision, deep learning, machine learning

DEDICATION

I dedicate this project to small towns, backroads, six-strings, and muscle cars.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my supervisor and major professor, Dr. John Ball, for his unwavering support and guidance throughout all stages of this project.

The support and funding of this project by the Army Engineering Research and Development Center in Vicksburg, MS is truly appreciated.

I wish to acknowledge my family for their support throughout this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The goal of this thesis is to demonstrate compute-conscious data processing methods and deep learning techniques for off-road autonomous object detection. Object detection is an integral set of methods needed for building an environment model to be used by the autonomous system for path planning. The off-road environment is unstructured and requires a more robust object detection model due to the lack of key indicators present in traditional road autonomy.

1.1 Autonomous Vehicles

Autonomous vehicles are intelligent vehicular systems designed to safely transport or navigate a space without human control. Vehicular autonomy research has gain traction with the automobile industry pursuing full autonomy for passenger vehicles. The attention to autonomous systems has also driven innovations in sensor technology. The cost of enabling sensors such as lidar are decreasing in stride with the pursuit of autonomy. Researchers continue to perfect these autonomous systems for the societal impact on transportation safety and efficiency.

Autonomy in passenger vehicles will reduce accidents caused by human error. Over 90 percent of all car accidents are caused by human error and dangerous decisions [1]. Humans crash vehicles due to a sleuth of errors such as: recognition errors, decision errors, and performance errors. Autonomous systems are not prone to negligent driving behaviors, are less susceptible to performance and decision errors, thus reducing vehicle accidents and saving lives.

Though there have been great strides in the field, there are levels to autonomy. The autonomy capabilities of level zero through two are considered driver assistance features. These features provide the operator various levels of warnings and support such as lane centering, adaptive cruise control, and braking. Driver assistance requires the operator to actively drive or monitor the system. Level three bridges the gap between assistance and full system autonomy. However, the user must be ready to resume control when prompted. Though there aren't any fully autonomous level five passenger vehicles in production currently, the technology for level four autonomy is underway whereby the system is fully autonomous under certain conditions.

1.1.1 Autonomous Vehicle Framework

The Autonomous vehicle framework is an overview of the modules required for an autonomous vehicle to operate. The Eliot framework is a widely accepted framework describing the design and essential pieces required for full level five autonomy [5].

The sensor capture block, D-01, involves collecting data from various sensors such as a camera, lidar, radar, IMU, and GPS. These sensors are used to capture the state of the autonomous vehicle as well as the state of the environment surrounding it. Depending on the sensor, the data capture device may performs some transformations or the data may be preprocessed and transformed by the sensor itself.

The sensor fusion block, D-02, is in place to coherently combine the data from various sensors. Some sensors are better suited for certain conditions than others. In scenarios of sensor drop out and conditional limitations, the sensor fusion block consolidates the strengths of the various sensor data to mitigate any deficiencies.

The virtual world model block, D-03, keeps the state of the world model up to date. The world model contains context about the environment, relative position of other vehicles, as well as possible tendencies and scenarios that could happen in the current situation.

The system action plan block, D-04, determines how the vehicle should react given the state of the virtual world model. The actions in the plan must be feasible. The actions can't defy the laws of physics or require maneuvers outside the limitations of the physical system.

The controls activation block, D-05, is responsible for activating the controls in order to perform the actions specified by the system action plan block. D-05 utilizes the Automobile and CAN block, D-06, to control the underlying components of the vehicle. There is a feedback loop between blocks to insure the specified commands are actually being carried out as planned.

AI is heavily integrated into the framework. Deep Learning is integral throughout blocks D-01 to D-05. There are AI modules tasked with facilitating these functional blocks. Tactical AI, D-10, keeps up with the moment to moment driving task and all the particulars while Strategic AI, D-11, is concerned with completing the overall mission. Self Aware AI, D-12, watches over itself making sure the AI ecosystem is performing correctly and isn't faulty or confused.

This study is based in the sensor fusion block D-02 and the virtual world model D03. This project explores sensor fusion techniques and object detection that can be used to contribute to the world model.

1.2 Contributions

This project focuses on object detection for off-road autonomy. The main contributions of this thesis are methods to reduce computation and memory requirements for sensor fusion and deep learning algorithms.

- A sensor fusion pre-processing method that leverages a multi-layer convolutional autoencoder for pixel block pre-processing was developed and implemented. Co-registered lidar and camera sensors were leveraged for this new fusion method.
- Furthermore, SqueezeSeg was expanded to accept the fused data.
- A new SqueezeSeg based network was developed.
- SqueezeSeg was modified to increase accuracy without sacrificing real time inference speeds.
- Simulator code for the Mississippi State University Autonomous Vehicle Simulator (MAVS) was developed to generate scenes.

1.3 Publications

[6] - T. Foster, A. Dahal, J. E. Ball, “RGB Pixel-Block Point-cloud Fusion for Object Detection” *Proceedings of SPIE DCS*, 2021 [Accepted]

CHAPTER II

BACKGROUND

To perform compute-conscious object detection or object detection in general, there must be data collected from the environment as well as a method to process the data for detection. Autonomous vehicles typically use different sensors depending on the level of autonomy required. Multiple sensors are used for robustness. The sensor data goes through some preprocessing method and piped to a machine learning model for object detection.

2.1 Sensors

In this section, data collection is discussed. This study focuses on camera, lidar, and sensor fusion. These sensors have different characteristics that are suited for different conditions.

2.1.1 Camera

Cameras are passive sensors that perceive the world much like humans do. These sensors capture reflected light and provide high resolution data to the autonomous system. Stereo camera processing can also provide depth information along with the high resolution color, texture, and shape information. Because cameras are passive sensors that see the world much like our eyes, they can handle a myriad of weather conditions with limitations much like humans. Furthermore, cameras are inherently limited in the data they can provide for modeling the world. Specifically, they are hampered in low light situations and can't operate if there is no light such as at night. Intense direct light as well as certain situations with shadows could disable or confuse a system solely using cameras as the input to generate the world model. Cameras are widely used since high-quality and inexpensive units are readily available and because there are numerous high-performance networks to process camera imagery.

2.1.2 Lidar

Lidars, light detection and ranging, are active sensors that collect data by measuring the returns from pulses emitted from a laser. Lidars can provide data about the distance between the vehicle and surrounding objects based on the time it takes the emitted laser beam to return. It's also able to provide the intensity of the returns which can be used to further assist object identification. Because lidars are active sensors, they are not affected by lighting conditions. However, they are fairly robust to weather but some conditions can introduce inconsistencies that must be accounted for. Fog, dust and rain can cause some degradation or attenuation in the lidar returns.

2.1.3 Fusion

Sensor fusion increases the fidelity and robustness of the data used to create the world model. With sensor fusion, the limitations of one sensor are accounted for by other sensors. For example, the lidar isn't able to perceive some visual information like the text on road signs whereas the camera has no problem. On the other hand, a camera, unlike lidar, can't range or see objects in the dark.

Cameras provide images, which are stored as arrays of red, green, and blue values in computer memory. Lidars provide very different data, point clouds, which makes processing much more challenging. Having co-registered lidar and cameras allows direct data-level fusion by projecting the lidar points into the camera images or vice versa. Lidars provide high-accuracy 3D information, but most lidars are monochromatic and most scanning lidars have a limited amount of azimuthal resolution, since they only have 64, 128 or 256 beams (typically). Cameras, on the other hand, provide rich texture and color information, but camera processing does not provide nearly as accurate and precise 3D information. By fusing camera and lidar data, strengths of both platforms can be exploited.

2.2 Artificial Neural Networks

In this section, neural networks are discussed. Neural networks are an integral part for full autonomy. This project uses convolutional autoencoders for data preprocessing and convolutional neural networks for object detection.

2.2.1 Autoencoders

Autoencoders are unsupervised networks that are able to consume input data, learn a latent representation of the input, then reconstruct the input from the new compressed representation. These networks are considered unsupervised because they don't require labels. They can be trained with raw data. Autoencoders consist of an encoder and a decoder section. The encoder is an artificial neural network that compresses the dimentionality of the input into a code. The decoder is an artificial neural network designed to reconstruct the original input from the compressed code representation. The autoencoder has a reconstruction loss function to fine tune the weights through backpropogation. Autoencoders have many applications such as denoising, compression, feature extraction, and data generation. In this study, autoencoders are used for dimentionality reduction.

2.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are supervised networks use for feature extraction and classification. These networks use a series of convolution layers along with pooling layers to learn filters for feature extraction. These filters allow the network to extract learnable features from the labeled data in the image. To extract these features, a sliding window of convolution operations are performed on the entire image. Through training and backpropogation, the network learns what filters are necessary to extract relevant features. After the filters extract the features, the data typically propagates through a fully connect layer and a softmax layer for classification.

2.2.3 SqueezeSeg

In this thesis, a neural network named SqueezeSeg was modified and utilized for comparative analysis. SqueezeSeg is a 3D semantic segmentation network built on a CNN model. The network takes a lidar point cloud transformed into a 2D grid representation and outputs a semantic label map (that is, each pixel in the map has a class label). Because CNN models operate using images, the data is represented as 3D tensors of size $H \times W \times C$. Height (H) and width (W) represent the number of beams or vertical channels the lidar has and the number of vertical columns in the grid or field of view respectively, and C is the number of channels. SqueezeSeg reduces network parameters by optimizing the convolution operation. SqueezeSeg's FireModule are used instead of traditional convolution layers. These modules have a squeeze and an expand convolutional layer consisting of 1×1 filters followed by parallel 1×1 and 3×3 filters that are finally concatenated. These modules require less computation than standard 3×3 convolutional filters. The SqueezeSeg architecture is shown in figure 2.1 and the fire convolution and deconvolution modules are shown in figure 2.2.

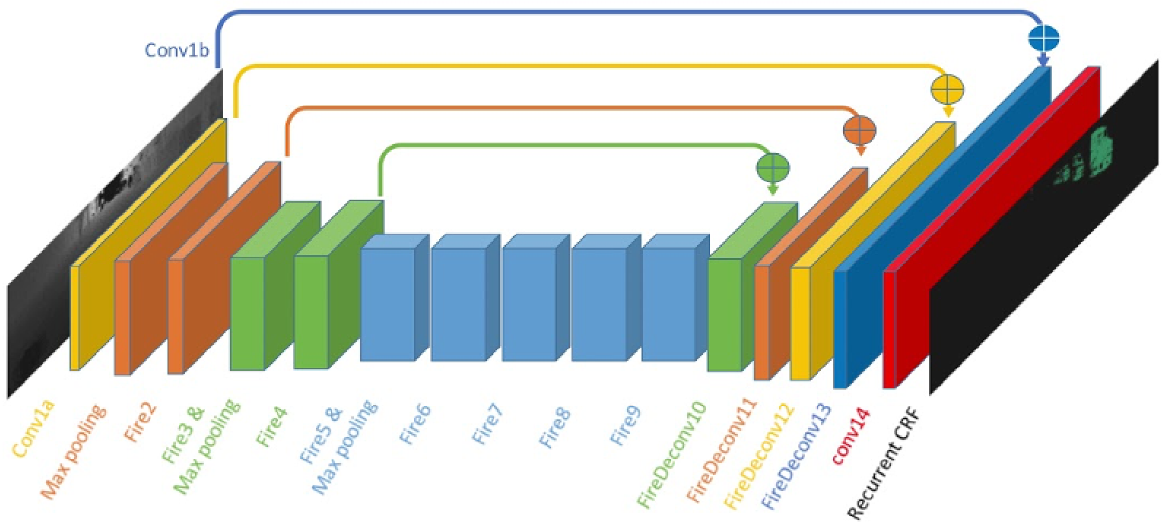

Figure 2.1

The SqueezeSeg Architecture. [18]

2.3 MAVS

The MAVS software library was developed by the Center of Advanced Vehicular Systems (CAVS) at Mississippi State University [8] [7]. MSU Autonomous Vehicle Simulator (MAVS) is a realistic physics-based simulator for autonomous ground vehicles and sensors. MAVS captures realistic digital terrains and evaluates the performance of autonomous perception and navigation software with various sensors such as lidar, GPS, cameras, and other sensors. A Message Passing

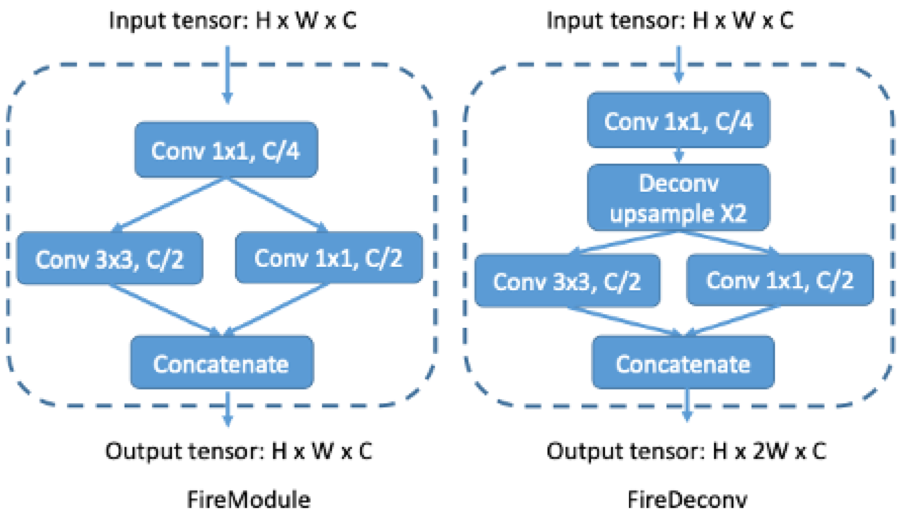

Figure 2.2

SqueezeSeg fire module architecture. [18]

Interface (MPI)-based framework was used to couple parallel processes such as ray tracing. MAVS is a fully functional standalone simulator that has additional wrappers that allow MAVS to be integrated with robotic development tools such a Robotic Operating System (ROS). MAVS is built on C++ and can be interfaced through the C++ API, MATLAB, and the Python API.

In this project, MAVS generated scenes and captured the lidar point-cloud and camera data. MAVS also generated the segmentation labels of each point in the point-cloud. The scene in 2.3 demonstrates MAVS photorealistic scene generation capability.

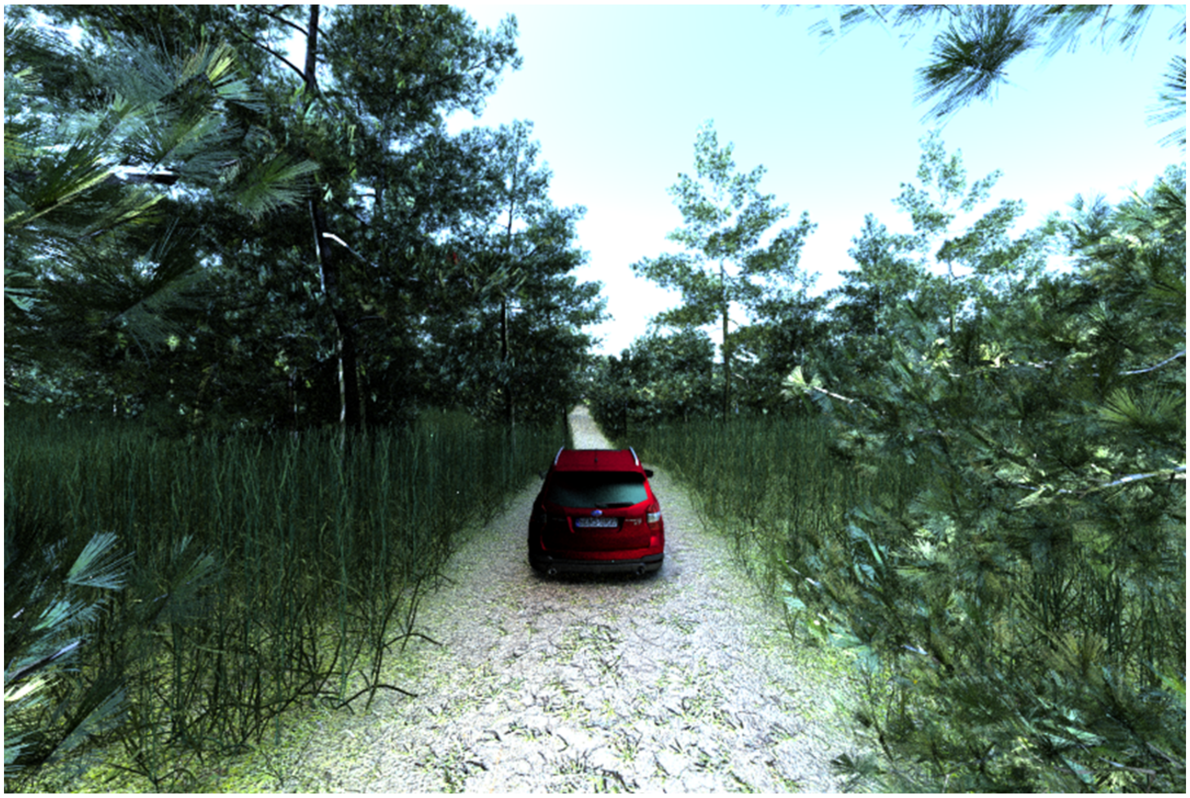

Figure 2.3

MAVS generated photorealistic scene generation.

CHAPTER III

AUTOENCODERS FOR SENSOR FUSION

This chapter focuses on an autoencoder leveraged preprocessing step for sensor fusion.

3.1 Introduction

Off-road autonomy requires highly accurate object detection for obstacle avoidance and navigation. Real-time accurate object detection is essential for off-road autonomous vehicle navigation. The autonomous system must be able to classify objects that reside in the off-road environment such as trees, vegetation, paths, and other obstacles. The off-road environment introduces a different set of challenges than traditional structured road autonomy. In the traditional structured road environment, there are lane markings and known indicators for navigation. However, the off-road problem space presents many issues for computer vision. The deep learner has to contend with dense vegetation, occlusion, and lack of structure in the environment, to name a few issues.

The state-of-the-art methods in object detection almost always utilize a combination of sensors in order to more accurately model the environment and the corresponding obstacles. Choosing the most effective combination of sensors is a key element to the solution. Furthermore, the method of combining the data from multiple sensors must be considered. The sensor data has to be effectively fused using a method that retains the valuable properties of each sensor. There are various deep learning architectures and techniques available for object detection. However, the off-road environment is unstructured with both clutter and sparsity. The type of sensors used to provide data to the deep learner is one of the most important pieces. The autonomous driving industry mostly relies on cameras, lidars, and or radar sensors to capture data about the environment. There are tradeoffs and compromises that come with choosing various sensors. Lidars provide distance information and are not affected by lighting conditions. They operate as intended night or day and are fairly resilient to weather conditions. Cameras provide color and luminance data but are disadvantaged in low light conditions. Radars provide information on range rate via Doppler processing and distance. In this paper, we will explore specific techniques for camera and lidar sensor fusion.

The deep learning network used to evaluate the proposed sensor fusion techniques is based on SqueezeSeg [18]. SqueezeSeg uses a convolutional neural network to perform semantic segmentation on road-objects. The layers within the CNN are designed to reduce the number of parameters in the network while retaining accuracy. The architecture includes convolutional modules comprised of a squeeze layer along with an expand layer to reduce parameters introduced by stacking multiple filter operations. The input layer of the module is called the squeeze layer. It is a 1×1 convolution operation that reduces channel size. The expand layer is comprised of a 3×3 and 1×1 convolutional layers in parallel. The output of those filters is concatenated regaining the original channel size. This module requires fewer parameters and computations than a standard 3×3 convolutional layer. With fewer parameters, memory requirements and complexity is reduced. In turn, the inference speed of the network increases to meet the real-time speeds required for road autonomy. Scanning lidar data is dense in azimuth (left-to-right) but typically sparse in elevation (up and down), due to a limited number of beams. In SqueezeSeg, the point cloud is transformed onto a spherical projection to create a dense, more regularly distributed, 2D grid representation.

The CNN accepts the transformed lidar point cloud and generates a label map per each lidar point. This process is called semantic segmentation. Any blurry boundaries in the label map are refined by a conditional random field that is implemented as a recurrent neural network module and trained along with the CNN. Each class in the label map is clustered to provide instance level labels. In this paper, SqueezeSeg is modified to accept a larger input vector to accommodate the additional image data along with the transformed point cloud data.

3.2 Background

In this section, we examine the state-of-the-art methods in 2D sensor detection, 3D sensor detection, and fusion.

3.2.1 2D Sensor Detection

There have been several image-based classifiers for object detection. SqueezeNet [10] retained accuracy of larger networks while reducing network parameters by introducing convolution modules and downsampling late in the network. These modules contained specific numbers of filters and input channels structured to reduce the number of parameters in the network. Faster R-CNN [14] uses pipeline processing including region proposals, classification, and refining detection boundaries. R-FCN [4], a region proposal network (RPN) similar to R-CNN, except it generates score maps before region of interest (ROI) pooling eliminating fully connected layers after ROI pooling. YOLO [13] bypasses pipeline processing and frames object detection as a regression problem to perform classifications in one step.

3.2.2 3D Sensor Detection

To take object detection performance to the next level there has been several classifiers designed to process lidar point cloud data. SqueezeSeg [18] is essentially a 3D segmentation with an architecture inspired by SqueezeNet [10] to retain accuracy while reducing network parameters to retain real-time inference. PV-RCNN [15] combines the voxel CNN for accurate proposals and PointNet-based set abstractions for accurate location and context information. SGPN [16] generates similarity matrices within the feature space to group points for instance segmentation.

3.2.3 2D/3D Sensor Fusion Detection

After the exploration of object detection with various sensors of various dimensionalities, research moved toward sensor fusion. The idea is to capitalize on the strengths of different sensors while minimizing the shortcomings faced when utilizing single sensors. RoIFusion [3] generates regions of interest. Then, the 2D and 3D data in those regions are fused and classified. This reduces computation because it does not have to fuse the entire point cloud with the entire image. In [2], a fully connected was used in tandem with cross fusion. In cross fusion, the features of the 2D and 3D data are fused in sections of the feature extraction pipeline. The optimum places to fuse in the network are learned by the deep learner.

Previous fusion methods provide classifiers with more data about objects of interest. There are also various methods of fusion to help the classifier extract features from the extra data. However, this paper addresses one of the main compromises associated with sensor fusion itself. Sensor fusion increases the amount of data that the deep learner has to process. Though this additional data can create a more robust model, it also increases memory and computational requirements. In an autonomous system, it is beneficial to have the most robust but also light weight model to fulfil processing speed constraints. These systems require real time detection. Our proposed method reduces memory requirements of sensor fusion while retaining enough data to yield a robust model.

3.3 Methodology

This section explores the methods performed for experimental analysis of the proposed method.

3.3.1 Proposed System

The proposed autonomous system is comprised of a multi-sensor platform mounted on an off-road autonomy capable vehicle. The sensor platform is equipped with a 64-beam lidar and a wide angle camera. The camera and lidar should be calibrated to map points from the lidar coordinate system to the camera coordinate system. This allows for proper sensor fusion. Furthermore, the sensor platform should also be equipped with a GNSS/INS unit. This will allow for localization and position tracking. The processing unit should be able to support the bandwidth required to operate all sensors in tandem.

3.3.2 Data

Data is the most important aspect of training a machine learning (ML) model. The proposed network is designed to output segmentation. The data for training the network was generated by MAVS, a high-fidelity vehicle automation simulator. Simulated point cloud and camera data was generated for various off-road scenes containing various classes for detection. The simulated scenes include various forest types, plains, valleys, country roads, etc. Every scene features a trail on which a simulated vehicle loaded with sensors traverses the defined terrain while collecting data.

There are six classes including the following: ground, trail, trees, vegetation, obstacles, and fences. The simulator labels every point in the point cloud as well as every pixel in image for segmentation. Also, the camera and lidar sensors are co-registered. This allows us to map every point of the point cloud to a block of RGB pixels. Figures 1a and 1b show examples of the camera data generated by MAVS. Figure 1a depicts a flatland scene with sparse trees and various obstacles. Figure 1b depicts a hilly environment with dense trees and sparse vegetation. Additionally, figure 3.1 depicts the class segmentation that MAVs generates for each data-frame captured as the sensors travel along the path. figure 3.2 on the next page features semantic segmentation of a desert scene, and figure 3.2 on the following page is the segmentation of the scene shown in in figure 3.1a.

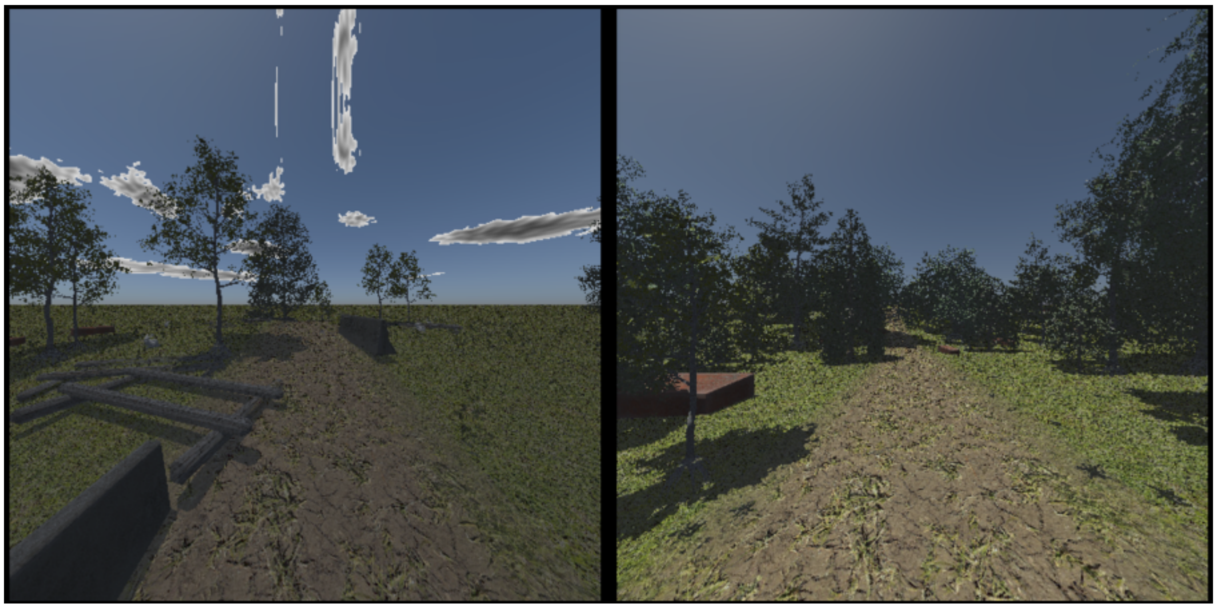

Figure 3.1

(a) Left - Scene generated by MAVS. (b) Right - Scene generated by MAVS.

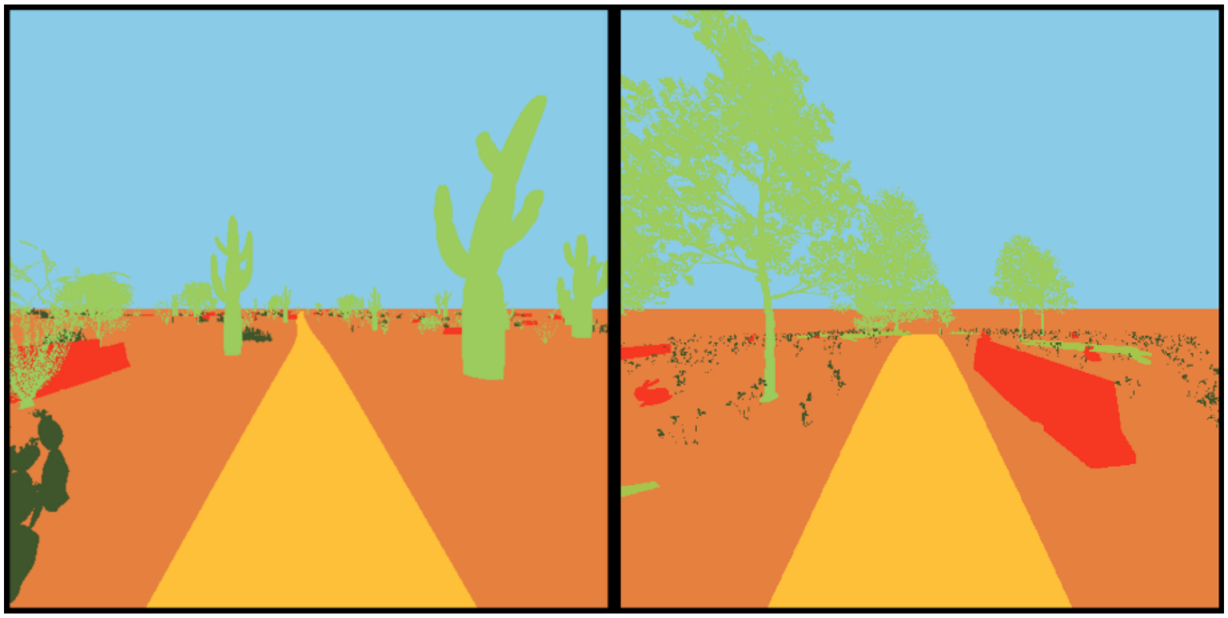

Figure 3.2

Segmented scenes generated by MAVS

3.3.3 Point Cloud to Image Mapping

Lidars typically have a wider field of view than standard cameras. To utilize more of the lidars field of view, it's helpful to use a wide-angle camera. To project the lidar points onto the image plane, we use the following formula to map the lidar points to a pixel coordinate in the image. To correlate a lidar point's x, y and z coordinates to a pixel's x and y coordinate, we use the following equations.

$$nxpix = \text{number of pixels in the } x \text{ dimension} \tag{3.1}$$

$$nypix = \text{number of pixels in the } y \text{ dimension} \tag{3.2}$$

$$xpixdim = \frac{\text{pixel plane horizontal dimension}}{nxpix} \tag{3.3}$$

$$ypixdim = \frac{\text{pixel plane vertical dimension}}{nypix} \tag{3.4}$$

$$u = -\text{focal length } \times tan^{-1}\left(\frac{y}{x}\right) \tag{3.5}$$

$$v = -\text{focal length } \times tan^{-1}\left(\frac{z}{x}\right) \tag{3.6}$$

$$xp = \text{floor}\left(\frac{nxpix}{2} + \frac{u}{xpixdim}\right) \tag{3.7}$$

$$yp = nypix - \text{floor}\left(\frac{nypix}{2} + \frac{v}{ypixdim}\right) - 1 \tag{3.8}$$

In a system using a co-registered camera and lidar, the intrinsic and extrinsic matrices for the camera and lidar along with the co-registration data would be used to convert the lidar coordinates into camera coordinates. After the pixel coordinates (xp, yp) are calculated, they are checked to ensure that they are within the bounds of the image. If they are within the boundaries of the image, the RGB value of the pixel can be found using the lidar's point coordinates and the transform that maps to the pixel's location. Otherwise, the point cloud is outside of the image's field of view and can't be used for fusion. Figure 3.3 shows the lidar points projected onto the co-registered image. Figure 3.4 shows the lidar points projected onto a wide angle camera image. Lidar sensors are capable of collecting data with a 360 degree field of view. To take advantage of the lidar's field of view with sensor fusion, a wide angle camera is used to maximize the number of points that can be projected onto the image. However, increasing the camera field of view also increases distortion in the image. Therefore, there are limits to how wide the camera angle can be set.

3.3.4 Architecture

The network's architecture is composed of two main components. The first component is an autoencoder employed as a preprocessing stage for the RGB pixel blocks. The features from this stage will be fused with the point cloud data and processed by the second component. The second component is a network architecture we call Pixel-Block Point-cloud Fusion Network or PBPC Fusion. PBPC Fusion is a SqueezeSeg based network that will output the object detection segmentation. This tandem of stages allows the deep learner to extract the most useful features from the pixel blocks retaining context before fusing with the point cloud data. In the first stage, 5×5 blocks of pixels centered around each lidar point are extracted. These pixel blocks are then encoded by the trained autoencoder. Then the second stage is responsible for further feature extraction from the fused data. By using a SqueezeSeg based network with relatively few parameters, the time required for processing the first stage can be afforded to maintain real time detection. Figure 3.5 shows the end to end data flow.

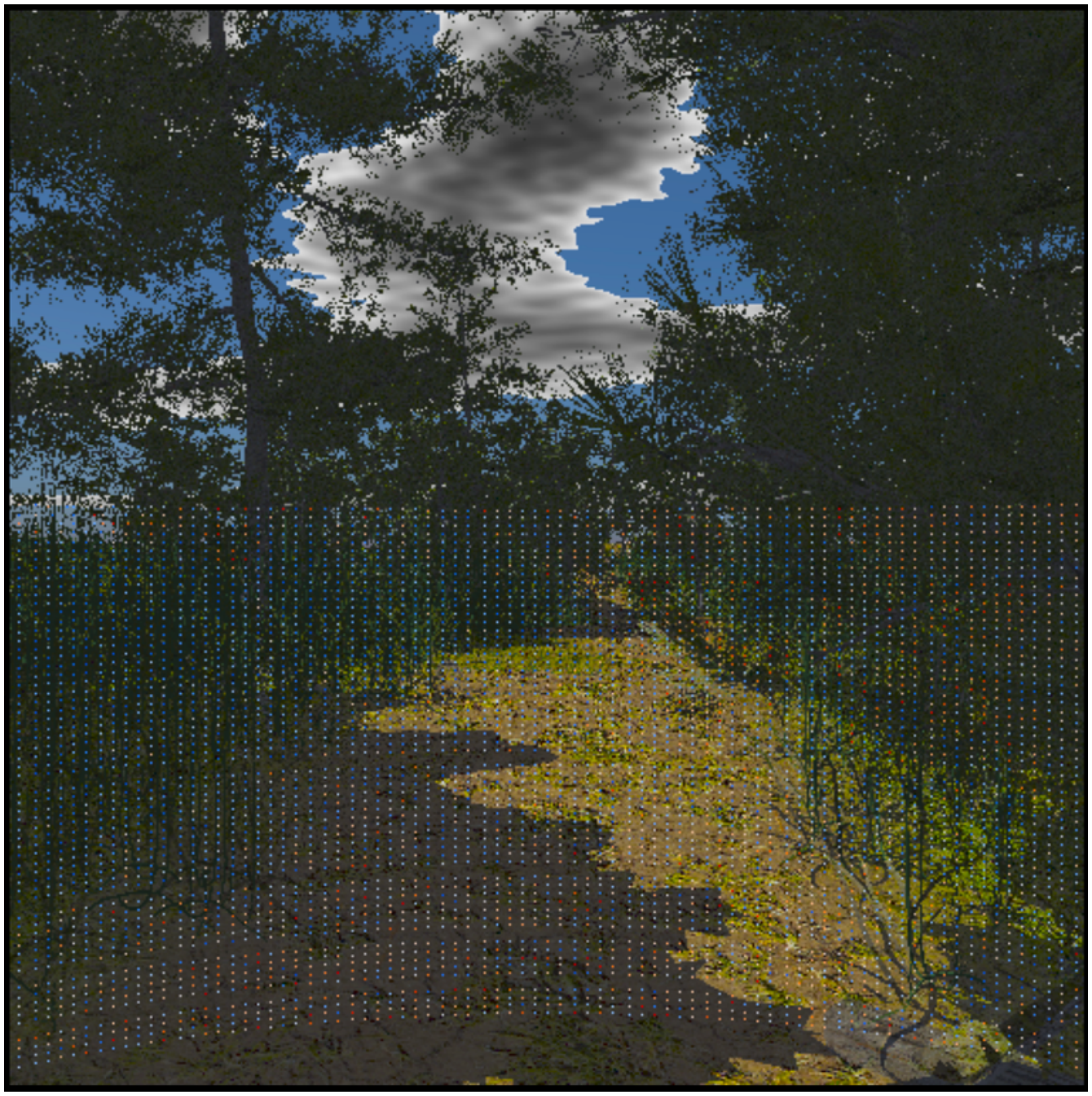

Figure 3.3

Point cloud and image generated by MAVS. The lidar data is superimposed onto the camera imagery. Best viewed in color.

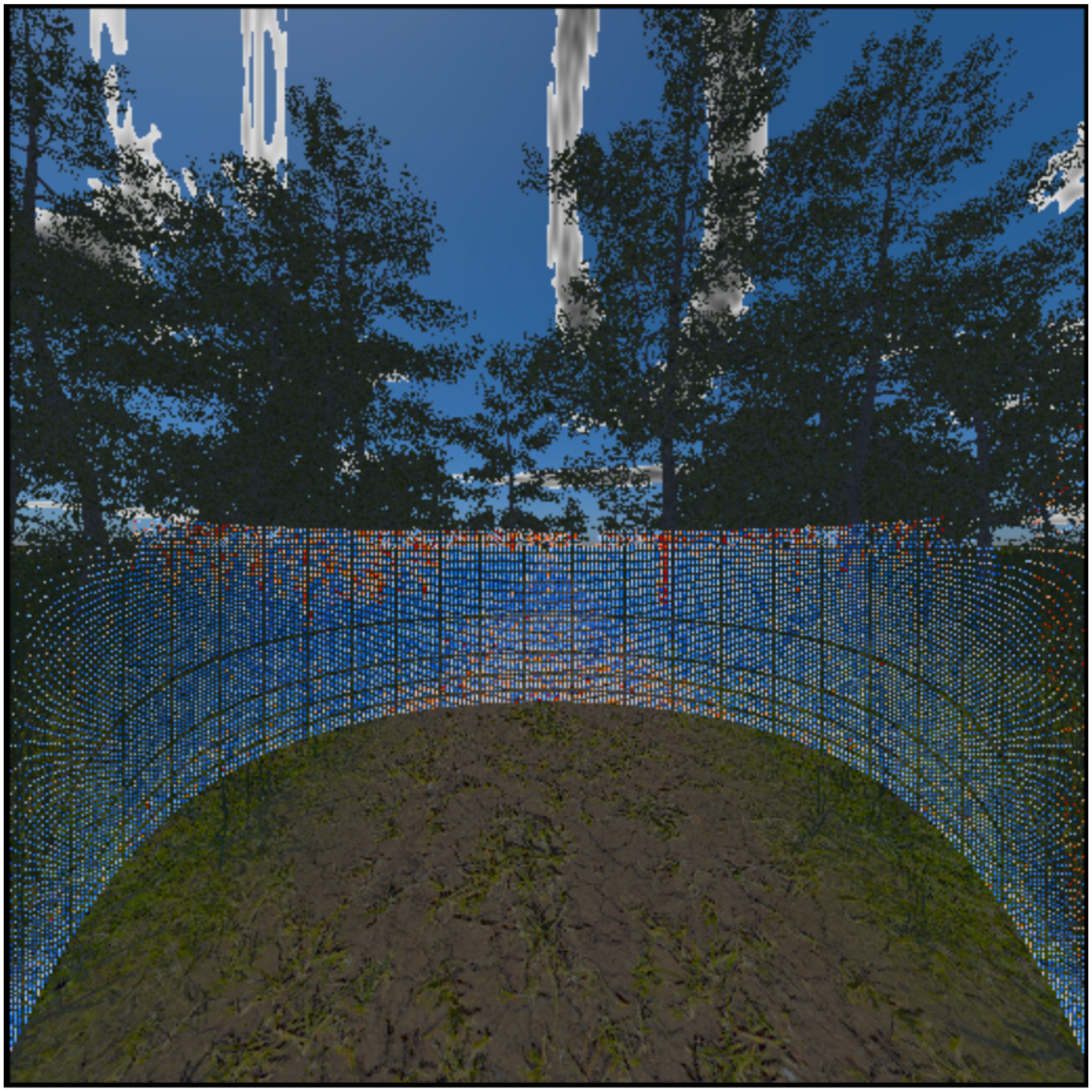

Figure 3.4

Point cloud and image generated by MAVS. Wide angle camera parameters.

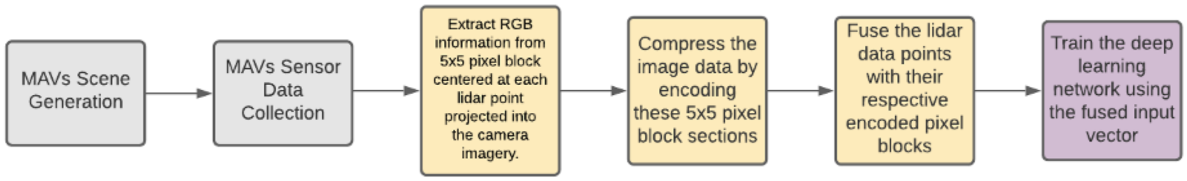
MAVs Scene Generation
MAVs Sensor Data Collection
Extract RGB information from 5x5 pixel block centered at each lidar point projected into the camera imagery.
Compress the image data by encoding these 5x5 pixel block sections
Fuse the lidar data points with their respective encoded pixel blocks
Train the deep learning network using the fused input vector

Figure 3.5

The processing procedure from data collection to training the network.

3.3.5 Autoencoder

A multilayer autoencoder is trained to represent inputs, the pixel blocks, in a dimensionally reduced encoding. The encoder side of the autoencoder learns to process higher dimensional data and represent it as a code in a lower dimensionality. The decoder side learns to take the code and reconstruct the original inputs. We propose to use the code from a trained autoencoder instead of the actual pixel blocks. Large blocks of input data tend to increase the training and inference time of vision networks. By encoding part of the input to a lower dimension, the input size is reduced by more than half. A RGB encoding of a 5×5 pixel block requires 75 numbers. The proposed autoencoder represents the same 5×5 block of pixels as a code of 25 numbers, 33 percent smaller. In essence, the encoder dropped the equivalent of two channels from the RGB data. The input of our PBPC Fusion network will be these pixel-block codes along with the point cloud data. Figure 3.6 show the architecture of the proposed Autoencoder. The encoder contains two 2D convolutional layers with 72 filters and one filter, respectively, for dimensionality reduction. The decoder also has two 2D convolutional layers. The first layer has 72 filters while the second layer has three filters to reconstruct the original input. The 2×2 kernel was used throughout to extract features from the relatively small 5×5 pixel blocks. The encoder loses some color information but retains texture information exceptionally well.

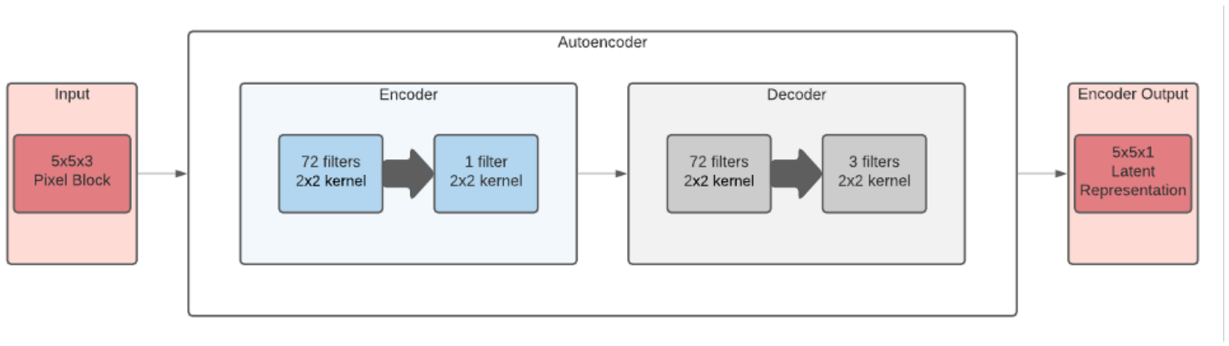

Figure 3.6

The proposed autoencoder architecture to process 5×5 pixel blocks.

3.3.6 Pixel Point Fusion Network

The proposed methods can be adapted to work for any deep learning network. In this paper, SqueezeSeg was expanded to process additional inputs. Originally, the network input only accepted 3-D coordinates, lidar intensity, and depth. The input vector only contains 6 numbers. With our network, we experimented with 3 different fusion methods for the feature vector.

The first fusion method termed Pixel Block Fusion (PBC) was designed to fuse enough RGB data to represent color and texture centered around each lidar point within the image frame. The input vector included the following: 5×5 pixel block, point-cloud coordinates, intensity, depth, and label. This input vector contains 81 numbers. The size of the pixel block was chosen based on memory and computation constraints. Because each pixel is represented as a RGB tuple, increasing the size of the pixel block exponentially increases the size of the input vector.

This leads us to our proposed method termed Autoencoded Pixel Block Fusion. The input vector included the following: an encoded 5×5 pixel block, point-cloud coordinates, intensity, depth, and label. An autoencoder is used to generate a latent representation of the pixel blocks. This input vector contains 31 numbers and is 38 percent smaller than the PBC method while representing the same data. Using this lower dimensional representation of the pixel blocks allows us to increase the size of the pixel block without exponentially increasing the size of the input vector.

Finally, we implemented a fusion method utilizing the minimal amount of RGB data. Here we only fuse one RGB pixel per lidar point instead of a block of pixels centered around a point. The method is termed Singe Pixel Fusion. It's the least computationally and memory expensive method apart from the baseline, SqueezeSeg. The input vector included the following: one pixel, point-cloud coordinates, intensity, depth, and label. This input vector contains 31 numbers and is 38 percent smaller than the PBC method while representing the same data.

3.4 Results

The proposed Autoencoded Pixel Block Fusion method categorically outperforms all the other methods in accuracy while maintaining real time detection speeds within 30-40 fps. The dimensionality reduction performed on the 5×5 pixel blocks kept the memory and computational requirements manageable while still maintaining a robust data representation for improved detection. The Singe Pixel Fusion method categorically outperformed standard SqueezeSeg without

Table 3.1

Table to compare the IOU accuracy of different fusion methods to SqueezeSeg. The networks were trained for 50,000 steps.

Classes	SqueezeSeg	Single Pixel Fusion	Pixel-Block Fusion	Autoencoded Pixel-Block
ground	75.2	77.3	79.2	83.2
trees	79.6	80.6	80.6	84.3
vegetation	70.3	71.1	69.4	75.5
trail	37.4	64.4	86.0	89.8
obstacles	52.4	60.5	45.8	68.0
fence	0	0	0	0

Table 3.2

Table to compare the inference speed of various fusion methods to SqueezeSeg

Method	Detection Speed
Baseline SqueezeSeg	13 ms
Single Pixel Fusion	13 ms
Autoencoded Pixel-Block Fusion	24 ms
Pixel-Block Fusion	43 ms

decreasing detection speed. The full Pixel Block Fusion (PBF) method outperformed SqueezeSeg overall, but two classes decreased in accuracy. The PBF method's feature vector is large containing

81 numbers. This large input vector requires more training and increases detection time beyond the real time processing constraints. Accuracy and detection speeds are shown in tables 3.1 and 3.2 respectively.

The networks were trained on an Alienware running ubuntu with 31.3 GB of memory, an 8th gen Intel i7 2.2GHz x12, and a GeForce GTX 1080 GPU with 8 GB of memory.

The deep learning networks were developed using tensorflow 1.14. Each network was trained on 50,000 steps with a batch size of 8. The learning rates were 0.003 with a momentum of 0.09 and a decay factor of 0.5. Data augmentation and random flipping was also applied during training.

3.5 Conclusions

In this paper, we propose a novel deep fusion method, Pixel Point Fusion, to fuse point-cloud data with pixel blocks for 3D object detection. We have co-registered camera and lidar data, a CNN autoencoder to generate features from the 2D pixel blocks, and a SqueezeSeg based network architecture to process the pixel features along with the co-registered point cloud. The 2D features are fused with the point cloud data at the input level to predict the 3D object segmentation. Our fusion method can be adapted to other network architectures. This model architecture along with the proposed fusion method shows state of the art performance for 3D object detection.

CHAPTER IV

PROCESSOR EFFICIENT COMPUTER VISION ARCHITECTURE

This chapter focuses on redesigning the SqueezeSeg architecuture for more accuracy while maintaining its processing efficient convolution modules.

4.1 Introduction

There are numerous companies and researchers working to develop fully autonomous vehicles. However, most of the work being done is geared toward autonomy in the relatively structured, traditional roadway environments. Off-road autonomy presents it own set of challenges due to the unstructured nature of the environment. The off-road environment contains objects of interest that range from relatively small to relatively large. The environment can present dense vegetation and various levels of occlusion. To successfully navigate, the system needs to be able to detect objects of interest and identify drivable areas. Well-designed deep learning networks are capable of reliable object detection in these environments. However, the system has to process data in real time for obstacle avoidance.

In this chapter, the focus is on modifying SqueezeSeg to increase performance while not sacrificing data processing speed. SqueezeSeg [18] performs semantic segmentation on lidar data. The SqueezeSeg network has been considered state of the art for autonomous driving. However, in the off-road environment, the network doesn't perform as well. The network is designed to reduce the number of parameters that convolutional neural networks incur when multiple convolutional layers are stacked. Typically, networks with more parameters are inherently slower due to the extra computations and memory requirements. SqueezeSeg reduces the rate that network parameters increase per convolution layer by implementing special convolution modules. The input data is processed through a 1×1 convolution operation to squeeze the channel size, then propagated through 3×3 and 1×1 convolutional layers in parallel to regain the original channel size. This progression of operations requires fewer parameters and computations than a standard 3×3 convolutional layer. However, the squeeze reduces the receptive field of the network.

There are various methods available to increase the receptive field of a network such as: adding convolution layers, subsampling, and utilizing dilated convolution layers. Making a network deeper by adding convolution layers will linearly increase the receptive field while exponentially increasing the number of parameters. Subsampling convolution layers increase the size of the receptive field multiplicatively based on the stride used. Subsampling or striding convolution is comparative to pooling operations. These operations increase receptive field at the cost of resolution [11]. Therefore, this paper focuses on dilated convolution to expand the receptive field. In dilated convolution, the kernel values are spaced out by a dilation factor which increases the receptive field without sacrificing resolution. The images in figure 4.1 shows the receptive field and resolution of these convolution operations. With dilated convolution, the receptive field increases exponentially while the number of parameters increases linearly. For example, a 3×3 filter has nine parameters with a receptive field of 3×3. A 3×3 filter with a dilation factor of three has nine parameters as well. However, the dilation increased the receptive field from 3×3 to 11×11. The images in figure 4.2 show various dilation factors and receptive fields.

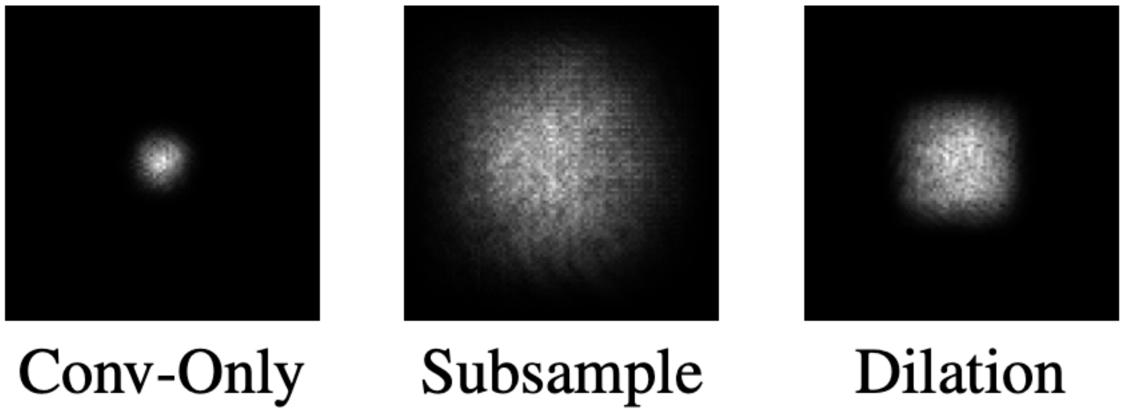

Figure 4.1

Shows receptive field and resolution of different convolution operations. [12]

4.2 Background

To take object detection performance to the next level there have been several classifiers designed to extract a range of features from small to large scale without losing valuable context. SqueezeSeg V2 [19] is derived from SqueezeSeg to include context aggregation modules (CAM). The CAM are placed at the beginning of the network to reduce dropout noise and increase the receptive field by using a large max pooling kernel followed by 1×1 convolution operations. PointSeg [17] proposed an enlargement layer for their SqueezeNet [10] based network. The enlargement layer is placed at the end of the network to increase the reception field. It is composed of multiple dilated convolution filters of varying rates to get multi-scale features simultaneously. These dilated filters are concatenated with a global average and 1×1 convolution then feed to the input of another 1×1 convolution layer. [9] proposed a local feature extraction module composed of dilated convolution filters with decreasing dilation factors for small feature extraction.

Previous methods outlined here utilize dilated convolution as well as other methods to increase the receptive field of the network. However, the proposed method builds upon the SqueezeSeg [18] architecture to take advantage of the squeeze and expand modules designed to reduce the computational cost of performing convolution operations. Some previous approaches have explored increasing the receptive field of SqueezeSeg at the beginning or end of the network, but the approach outlined in the following section is designed to increase the receptive field throughout the entire network for large scale context aggregation.

4.3 Methodology

The proposed network's architecture is based on SqueezeSeg. The proposed modifications are focused on SqueezeSeg's fire modules. The original SqueezeSeg architecture is shown in figure 2.1. The fire modules are composed of a squeeze layer for cross channel dimensionality reduction and an expand layer for feature extraction. The fire modules are shown in figure 2.2. The method proposed assigns a dilation rate to each fire module. These are termed dilated fire modules. The dilation rates are applied to the convolution filters in the fire module. Fire module 2 and 3 were given a dilation factor of 3. Fire module 4 and 5 were given a dilation factor of six. Fire module 6 and 7 were given a dilation factor of nine. Fire module 8 and 9 were given a dilation factor of 12. The FireDeconv modules in the deconvolution section of the network were matched with the fire modules in the convolution section. FireDeconv10 was given a dilation factor of 12. FireDeconv11 was given a dilation factor of 6. Fire Deconv12 was given a dilation factor of three.

The new architecture increases the receptive field throughout the network for large scale context aggregation. Typically, the early layers of the CNN extract lower level features. To retain the ability

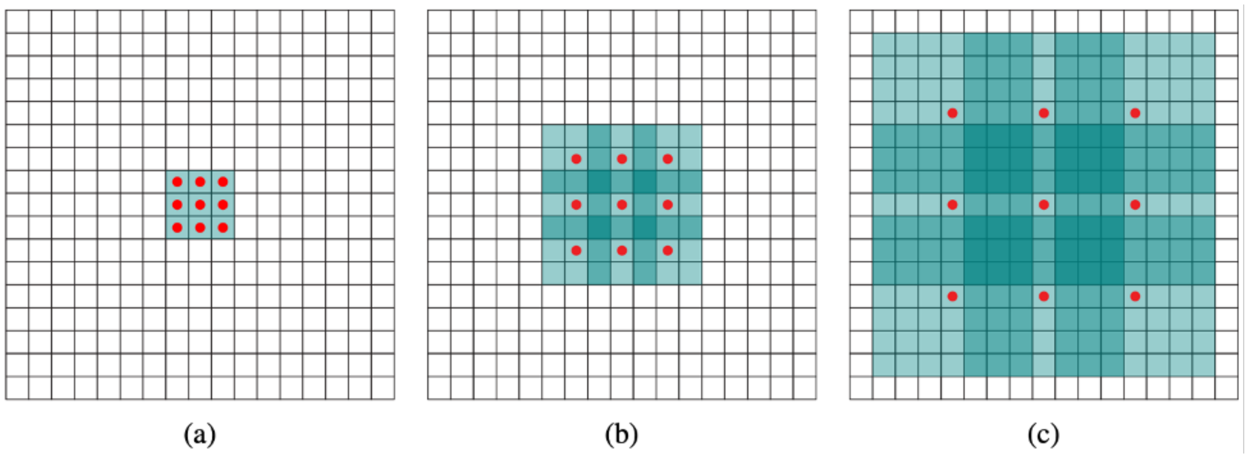

Figure 4.2

(a) 1-dilated convolution; has a receptive field of 3×3. (b) 2-dilated convolution; has a receptive field of 7×7. (c) 4-dilated convolution; has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly. [20]

to extract these low-level features, the early dilated fire modules have lower dilation rates. As the layers progress, the network begins to extract higher level features, hence the gradual increase in dilation rates. The higher dilation rates toward the end of the network have a larger receptive field to better extract high level features. The progression of increasing dilation rates builds this network up for large scale context aggregation whereby it can extract low-level features without losing bigger picture context. There is a balance to fine tune between receptive field size and fine feature extraction. Using dilation rates too high will overlook fine features, and dilation rates too low will leave context information untapped. The context information is necessary to maximize accuracy.

4.3.1 Dilated SqueezeSeg with Autoencoded Pixel-Block Sensor Fusion

In this section, the most successful methods discussed in this project are combined. In chapter three, sensor fusion methods were discussed. The autoencoded pixel-block fusion method proved to be the most accurate sensor fusion method. This method combined lidar point coordinates with the encoded dimensionally reduce blocks of pixels that surround each lidar point. This encoding reduced the amount of data to be processed while retaining color and texture information correlated to each lidar point. To conclude this chapter, the Dilated SqueezeSeg architecuture will be modified to accept the Autoencoded Pixel-Block fusion data. This will be termed Dilated SqueezeSeg w/ Autoencoded Fusion.

4.4 Data

The network was trained using lidar point cloud data. The data was generated using the MAVS simulator. This simulator generated off-road scenes with various classes and labels the data. The classes included ground, trees, vegetation, trail, obstacle and fence. Each data point consisted of the xyz point cloud coordinates, intensity, distance and a label. The image in figure 4.3 shows a scene and corresponding segmentation generated by MAVS.

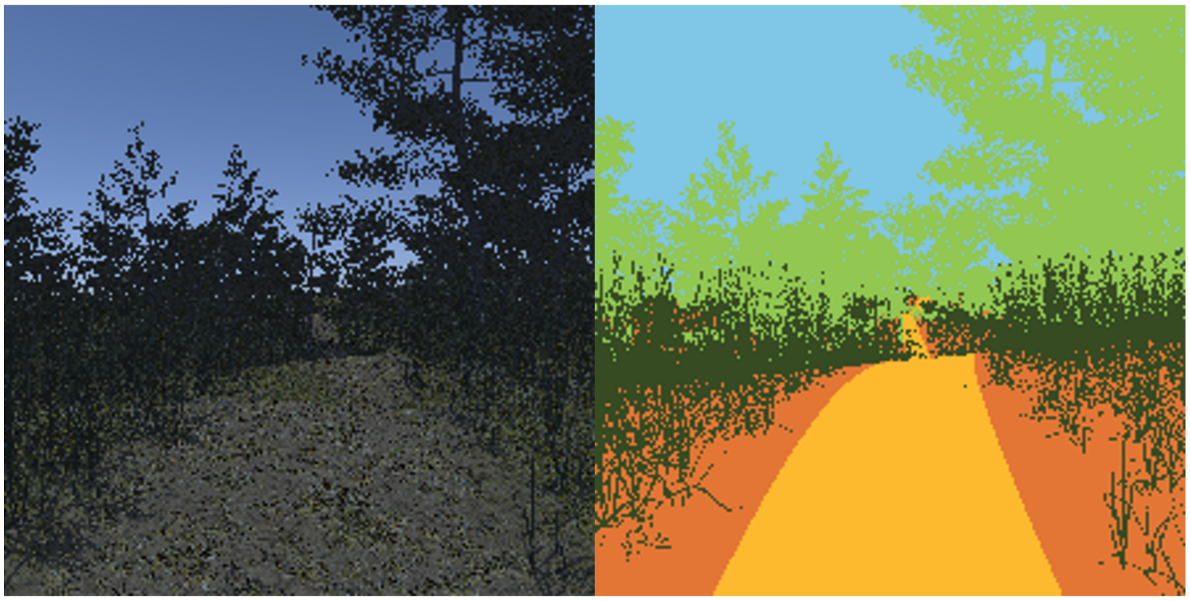

Figure 4.3

MAVS scene generation and segmentation.

4.5 Results

The proposed Dilated SqueezeSeg categorically outperforms SqueezeSeg while maintaining real time detection speeds within 30-40 fps. On average, Dilated SqueezeSeg is six percent more accurate than the original. As shown in table 4.1 and table 4.2, the main accuracy gains were realized in the trail and obstacle classes. The accuracy increased by 13 and 5 percent respectively.

Table 4.1

Table to compare the IoU accuracy of SqueezeSeg, Dilated SqueezeSeg, Dilated SqueezeSeg with Autoencoded Fusion, and SqueezeSeg with Autoencoded Fusion respectively. These networks were trained for 150,000 steps.

Classes	SqueezeSeg (SS)	Dilated SqueezeSeg (DS)	DS w/ Auto Fusion	SS w/ Auto Fusion
ground	72.8	75.9	87.0	87.0
trees	78.6	81.2	86.2	87.0
vegetation	71.0	73.1	80.5	80.0
trail	53.2	66.6	93.3	93.6
obstacles	58.7	64.1	71.3	73.2
fence	0	0	0.5	41.7

The fence class was a special case. The fences where heavily occluded by vegetation and difficult to detect. Neither network was able to detect them. This may improve with further training or by adding more instances of the class to the dataset. Shown in table 4.3, the increase in detection time between Dilated SqueezeSeg and the original is negligible. Dilated SqueezeSeg can also handle real time object detection.

The Dilated SqueezeSeg network with the Autoencoded Pixel-Block Sensor Fusion (APBS) method from chapter 3 realized accuracy gains over the Dilated SqueezeSeg network using lidar data only. However, standard SqueezeSeg with APBS marginally outperformed Dilated SqueezeSeg with APBS. Also, standard SqueezeSeg with APBS was able to detect the fence class unlike

Table 4.2

Table to compare the Precision/Recall accuracy of SqueezeSeg, Dilated SqueezeSeg, Dilated SqueezeSeg with Autoencoded Fusion, and SqueezeSeg with Autoencoded Fusion respectively. These networks were trained for 150,000 steps.

Classes	SqueezeSeg (SS)	Dilated SqueezeSeg (DS)	DS w/ Auto Fusion	SS w/ Auto Fusion
ground	83.8 / 84.7	88.6 / 84.1	94.9 / 91.3	94.4 / 91.8
trees	84.0 / 92.4	90.6 / 88.7	92.3 / 92.9	91.8 /94.3
vegetation	81.7 / 84.4	79.5 / 90.1	86.4 / 92.2	87.7 / 90.2
trail	82.1 / 60.1	84.8 / 75.6	96.9 / 96.2	95.8 / 97.6
obstacles	84.2 / 65.9	85.3 / 72.0	83.4 / 83.2	85.5 / 83.6
fence	0.0 / 0.0	0.0 / 0.0	24.0 / 0.5	68.9 / 51.4

Table 4.3

Table to compare the inference speed of SqueezeSeg to Dilated SqueezeSeg.

Method	Detection Speed
Baseline SqueezeSeg	13 ms
Dilated	14 ms
SqueezeSeg w/ Autoencoded Fusion	25 ms
Dilated SqueezeSeg w/ Autoencoded Fusion	30 ms

standard SqueezeSeg and Dilated SqueezeSeg. Dilated SqueezeSeg with APBS also had trouble detecting the fence class.

The networks were trained on an Alienware running ubuntu with 31.3 GB of memory, an 8th gen Intel i7 2.2GHz x12, and a GeForce GTX 1080 GPU with 8 GB of memory.

The deep learning networks were developed using tensorflow 1.14. Each network was trained on 150,000 steps with a batch size of 8. The learning rates were 0.003 with a momentum of 0.09 and a decay factor of 0.5. Data augmentation and random flipping was also applied during training.

4.6 Conclusions

The proposed Dilated SqueezeSeg categorically outperforms SqueezeSeg while maintaining real time detection speeds within 30-40 frames per second (fps). On average, Dilated SqueezeSeg is six percent more accurate than the original. As shown in table 4.1 and 4.2, the main accuracy gains were realized in the trail and obstacle classes. The accuracy increased by 13 and 5 percent respectively. The fence class was a special case. The fences where heavily occluded by vegetation and difficult to detect. Neither network was able to detect them. However, standard SqueezeSeg with APBS was able to detect the fence class unlike standard SqueezeSeg and Dilated SqueezeSeg. Dilated SqueezeSeg with APBS also had trouble detecting the fence class. This may be due to the size of the pixel blocks used in the fusion method. Dilated convolution has holes in the kernel. Therefore, the block size may need to be optimized in order to correlate with the sampling frequency of the dilated convolution operations.This may improve with further training or by adding more instances of the class to the dataset. Table 4.3 shows that the increase in detection time between Dilated SqueezeSeg and the original is negligible. The fusion methods as well as Dilated SqueezeSeg can also handle real-time object detection.

CHAPTER V

CONCLUSION

In this thesis, sensor fusion techniques and network architectures were explored for real-time object detection in the off-road environment. The preprocessing techniques and proposed architectures successfully realized improvements from previous methods. MAVS generated the sensor data and the ground truth.

The sensor fusion techniques involved autoencoders, lidar point cloud data, and camera data. To fuse the sensor data most efficiently, a group of pixel around each in-frame lidar point where dimentionally reduce using the encoder section of the trained autoencoder. The encoded rgb data was combined with the lidar data creating a 31-dimension feature vector. SqueezeSeg was modified to accept the new input. This sensor fusion method proved to increase the accuracy of SqueezeSeg and can used in other network architectures as well.

The new network architecture involved modifying the fire modules within SqueezeSeg creating a new architecture termed Dilated SqueezeSeg. The proposed fire modules contained dilated filters to increase the receptive field of the network. As the network progressed, the dilation factor was gradually increased to create large scale context aggregation within the feature extractors. Also, the Dilated SqueezeSeg Architecture was further modified to accept the autoencoder sensor fusion feature vector as outlined above. Both of these techniques proved to increase the accuracy of SqueezeSeg.

5.1 Future Work

There are a few areas to address in future work. The data used to test these methods was synthetic data generated by MAVS. The next step is to test these methods using real-world data. However, utilizing real-world data comes with another set of challenges. The data must be labeled via segmentation. Also, the wide angle camera and lidar sensors will need to be calibrated and co-registered for sensor fusion. The camera distortion created by the wide angle parameters will also have to be accounted for before data fusion. Furthermore, the autoencoder outlined here is pretrained to encode a compressed representation of small pixel blocks. To maximize the accuracy of the encoder, new data can be introduced to the autoencoder in real time constantly refining the encoder. By increasing the effectiveness of the encoder, the accuracy of the overall model should also increase. Also, more testing can be done to determine the optimal dilation factors for the convolution filters in the fire modules.

REFERENCES

[1] N. H. T. S. Administration, “Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey,” *TRAFFIC SAFETY FACTS*, 2015.

[2] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, “LIDAR–camera fusion for road detection using fully convolutional neural networks,” *Robotics and Autonomous Systems*, vol. 111, 2019, pp. 125–131.

[3] C. Chen, L. Z. Fragonara, and A. Tsourdos, “RoIFusion: 3D Object Detection from LiDAR and Vision,” *arXiv preprint arXiv:2009.04554*, 2020.

[4] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” *arXiv preprint arXiv:1605.06409*, 2016.

[5] L. B. Eliot, *Introduction to Driverless Self-driving Cars: The Best of the AI Insinder*, LBE Press Publishing, 2018.

[6] T. Foster, A. Dahal, and J. E. Ball, “Lidar and RGB Pixel Block Point-cloud Fusion for Object Detection,” *Porceedings of SPIE DCS 2021*, 2021.

[7] C. Goodin, D. Carruth, M. Doude, and C. Hudson, “Predicting the Influence of Rain on LIDAR in ADAS,” *Electronics, 8(1), p.89*, 2019.

[8] C. Goodin, M. Doude, C. Hudson, and D. Carruth, “Enabling Off-Road Autonomous Navigation-Simulation of LIDAR in Dense Vegetation,” *Electronics, 7(9), p.154*, 2018.

[9] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka, “Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery,” *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018, pp. 1442–1450.

[10] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[11] W. Lou, “Understanding the effective receptive field in deep convolutional neural networks,” *arXiv preprint arXiv:1701.04128*, 2017.

[12] W. Luo, Y. Li, R. Urtasun, and R. Zemel, “Understanding the Effective Receptive Field in Deep Convolutional Neural Networks,” *arXiv preprint arXiv:1701.04128*, 2017.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[15] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10529–10538.

[16] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2569–2578.

[17] Y. Wang, "Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *arXiv preprint arXiv:1807.06288*, 2018.

[18] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893.

[19] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4376–4382.

[20] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.