On Semantic Segmentation and Path Planning for Autonomous Vehicles within Off-Road Environments

Christopher J. Holder

Supervised by Professor Toby P. Breckon

Thesis Presented for the degree of

Doctor of Philosophy

Department of Computer Science

Durham University

2018

Abstract

There are many challenges involved in creating a fully autonomous vehicle capable of safely navigating through off-road environments. In this work we focus on two of the most prominent such challenges, namely scene understanding and path planning.

Scene understanding is a challenging computer vision task with recent advances in convolutional neural networks (CNN) achieving results that notably surpass prior traditional feature driven approaches. Here, we build on recent work in urban road-scene understanding, training a state of the art CNN architecture towards the task of classifying off-road scenes. We analyse the effects of transfer learning and training data set size on CNN performance, evaluating multiple configurations of the network at multiple points during the training cycle, investigating in depth how the training process is affected. We compare this CNN to a more traditional feature-driven approach with Support Vector Machine (SVM) classifier and demonstrate state-of-the-art results in this particularly challenging problem of off-road scene understanding. We then expand on this with the addition of multi-channel RGBD data, which we encode in multiple configurations for CNN input. We evaluate each of these configuration over our own off-road RGBD data set and compare performance to that of the network model trained using RGB data.

Next, we investigate end-to-end navigation, whereby a machine learning algorithm optimises to predict the vehicle control inputs of a human driver. After evaluating such a technique in an off-road environment and identifying several limitations, we propose a new approach in which a CNN learns to predict vehicle path visually, combining a novel approach to automatic training data creation with state of the art CNN architecture to map a predicted route directly onto image pixels. We then evaluate this approach using our off-road data set, and demonstrate effectiveness surpassing existing end-to-end methods.

Contents

List of Figures

List of Tables

Abbreviations

Advanced Driver Assistance System	ADAS
Autonomous Land Vehicle in a Neural Network	ALVINN
Adaptive Support Weights	ASW
Convolutional Neural Network	CNN
Defence Advanced Research Projects Agency	DARPA
DARPA Autonomous Vehicle	DAVE
Fully Connected	FC
Fully Convolutional Network	FCN
Frames Per Second	fps
Global Positioning System	GPS
Graphics Processing Unit	GPU
Imagenet Large Scale Visual Recognition Challenge	ILSVRC
Inertial Measurement Unit	IMU
Intersection Over Union	IoU
Long Short-Term Memory	LSTM
Red Green Blue	RGB
Red Green Blue Depth	RGB-D
Recurrent Neural Network	RNN
Semi-Global Block Matching	SGBM
Speeded Up Robust Features	SURF
Support Vector Machine	SVM

Statement of Copyright

Acknowledgements

I would like to express my appreciation to my supervisor, Professor Toby Breckon, whose time and support for the past four and a half years have made this possible. Thanks must also be directed towards all past and present inhabitants of the Lovelace Lair, whose many ideas, distractions, and proof-reading skills have proven invaluable.

During this work I was offered the opportunity to spend time at the A*STAR Institute for Infocomm Research, for which I am very grateful. I am particularly thankful to Dr Xiong Wei for making me feel welcome within the Knowledge Assisted Vision lab.

I would like to acknowledge the industrial sponsor Jaguar Land Rover for providing the initial motivation for this work as well as their financial support. Particular thanks should go to Anna Gaszczak and Dereck Webster for their assistance in collecting much of the data used for this work.

Finally, I would like to express my gratitude to my family for all their help and support, in particular Ruth for always being there and for always understanding, even when we are on opposite sides of the world.

Chapter 1

Introduction

This thesis addresses two major research challenges relating to autonomous driving in an off-road environment, specifically scene understanding and path planning.

Figure 1.1 A vehicle being driven off-road.

Our scene understanding work involves the training of a classifier to label the constituent parts of an image of an environment. We compare a traditional computer vision approach, utilising an assortment of manually selected features to train a support vector machine (SVM) based classifier, with a state-of-the-art deep convolutional neural network (CNN) based classifier [1] both trained on our own off-road dataset (Chapter 3). We then investigate whether classification performance can be improved with the addition of depth features, evaluating several approaches to encoding depth information computed from stereoscopic disparity (Chapter 4).

Off-road path planning involves the identification of a safely traversable route through an unknown environment. We evaluate a state-of-the-art approach that uses a deep convolutional neural network to predict the vehicle control inputs made by a human driver

from a single image taken by a forward-facing vehicle-mounted camera [2] We then propose to improve on this approach through our own visual path prediction method, which takes the same forward-facing images and trains a deep convolutional neural network to map a path directly onto image pixels (Chapter 5).

1.1 Motivation

Autonomous driving is an area of research that has received significant recent attention from both academia and industry [3] [4], with clear benefits in terms of safety and convenience and decreases in congestion and environmental impact associated with vehicles capable of operating without a human driver. While this work is almost exclusively focused on the on-road environment, where inherent rules and structures remove much of the challenge associated with off-road driving, there are many applications, such as military transport [5], search and rescue, agriculture [6], and even planetary rovers [7] where there is a requirement for a vehicle to be capable of navigating unstructured environments autonomously.

There are many challenges involved in creating a fully autonomous vehicle capable of safely navigating any off-road environment that a human driver could handle, however two of the most prominent such challenges are scene understanding and path planning.

Scene understanding is necessary for providing context to the environment surrounding a vehicle, and in the off-road environment serves two purposes: Terrain classification and obstacle classification. Classification of terrain [8] [9] can aid in the identification of traversable routes and selection of mechanical parameters, such as those for suspension, gearbox and differential, appropriate for current levels of grip and roughness. Current off-road terrain response driver assistance systems, such as that used in Land Rover vehicles as shown in Figure 1.2, either require the driver to manually select the current terrain type, or rely on mechanical feedback, such as wheel slip or suspension travel, to identify

terrain characteristics. Classification of obstacles [10] can help to differentiate between those that can be driven over, such as low, flat rocks, or through, such as foliage, and those that cannot, such as tree trunks.

Figure 1.2 The terrain response selector in a modern Land Rover vehicle, including 'Auto' mode which relies on mechanical sensors to identify terrain type.

For the problems of both terrain and obstacle identification, we propose a deep convolutional neural network based approach, as similar methods have demonstrated state-of-the-art performance in other image classification tasks [11] [12].

Path planning involves identifying the optimal route through the immediate environment [13]. On-road solutions are generally only required to identify lane edges and place the vehicle between them [14], however off-road environments often do not contain clearly marked lanes and can include any number of potential hazards that must be considered. The variability and lack of structure inherent in such an environment mean that a skilled human driver will often rely on experience and intuition, potentially limiting the utility of any hard-coded rule-based approach. Whereas neural networks, as systems influenced by the architecture of the human brain, have the potential to excel at this kind of task by analysing large amounts of data demonstrating the decisions taken by human drivers and learning the patterns and features that lead to those decisions.

Any approach we propose, for either scene understanding or path planning, must feasibly be able to run in real-time on a vehicle, placing limitations on weight, cost, size, and power usage, on both computing devices and sensors used. The advent of powerful portable GPU solutions, such as Nvidia's Tegra system-on-a-chip [15], has introduced the possibility to deploy complex deep learning models onto embedded devices that can be installed in vehicles to process sensor data in real-time. For input data, we use images taken by a single colour camera, keeping cost, weight, and power requirements low.

1.2 Thesis Contributions

The contributions in this work advance the state of the art in the following areas:

- Adapting state-of-the-art deep learning techniques to the problem of off-road scene segmentation and classification (Chapter 3).
- Evaluating two novel approaches aimed at improving off-road scene segmentation and classification performance: stereoscopic depth features and temporal consistency (Chapter 4).
- Creating and evaluating a novel approach to off-road path planning, by training a convolutional neural network to visually predict future vehicle path using data created autonomously by visually tracking a human-driven vehicle (Chapter 5).

1.3 Publications

Results from this work were published as the following:

- Christopher J. Holder, Toby P. Breckon, and Xiong Wei. "From on-road to off: transfer learning within a deep convolutional neural network for segmentation and classification of off-road scenes." 14th European Conference on Computer Vision Workshops, pp. 149-162, Springer International Publishing, 2016.

- Christopher J. Holder and Toby P. Breckon. “Encoding Stereoscopic Depth Features for Scene Understanding in Off-Road Environments.” 15th International Conference on Image Analysis and Recognition, pp. 1-8, Springer International Publishing, 2018
- Christopher J. Holder and Toby P. Breckon. “Learning to Drive: Using Visual Odometry to Bootstrap Deep Learning for Off-Road Path Prediction.” Intelligent Vehicles Symposium, pp. 1-7, IEEE, 2018

1.4 Scope

The scope of this work is limited by several factors, mainly the availability of data and hardware as well as the priorities of the project’s industrial sponsor.

No suitable datasets are publicly available, and so all data used to train and evaluate the approaches detailed in this work have been collected specifically for this work. This limits both the amount and variability of the available data, which was all captured within rural environments in the United Kingdom under a limited set of weather conditions. As such, work conducted using this data is intended only to demonstrate the efficacy of the described approaches in similar environments under similar conditions. Given additional datasets, this work could potentially be expanded on to function under other environmental conditions, however this is beyond the scope of this project.

All the data used in this work was captured using RGB stereoscopic cameras. Data from additional sensing equipment, such as LIDAR, RADAR or inertial measurement unit (IMU) could be used to supplement the collected data, either as additional input information or as additional ground truth to better evaluate results. However, one of the goals of the industrial sponsor of this work was to identify how much could be achieved using visual data alone, due to availability and cost considerations of other sensors and the ease with which cameras can be integrated with other vehicle systems.

1.5 Thesis Structure

Chapter 2 of this thesis will review the existing literature relating to autonomous driving, both on- and off-road, and deep learning, specifically convolutional neural networks as applied to scene segmentation and labelling problems. The current state of the art in terms of scene understanding and off-road path planning will be investigated extensively, and the limitations of current methods identified and discussed.

Chapters 3 to 5 will each discuss one of the three problems this work sets out to address. In Chapter 3 we discuss scene labelling, comparing a traditional computer vision approach, based on SURF features, K-Means clustering and SVM classification, to a state-of-the-art CNN-based scene labelling technique developed for urban road scene segmentation and classification [1], and evaluate the performance of both using our own challenging off-road dataset.

In Chapter 4 we discuss the methods we evaluated in our attempts to further improve performance of our chosen CNN-based approach to off-road scene labelling. We utilised stereoscopic disparity to compute several sets of depth features, including height and normal orientation, which we concatenated to the existing RGB data for CNN input, yielding slightly improved classification results.

In Chapter 5 we discuss path planning. We select a state-of-the-art approach, involving training a CNN to predict steering input from a single monocular image, that has demonstrated good performance in structured on-road environments, and re-train it using our own dataset to evaluate its performance in a more challenging off-road environment. After identifying several limitations of this technique, we propose our own novel visual path-prediction method, in which a CNN learns to segment an off-road vehicle's future path within image space, using a stereoscopic visual odometry technique to autonomously create training data.

Finally, in Chapter 6 we summarise the results of the previous three chapters and discuss the implications and potential future directions of the work.

Chapter 2

Literature Review

A huge body of research has been conducted in the field autonomous driving, from both academia and the automotive industry, with much notable work in the areas of scene understanding [4] and road detection [16]. However, the more challenging problem of off-road autonomous driving has received comparatively little attention. In the off-road environment, path planning can be much more difficult than on-road, due to uneven terrain, hidden obstacles and an overall lack of structure, however there are many real-world applications for such technology, including in agriculture [17], military [10], and planetary exploration [18]. Many recent works in the field make use of Convolutional Neural Networks (CNN), which have demonstrated unprecedented performance at a multitude of image classification tasks [11], revolutionising computer vision research. Loosely based on the biological brain, a CNN applies a series of learned convolutions to an input signal to generate an output. In this work, CNN based approaches are applied to the problems of off-road scene understanding and path planning, and as such this chapter discusses the current state-of-the-art in these areas as well as in CNN research in general.

2.1 Autonomous Vehicles

Vehicles capable of driving themselves has long been a desirable goal for governments, vehicle manufacturers and transportation industries, responsible for a significant body of research over several decades [18] [19] [20]. The main benefits relate to safety – a majority of vehicle accidents are the result of human factors [21] – and economics – training and employing professional drivers is costly. Other benefits include the decreased environmental impact of autonomous cars potentially being shared by many users rather

than owned and used by one, and the gains in efficiency and convenience that comes with vehicles able to coordinate and communicate traffic information and plan optimal routes.

2.1.1 Levels of Autonomy

In 2014 the Society of Automotive Engineers published a taxonomy for classifying vehicle autonomy levels [22], comprising six levels broadly as follows:

- Level 0 – the vehicle is under full control of a human driver, although may potentially have autonomous warning and intervention systems.
- Level 1 – the vehicle is capable of either autonomous steering or autonomous acceleration and braking, while a human must control all other functions.
- Level 2 – the vehicle can autonomously accelerate, brake and steer under certain conditions but a human driver must monitor the driving environment and be ready to intervene at any time.
- Level 3 – the vehicle can autonomously accelerate, brake and steer under certain conditions and is capable of monitoring the driving environment, however it may at times require a human driver to intervene.
- Level 4 – the vehicle is capable of performing all driving tasks autonomously but under certain circumstances may request human intervention, however unlike at level 3, if a human driver does not intervene when requested the vehicle is capable of safely leaving the road and coming to a stop.
- Level 5 – the vehicle is capable of performing all driving tasks that a human driver could perform with no need for human intervention.

2.1.2 Autonomous Road Vehicles

Many manufacturers currently sell vehicles with level 2 capabilities [23] [24], in 2017 Audi launched a level 3 system in their A8 model [25], and several prototype level 4

systems are being developed and tested [26] [27]. Commercially available level 2 systems generally rely on an array of vision, and radar sensors, whereby colour imagery is typically used to detect lane markings and road boundaries to inform steering decisions, and radar is used to detect the location and speed of other vehicles to inform acceleration and braking decisions. A front-facing stereo camera is often used to supplement radar sensors and detect obstacles such as pedestrians so that automated emergency braking system can be activated, while the level 3 system implemented by Audi also includes a laser scanner to provide more robust information about surrounding traffic.

Detection of lane markings and road edges is vitally important for any autonomous vehicle that navigates urban or highway environments. The visually distinctive pattern of brightly coloured markings contrasting with a dark road surface typically makes the problem of their detection trivial, as demonstrated by the significant body of work in this area going back more than twenty years [20] [28] [29]. Generally, these approaches take a grayscale image from a camera mounted at the front of the vehicle as input, extract the region of interest, i.e. the road surface directly in front of the vehicle, carry out a perspective transform to give a top-down view of this road surface, perform a Canny edge detection to find regions of high contrast, such as the boundary of lane markings, and finally apply a Hough transform to identify edge pixels that form consistent lines, from which current lane boundary lines can be determined. This approach forms the basis of many modern driver assistance systems, including lane departure warning, which notifies a driver if their vehicle approaches a lane boundary, lane keeping systems, which will actively steer a vehicle back into its lane if it approaches a boundary, and level 1, 2 and 3 autonomy systems, which constantly monitor lane boundaries and attempt to maintain a course as close as possible to the centre of the lane. Other approaches include that of [30]

which proposes a RANSAC based method combined with Kalman filtering to increase robustness against varying road conditions.

Obstacles, such as other vehicles and pedestrians, are usually detected through either laser, radar, or stereoscopic vision-based sensors. Both radar and laser rangefinders are active sensors, measuring some property of a reflected electromagnetic signal originally emitted from the vehicle. The distance to an object can be calculated based on the time taken for the emitted signal to be reflected. Stereoscopic sensors are passive sensors, relying only on light already present in the environment. The distance to an object can be computed based on the difference, or disparity, in its location within the images captured by two cameras positioned a known distance apart. On current production vehicles, the information from these sensors can be used to inform adaptive cruise control systems, which adjust vehicle speed based on relative speed and proximity of other vehicles, and autonomous emergency braking, which is able to slow or stop a vehicle if an obstacle is detected in its path.

Significant computational power is required if the data generated by these sensors is to be processed in real-time, a problem compounded by the size and power limitations of vehicular platforms. Traditionally integrated circuits would be built specifically for the task: Mobileye, a company that was bought by Intel in 2017 [31], has been producing its EyeQ system-on-chip since 2008, initially offering advanced driver assistance systems (ADAS) and working towards chips capable of level 5 autonomous driving by 2020 [32]. Portable computer systems, such as the Nvidia Drive PX-series [33] which integrate GPUs to greatly speed up computer vision and deep learning algorithms, offer greater flexibility. Drive-PX based systems are currently used by several vehicle manufacturers including Toyota [34] and Tesla [35] and will be a vital component of the autonomous vehicles used in the Roborace motor racing series [36].

In 2016 [37] Nvidia demonstrated a convolutional neural network running on their hardware performing real-time detection of multiple classes relevant to an on-road environment. Two versions of the network were shown, producing either bounding boxes or per-pixel labels as output. Also in 2016, Nvidia demonstrated their end-to-end autonomous driving approach [2], whereby a neural network learned to predict steering inputs after training with data demonstrating human driving. This end-to-end system was demonstrated in an autonomous road vehicle running on a Drive-PX 2 system.

2.1.3 Off road autonomous Driving

While much recent work has focused on autonomous road vehicles for civilian transportation, the off-road environment has been subject to relatively little research. This may be partly because off-road driving poses significant additional challenges: the lack of standardised structure and features such as lane markings make the detection of traversable paths more difficult; parameters of different terrain types such as grip and roughness require consideration; obstacles, both positive (for example rocks and trees) and negative (holes and dips in terrain) that could impede progress need to be detected [38]. However, a large amount of early autonomous vehicle research was motivated by military applications, for which off-road environments are a major factor.

Figure 2.1 Stanley competing in the 2005 DARPA Grand Challenge [13].

In 1989 Pomerleau et al [19] presented their Autonomous Land Vehicle in a Neural Network (ALVINN) system, supported by the United States military. They trained a 3-layer artificial neural network to output the direction in which a vehicle should steer given an image from a camera mounted at the front of that vehicle along with laser rangefinder data, and demonstrated their proposed system on the Carnegie Mellon Navlab vehicles in both on- and off-road environments. The main contribution of this work was the introduction to the autonomous driving literature of what would become known as '*end-to-end learning'*, whereby a neural network learns to control vehicle movement through training data capturing human control of that vehicle. In 2006, Muller et al [39] built on this idea with the DARPA Autonomous Vehicle (DAVE), which used a convolutional neural network to autonomously control an off-road radio-controlled vehicle.

In 2004, the United States Defence Advanced Research Projects Agency (DARPA) ran the first of their Grand Challenges, a competition tasking autonomous vehicles with completing a 240Km route in California. None of the entrants completed the course during the inaugural contest, however five of the twenty-three teams who entered the 2005 Grand Challenge reached the finish line.

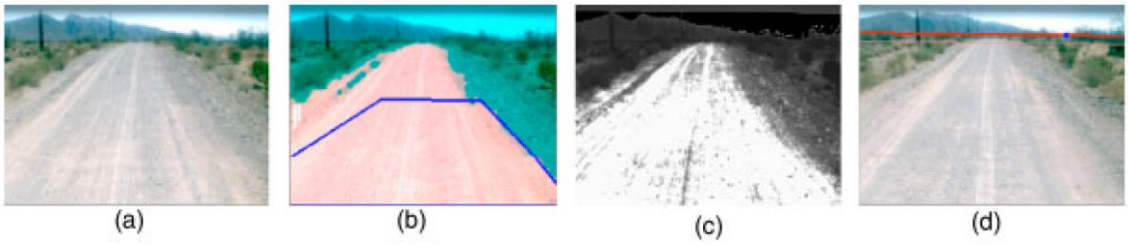

Figure 2.2 An illustration of the processing stages in the vision system of Stanley [13]. (a) a colour image received from the vehicle mounted camera; (b) the quadrilateral defined from LIDAR data with pixels classified as drivable (red) and not drivable (green); (c) unthresholded road probability values; (d) horizon detection step, used to remove irrelevant pixels.

Of the vehicles that successfully completed the course, Stanley, built by Thrun et al at Stanford University [13] was fastest. Based on a Volkswagen Touareg SUV modified to increase resilience to rough terrain and allow computer control of all driver functions,

Stanley, shown in Figure 2.1, senses its environment using five laser rangefinders, a colour camera, and two radar sensors, all mounted on the vehicle roof. Data is also collected from Global Positioning System (GPS), compass, and inertial measurement unit (IMU) sensors, and processing is carried performed by an array of six computers housed in the boot.

While the overall route of the DARPA Grand Challenge is predetermined and followed using GPS data, the local path planning approach of Stanley is based on sensed data from the laser range finders and camera mounted on the vehicle. Laser rangefinder data is used to build a 3D map of the environment in front of the vehicle, from which any two nearby points whose vertical distance is greater than a threshold can be considered an obstacle. Ground containing obstacles is considered '*occupied*', while flat ground is considered '*drivable*'. To find a safe path through the remaining drivable ground, a relatively flat quadrilateral between 10 and 20 metres in front of the vehicle is selected from the laser rangefinder data, for which a gaussian mixture model of pixel colour can be built and updated frame by frame as the vehicle progresses. This model is then used to identify visually similar areas of terrain which can be considered traversable and a path is propagated forward from the safe area, as shown in Figure 2.2.

Figure 2.3 The Terramax autonomous truck that entered the 2005 DARPA Grand Challenge [5].

Another competitor that completed the course, albeit much more slowly, was TerraMax, shown in Figure 2.3, developed by a team led by Oshkosh Corporation, a builder of military vehicles [5]. Based on an Oshkosh Medium Tactical Vehicle Replacement, Terramax is a six-wheel-drive tactical truck that senses its environment using an array of LIDAR sensors and a trinocular vision system. Stereoscopic disparity is used to detect orientation of terrain in front of the vehicle so that a safe path can be detected along with any obstacles present. Path planning is achieved through a clustering method which groups homogenous pixels up to 50 metres in front of the vehicle. Each of these clusters is then assigned a probability that it is part of the road using a Bayesian network that considers the visual properties of the cluster as well as current vehicle state, as shown in Figure 2.4. Obstacles are marked at any point where computed disparity deviates significantly from detected ground orientation, and precise obstacle location is refined through fusing stereo triangulation and LIDAR data.

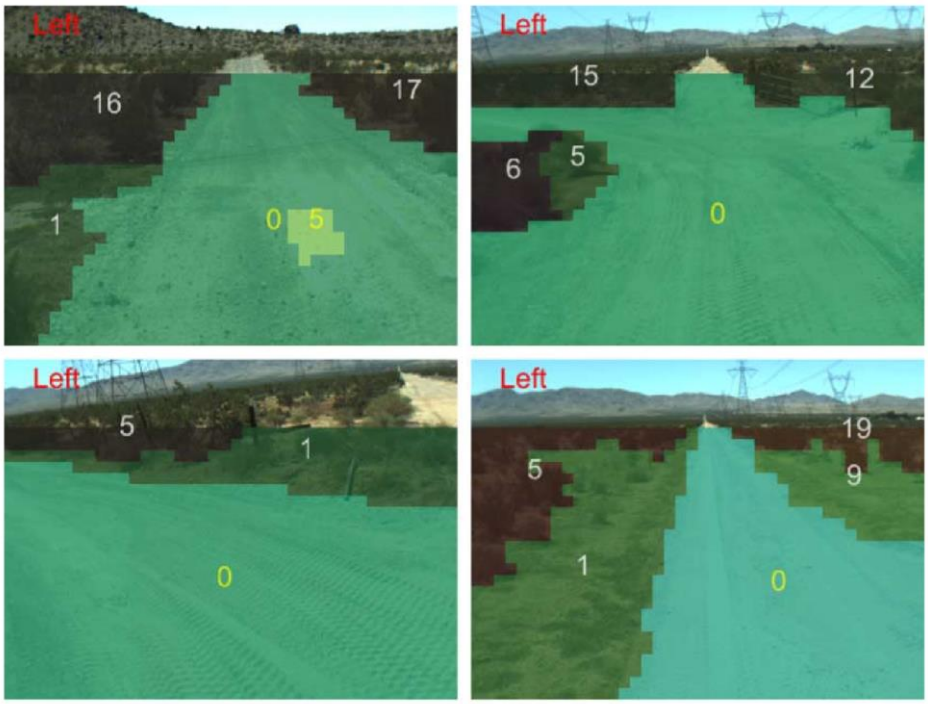

Figure 2.4 Results from the path detection approach of Terramax [38].

2.2 Convolutional Neural Networks

The idea of an Artificial Neural Network was first proposed by McCulloch and Pitts [40], who described how connections in the human brain might function using a computational model. A significant body of further work built upon these ideas over the next few decades [41] [42], however by the turn of the century other approaches such as Support Vector Machine (SVM) [43] and Decision Tree based methods [44] came to prominence.

In 2012, Krizhevsky et al. [11] brought about a resurrection of sorts for the neural network, with the Alexnet architecture illustrated in Figure 2.5. The era of Deep Learning, as it has become known was made possible by two recent advances: The availability of relatively cheap massively parallel processing, in the form of Graphics Processing Units (GPU) initially designed to perform the many simultaneous graphical computations required by 3D video games, has made it possible to train complex models that would previously have taken an infeasibly long amount of time; The availability of large datasets with which to train such models, with the advent of so-called '*Big Data'*.

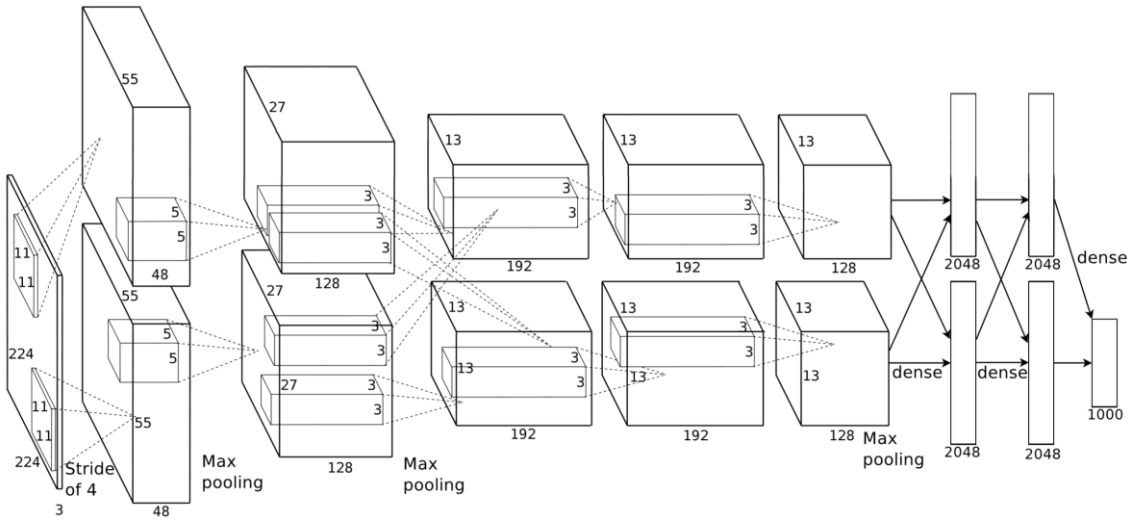

Figure 2.5 The CNN architecture proposed by Krizhevsky et al [11], illustrating how processing is divided between two GPUs.

The ImageNet dataset [45], comprising about 14 million images (some examples of which are shown in Figure 2.6) classified using a hierarchical structure with about 22,000 labels at the highest level of granularity, was created to train and evaluate object classification, detection and localisation approaches. The Imagenet Large Scale Visual Recognition Challenge (ILSVRC) [46] is an annual series of contests that has been used to assess such approaches since 2010, using a subset of the Imagenet dataset comprising about 1.5 million images and 1000 classes. ILSVRC broadly consists of three challenges: classification, localisation and detection. In the classification contest, each method should output a list of up to 5 classes it believes are most likely to be present in an image. For each image there is a single ground truth label, and prediction error is counted as 0 if this label is included within the 5 predictions, or 1 if it is not. Before the advent of Deep Learning, the best classification result was achieved by [47] in 2011, with a top-5 classification error of 25.8%. In 2012, Krizhevsky et al. [11] achieved a classification error of 16.4% on the same challenge using a deep learning based approach. As of 2017, the best performing approach, Squeeze-and-Excitation Networks by Hu et al. [48], achieved an error of less than 2.5%. In 2014 Karpathy [49] demonstrated that a human

can perform the same task with a classification error of 5.1%, suggesting CNN based approaches are now capable of greater-than-human performance at this particular task.

Figure 2.6 Images from the ImageNet [45] dataset embedded by Karpathy ***Invalid source specified.*** *using t-Distributed Stochastic Neighbour Embedding (t-SNE) [51] such that images with similar labels should be close to each other.*

Generally, CNN architectures for addressing this task are designed along broadly similar principals [52] [11] [53]. Images are convolved with a series of kernels, each populated with values optimised during training. Interspersed between these convolutions are nonlinear activation functions, which determine whether a particular value should be passed onto the next layer of the network, and pooling operators, that downsample their input by a given scale, both to reduce the number of parameters required by the network and so that later convolutions are receptive to a larger area of the original input. These operators are often followed by several Fully Connected (FC) layers, in which every entry in the input data is connected to every entry in the output, essentially a convolution operation where the kernel size matches the dimensions of the input data. An FC layer discards spatial information that might otherwise be retained by standard convolutions, however it ensures output takes account of the entire feature map generated by previous layers. The output of the final network layer is a one-dimensional vector of floating point

values of length *c*, where *c* is the number of possible class labels that could be assigned to an image, with each entry describing the network confidence that the respective label is the correct one.

During training, the network will be run for many thousands of iterations using data for which ground truth output is known. The generated class confidence vector that is output for each image is normalised by softmax function so that training loss can be calculated against this ground truth. The loss is then backpropagated through the network using gradient descent so that the parameters of each layer can be tweaked, and training continues until parameters are deemed to be optimised.

A massive body of work has extended this basic formula by altering the number and order of network layers [54] [55], inserting additional operations, such as batch normalisation [56], which normalises the data between layers such that each batch has a mean of 0 and a standard deviation of 1, regularising the input to subsequent layers and increasing the speed with which the network optimises; or dropout [57], which sets randomly selected values within the data to zero to reduce the likelihood of overfitting.

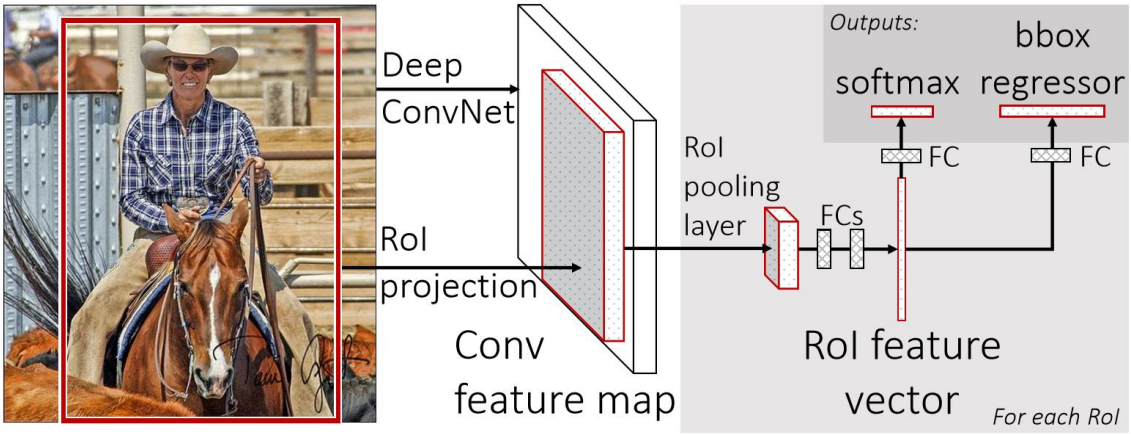

Figure 2.7 Fast R-CNN [58] outputs bounding box coordinates for detected objects.

Approaches that have extended this one image to one label relationship include [59] which assigns multiple labels to a single image, for example when multiple different objects are present; [58], shown in Figure 2.7, which generates coordinates for bounding

boxes to localise detected objects; [1] which assigns a label to every pixel in the input image, enabling object segmentation as well as classification; and [60] which retains information from prior input data so that multiple frames from a video sequence can be considered when making predictions.

2.3 Scene Understanding

Scene understanding is a widely researched topic, with many varied solutions to the problem of segmenting and labelling all parts of a sensed environment. Data may be 3D, in the form of point cloud [61] or voxel grid [62], obtained by active sensor, such as LIDAR [10] or structured light [63], or passive sensor, such as stereoscopic camera [4] or structure from motion [64]. 2D data may be obtained from visible light [65], infra-red [66] or other hyperspectral imagery [67]. Prior to deep learning based approaches, 2D scene understanding methods would typically employ a hand-crafted selection of features to build a descriptor that can be fed into a trained classifier to generate a label. The features used could be derived from local colour [68] or texture [69] information or a combination of the two [65], and classification could be performed by a Support Vector Machine [70], decision tree [71] or a nearest neighbour [72] based approach. The generated label may apply to a single pixel [73], an image region, either of fixed size and shape [74] or generated by a segmentation algorithm [75], or to the whole scene [76].

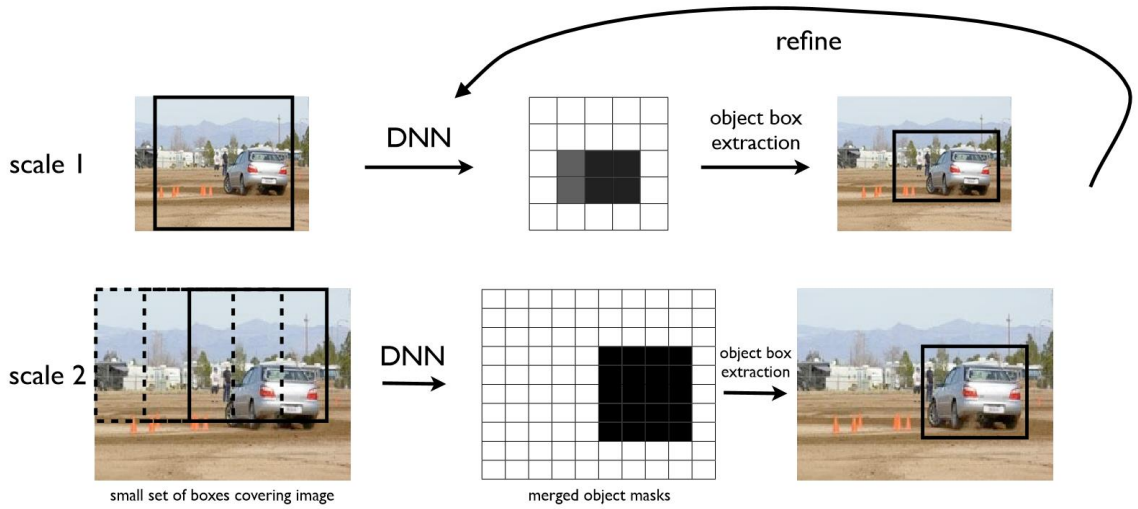

Figure 2.8 The approach of Szegedy et al [77] first detects regions of interest at low resolution then refines detections at higher resolution in cropped images.

More recently, deep-learning based methods have demonstrated state of the art results at a multitude of image recognition tasks [11] [78], and as such have come to dominate semantic scene understanding research. Several approaches have been proposed utilising deep CNNs to address the problems of object localisation and full scene labelling: The approach of Szegedy et al. [77], illustrated in Figure 2.8, creates a downscaled output mask which is subsequently refined at larger scales. The network proposed by Farabet et al. [79] outputs a label for a single pixel each time it is run, such that the network must iterate once for each pixel in the input image requiring a label, with regional consistency then enforced by conditional random field. The approach of Girshick [58] uses a selective search algorithm to propose regions, each of which are then passed through the CNN and classified.

The approach of Fully Convolutional Network (FCN) [80] creates a downsized class map that is upsampled by reverse pooling operations to create a final output comprising a vector of class label confidence values for every pixel in the original input image. Segnet [1] and u-net [81] propose similar approaches, with the additional step of reusing output from prior pooling operations to better inform corresponding upsampling operations and retain higher resolution spatial information that would otherwise be lost. These

approaches demonstrate state of the art segmentation and classification results in a wide variety of scenarios, including medical imagery as shown in Figure 2.9.

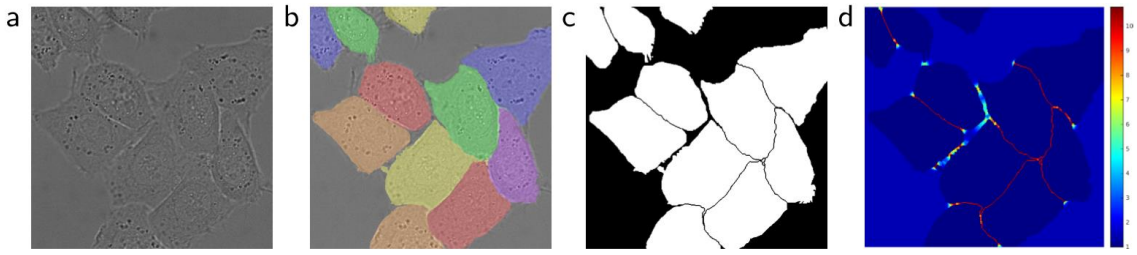

Figure 2.9 U-net [81] was motivated by biological image segmentation, however can easily be adapted to multi-class scene segmentation tasks. a. raw data b. ground truth segmentation c. u-net output d. pixelwise loss weighting to improve border detection.

Several methods have been proposed that attempt to improve results in scene understanding and semantic segmentation tasks, including the combination of CNN and conditional random field (CRF) by Zhang et al [82], the introduction of a pyramid pooling module by Zhao et al [83], which extracts features at multiple scales which are then concatenated, and the addition of temporal information when working with video frames [84] [85] [86]. Temporal approaches include recurrent neural networks (RNN) and long-short-term-memory (LSTM) modules, originally proposed in [87], which retain information over time to inform future predictions.

Scene understanding is a vital step in an autonomous vehicle processing pipeline, but this can be especially challenging in an off-road, unstructured environment. Knowledge about upcoming terrain and obstacles is necessary for deciding on the optimum path through such an environment, and can also be used to inform vehicle driving parameters to improve traction, efficiency and maximise passenger comfort and safety.

There is, however, relatively little scene understanding work focused on the off-road environment, where distinct but visually similar classes and a lack of structure can pose particular problems for computer vision. The approach of Jansen et al. [73], shown in Figure 2.10 in comparison with two earlier approaches, classifies pixels using colour-

based features and Gaussian Mixture Models, while Manduchi et al. [10] uses a combination of features from colour imagery and 3D geometry from a laser range-finder.

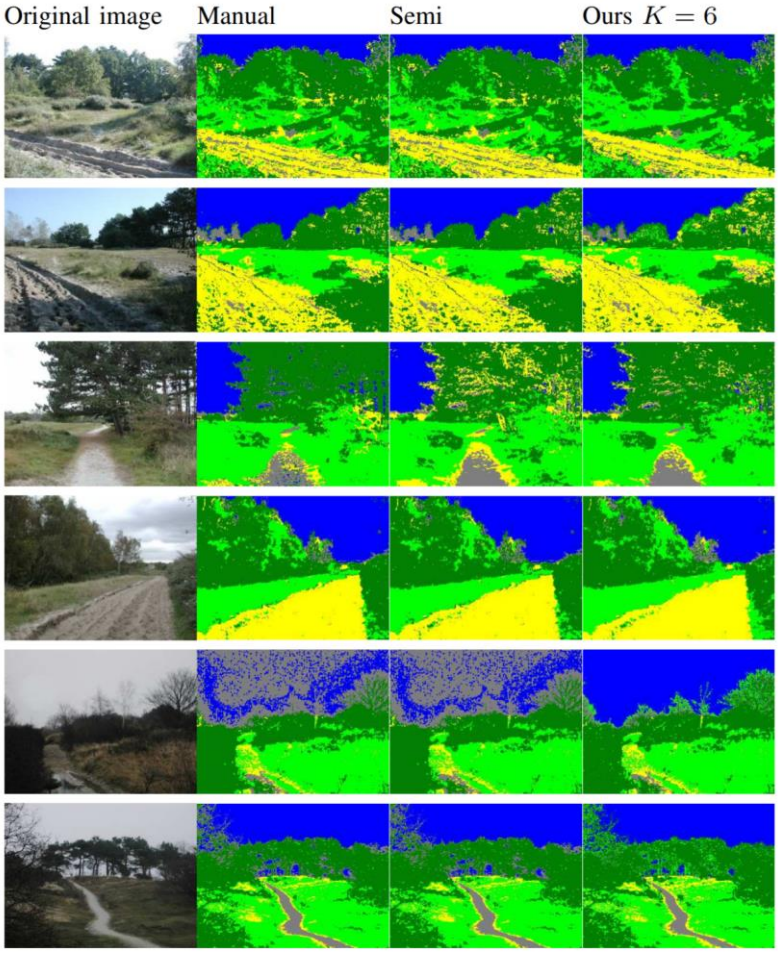

Figure 2.10 A comparison of off-road scene classification approaches from [73]. blue=sky, dark green=foliage, light green=grass, yellow=sand, grey=gravel.

2.4 RGB-D Features

Although the majority of scene understanding work only considers conventional 2D colour images (RGB), it has been demonstrated that the addition of 3D depth information (*D*) can improve classification performance in semantic scene understanding tasks, whether that information is obtained via laser scanner [10], stereoscopic camera [4] or structured light sensor [61].

In 2011, Silberman et al released their NYU Depth dataset [88] [89], some examples from which are shown in Figure 2.11. Captured using a Microsoft Kinect structured light depth camera, the NYU datasets contain approximately 4000 frames, each comprising a colour

image, aligned depth map and dense scene labels covering approximately 1000 classest, and have become a benchmark of sorts for RGB-D scene labelling approaches [80] [90].

Before deep-learning-based approaches became prominent, there were a wide assortment of approaches to the RGB-D labelling problem. Gupta et al. [90] propose a method which uses contour detection and hierarchical segmentation to generate superpixels which are then classified using an assortment of handcrafted depth and colour features and cascade SVM.

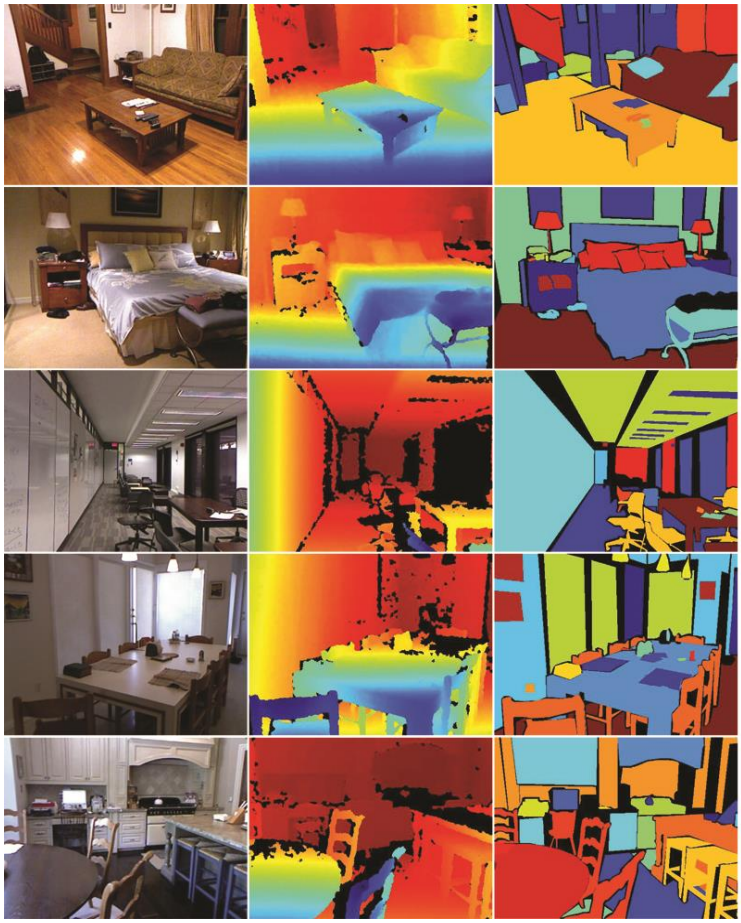

Figure 2.11 Example images from the NYU Depth dataset [88]. Left: RGB image Center: Depth image Right: Ground truth scene class labels.

Prior attempts to utilise depth information to improve CNN performance at classification tasks have shown that such features can bring about an improvement over results obtained from standard colour images: Gupta et al. [63] encode each pixel height, orientation and disparity into a 3 channel depth image to achieve state-of-the-art RGB-D object detection results, while Pavel et al. [91] utilise Histogram of Oriented Gradients and Histogram of

Oriented Depth, computed from the output of a consumer depth camera (Microsoft Kinect), to obtain competitive results in an object segmentation task.

2.5 Path Planning

A vitally important component of any autonomous vehicle processing pipeline is path planning. As opposed to route planning, which is concerned with navigation on a scale usually between metres and kilometres, often using GPS and pre-existing mapping information, path planning involves localising a vehicle at a much smaller scale, potentially centimetres, and identifying a path through its immediate environment, usually based on visual or 3D shape information obtained from cameras and LIDAR or other depth sensors.

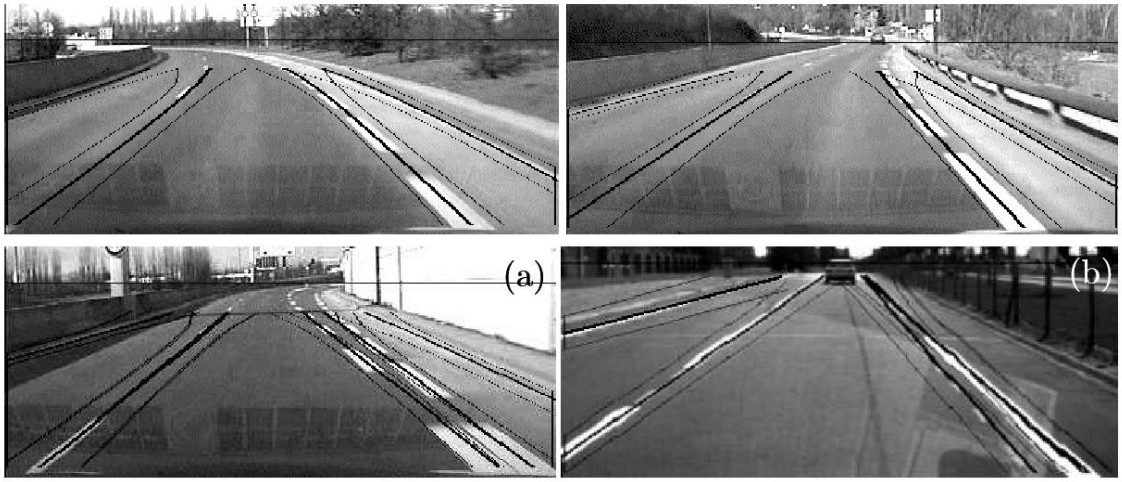

Figure 2.12 The approach of [28] detects lane markings using multiple models intended for different road types.

In on road environments, such as illustrated in, Figure 2.12 this can be achieved by identifying the edges of the road along with any lane markings present [20] [28], usually with a line detection algorithm such as the Hough Transform **Invalid source specified.**. This information can be used to compare the current position and trajectory of a vehicle to the centre line of the lane along which it is travelling and adjust driving inputs accordingly.

The concept of end-to-end autonomous driving was first proposed by Pomerleau in 1989 [19] with the Autonomous Land Vehicle in a Neural Network (ALVINN), which uses a neural network comprising a single fully-connected layer, taking a grayscale image and laser rangefinder data as input, trained to predict the steering wheel inputs made by a human driver. In 2004, the DARPA Autonomous Vehicle (DAVE) project [39], shown in Figure 2.13, trained a more complex, six layered network to drive a radio-control car in off-road environments, using data collected over several hours of human driving. More recent advances in deep-learning have led to the approach proposed in [2], which uses a network of 5 convolutional layers and 3 fully-connected layers, trained with 72 hours of human driving data, to successfully follow lanes on public roads. In all three approaches, a neural network is fed an image from a vehicle mounted camera and trained to predict the steering input a human driver would make at the time the image was captured.

Figure 2.13 Example images from the data used to test DAVE [39]. The black bars represent predicted steering direction and magnitude.

2.6 Summary

Despite a large body of early work investigating autonomous off-road driving [19] [13] [10], most significant recent work has focused on road-based driving, addressing problems such as lane detection [30], autonomous navigation [2] and urban scene understanding [1]. Off-road environments pose several major additional challenges to

autonomous driving, including hidden obstacles, variability of terrain and overall lack of scene structure. The approaches detailed in subsequent chapters are intended to address some of these challenges.

Chapter 3

Off-Road Scene Understanding

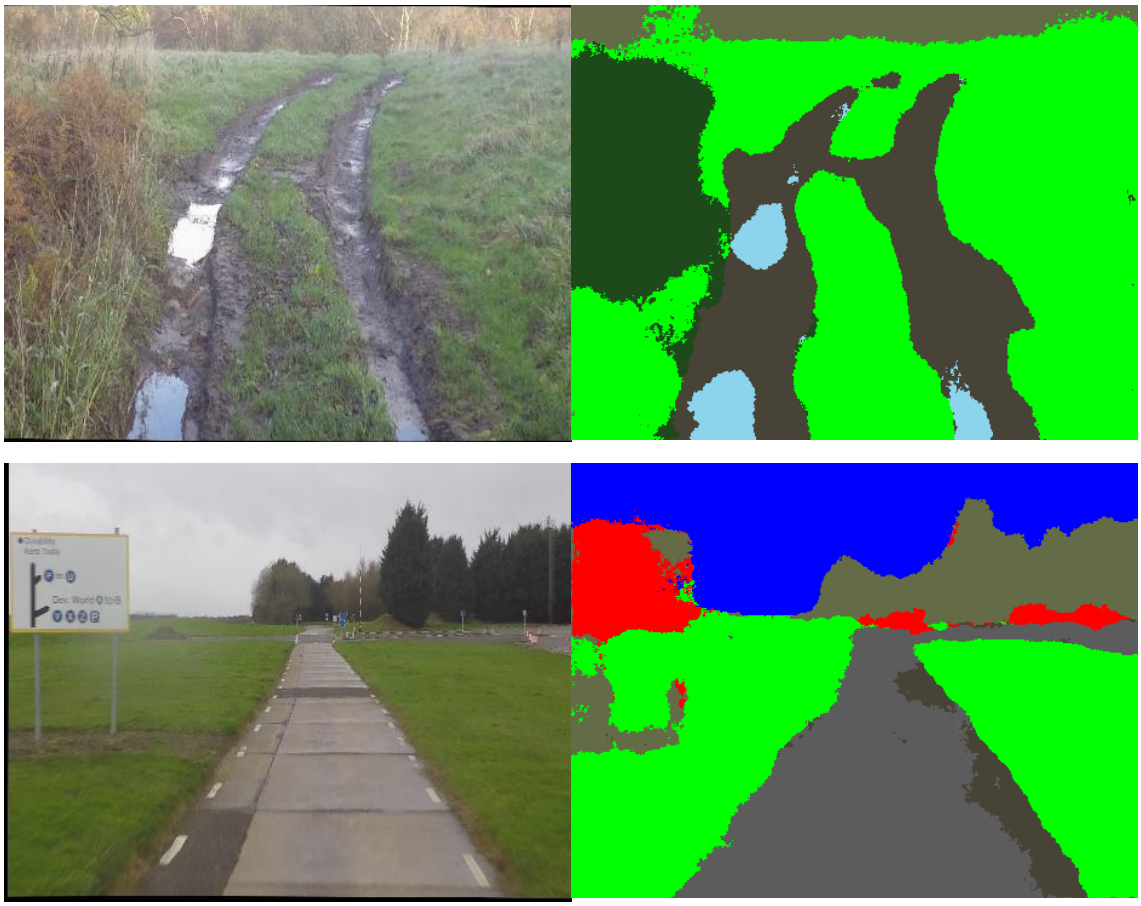

Figure 3.1 Example off-road images along with corresponding semantic segmentations, whereby every pixel is assigned one of eight labels: grass (light green), dirt/mud (grey), water (light blue), bushes (dark green), trees (brown), sky (dark blue), paved road (dark grey) or man-made obstacle (red).

Real-time road-scene understanding is a challenging computer vision task with recent advances in convolutional neural networks (CNN) achieving results that notably surpass prior traditional feature driven approaches [76]. Comparatively little work investigates the applicability of these techniques to the more challenging off-road environment, where the lack of structure and visual homogeneity of classes pose additional difficulties.

In this chapter, we describe our work addressing this problem of off-road scene understanding. Firstly, we implement and evaluate an approach based on traditional

computer vision methods that have demonstrated state-of-the-art performance at other tasks, utilising a feature-driven Support Vector Machine (SVM) classifier based on the work of [93], and demonstrate results comparable to existing work. Furthermore, we take a state-of-the-art CNN architecture [1], designed to perform full pixelwise labelling and segmentation of images, that has been pre-trained on an urban road-scene understanding dataset, and retrain it towards the task of classifying off-road scenes. We evaluate several configurations of this CNN, and demonstrate state-of-the-art performance in this particularly challenging problem of off-road scene understanding. Figure 3.1 illustrates the goal of this work, with input images of off-road scenes shown on the left, and their corresponding CNN output on the right, showing colour coded semantic segmentation results.

3.1 Introduction

Scene understanding is a vital step in an autonomous vehicle processing pipeline, but this can be especially challenging in an off-road, unstructured environment. Knowledge about upcoming terrain and obstacles is necessary for deciding on the optimum path through such an environment and can also be used to inform vehicle driving parameters to improve traction, efficiency and maximise passenger comfort and safety.

Work in the domain of scene understanding for autonomous vehicles has followed the wider trend in computer vision of transitioning to deep learning-based techniques, such as that of Alvarez et al [94], and Badrinarayanan et al [1], shown in Figure 3.2. However, there is very little work applying these methods to the more challenging off-road environment. This work aims to assess the applicability to such an environment of two approaches that have previously demonstrated good performance at other scene understanding tasks: a traditional computer vision method that makes use of hand crafted dense gradient features and an SVM classifier, based on the object category retrieval work

of [93], and a state-of-the-art CNN architecture that was originally designed and trained to perform per-pixel classification on urban road scene images [1].

In training this CNN we perform transfer learning, taking a network model that has already been trained to classify a large, often more generic data set, and fine-tuning its parameters during the final phase of training using a specialised, and often much smaller, data set, such that it learns to perform a new task.

In this case, a CNN trained to perform urban street scene classification using the CamVid dataset [95] is subsequently re-trained with a smaller, more specialised data set of off-road scenes. The idea is that the weights learned on the larger data set act to build a set of generic image filters that can be easily adapted for the task of classifying the more specialised imagery used later [96]. Transfer learning is generally thought to be beneficial when training with a small, specialised data set or when the time to train a new network from scratch is not available, so we investigate the effects of data set size and training time on classification performance of network models that have undergone different amounts of pre-training or no pre-training at all.

3.2 Approach

We implement and evaluate both a convolutional neural network-based approach and a support vector machine-based approach. In this section we describe these approaches.

3.2.1 CNN Architecture

The convolutional neural network architecture we use is nearly identical to the Segnet architecture described in [1], with only minor changes made to the final layer of the network. These changes consist of altering the size of the final network output to accommodate the classes present in our off-road data set (8 classes, versus 11 in the CamVid data on which it was initially trained), and adjusting the weights that determine

per-class learning-rate bias in order to improve classification results for classes that are less prominent within the training data. Similar network architectures exist [80], however we down-selected Segnet due to the authors' main motivation being autonomous vehicle applications and its ability to perform real-time classification.

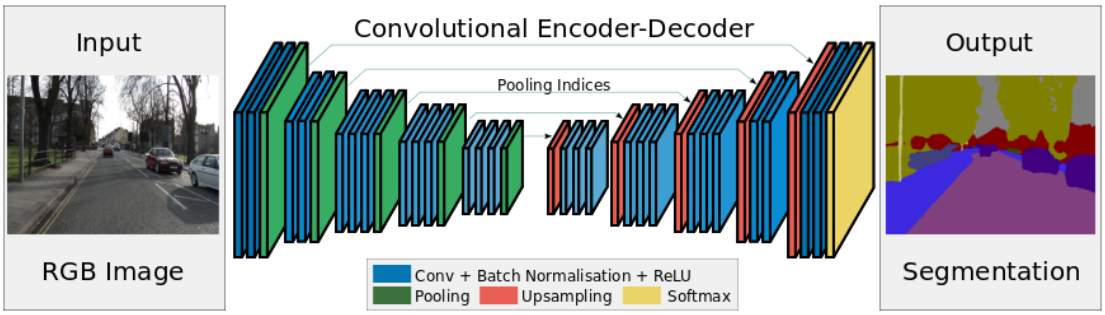

Figure 3.2 Architecture of the Segnet Convolutional Neural Network [1]. The encoder network, consisting of convolution and pooling layers, is followed by a mirror-image decoder network, consisting of convolution and up-sampling layers.

The Segnet architecture is visualised in Figure 3.2. It is comprised of a symmetrical network of thirteen encoder layers followed by thirteen decoder layers. The encoder layers correspond to the convolution and pooling layers of the VGG16 [53] object classification network, while the decoder layers up-sample their input so that the final output from the network has the same dimensions as the input image. During the encoding phase, each pooling layer down-samples its input by a factor of two and stores the location of the maximum value from each 2×2 pooling window. During the decoding phase, these locations are used by the corresponding up-sampling layer to populate a sparse feature map, with the convolution layers on the decoder side trained to fill in the gaps of a corresponding dense map. This technique facilitates full pixel-wise classification in real-time, making Segnet an ideal architecture for use in further autonomous vehicle applications.

3.2.2 CNN Training

We begin by training the network on the Camvid dataset [95] that was used by the original authors to assess Segnet. By training on a large, well labelled dataset that has already

been shown to work well with this network architecture we can ensure that our model learns a set of weights that are relevant to a vehicular scene understanding task. We then perform transfer learning, fine-tuning the model on our own off-road data so that its parameters adapt to better suit an off-road environment and discriminate between the classes present in these scenes.

Figure 3.3 Example images from the Camvid data set along with corresponding Segnet output with each pixel assigned one of eleven class labels: road (dark pink), vehicles (dark purple), pavement (light blue), buildings (red), sky (grey), poles (yellow), trees (light gold), cyclist (turquoise), road signs (light pink), fence (dark blue), pedestrian (dark gold).

The benefits of transfer learning are in the ease with which an existing trained network can be adapted to a new specialised task. The time taken to train the network and learn optimum weights should be greatly reduced when compared to a network being trained from an initial set of randomly generated parameters. In cases where the specialised data set is small or only partially labelled, a network trained from a random starting point may never achieve satisfactory results, however by performing the bulk of training with a

larger set of data and only utilising the specialised data set for the last few iterations, a better outcome can be achieved [96].

In our case, the initial training data consists of 367 labelled images of urban street scenes from the Camvid data set, resized to a resolution of 480 × 360. An example image from the dataset, along with its annotations, can be seen in Figure 3.3. The original authors chose eleven pixel classes for the Segnet classification task, *{sky, building, pole, road marking, road, pavement, tree, sign, fence, car, pedestrian, bicycle}*. As the network architecture performs classification of every pixel, this gives us up to 172,800 samples per image, or 63,417,600 samples in total. In practice, the total is slightly less than this as some images have pixels that do not fit into any of the eleven original classes and are labelled '*void*'.

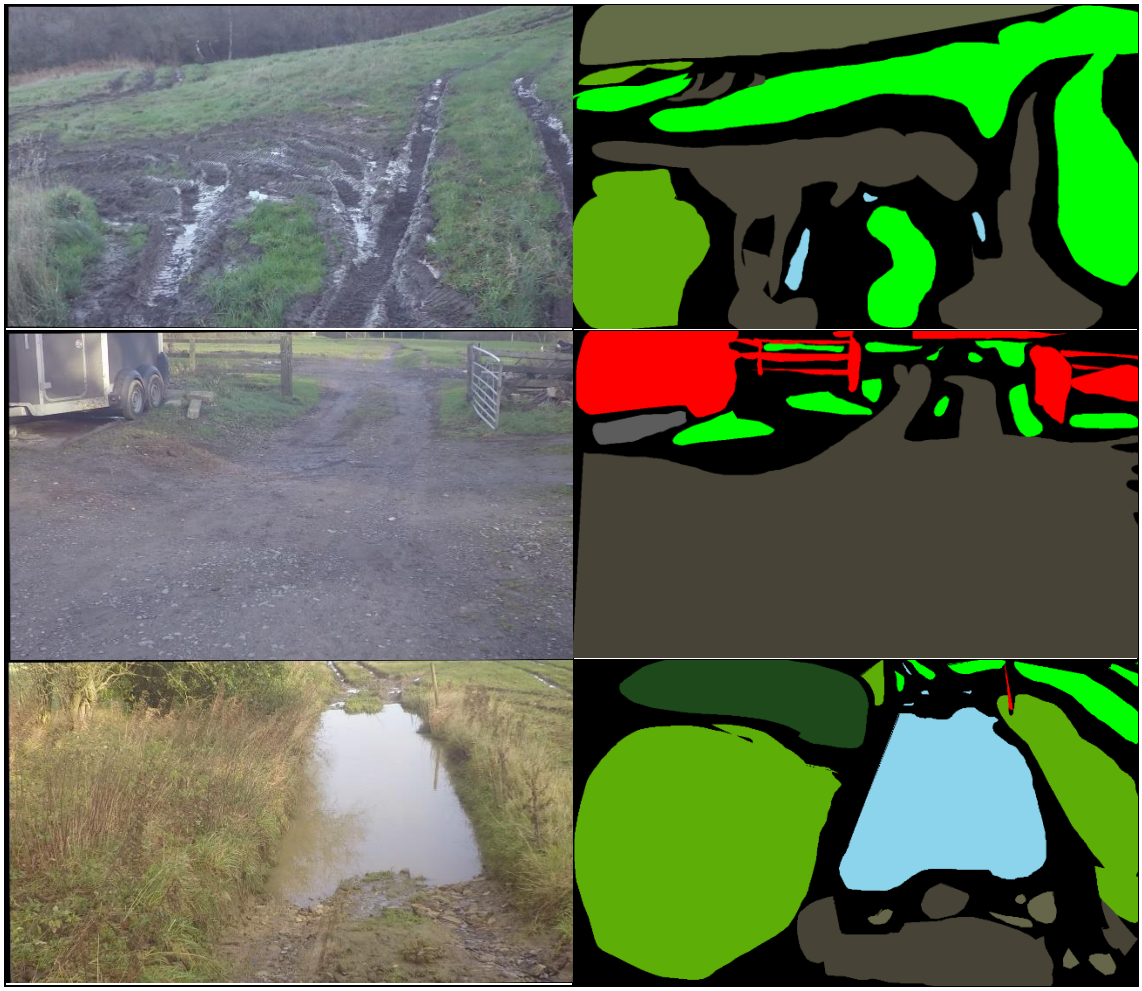

Figure 3.4 Example images from our off-road data set along with corresponding partially labelled ground truth

After the Camvid pre-training is completed, the network is fine-tuned using our own off-road dataset, which consists of 369 annotated images, split into 295 training images, 37 validation images and 37 test images. Images were captured by vehicle mounted camera driven at two different off-road driving facilities in the United Kingdom. Limited resources available for capturing and annotating data limited the size and variability within the data set, and so the scope of this work is constrained accordingly to wet, muddy agricultural environments in the United Kingdom.

For input to our CNN model, these images have been resized to the same resolution of 480 × 360 as the Camvid images. We identified 8 pixel class labels for our off-road data set, *{sky, water, dirt, paved road, grass, foliage, tree, man-made obstacle}*, 3 of which also existed in the Camvid data.

Fully labelling every pixel in even a small set of images can be very time consuming, so we only partially label our training images, as shown in Figure 3.4, to assess whether good classification results can still be achieved without full labelling. Our labelling strategy consists of hand drawing a shape that is entirely contained by, but not touching the edges of, each image segment. Every pixel within that shape is then considered a member of the chosen class. Another reason this approach was chosen is the lack of clear boundaries to delineate classes in off-road scenes, for example where long grass becomes foliage or a muddy surface is interspersed with patches of grass, as can be seen in the top image of Figure 3.4.

Figure 3.5 Partial and fully labelled versions of the same image from our off-road data set.

This gives us with a total of 35,016,288 labelled pixels for training, with the rest of the pixels (~31% of the total) labelled as void so that the network ignores them. For testing our classification results, we use one set of 37 images labelled in the same manner, as well as another set of 4 fully labelled images. Figure 3.5 shows the partial and fully labelled versions of one of our test images.

To evaluate the effectiveness of transfer learning when task-specific fine-tuning is carried out with a small data set, we train several versions of the network using different sized subsets of this data, one each containing 140, 70, 35, 17, and 8 of the original images.

Snapshots of the model are taken at several points during the Camvid training, so that we can observe the effects of different amounts of pre-training. Seven versions of the network will be trained and assessed on our off-road dataset: one which has been randomly initialised with no prior training, along with networks trained for 1000, 2000, 5000, 10,000, 20,000 and 30,000 iterations on the Camvid data.

Our training is performed on an NVidia Tesla K40 GPU, taking roughly one hour per thousand training iterations. We pass images through the network in batches of 6, with pixelwise softmax loss computed across each batch backpropagated by stochastic gradient descent. An initial learning rate of $1x10^{-3}$ is used, reduced by step function as training progresses, with momentum of 0.9 and weight decay of $5x10^{-4}$.

3.2.3 Traditional Computer Vision-Based Approach

For comparison, we train an SVM to classify the same data using dense gradient features, based on the approach used in [93] for object classification. For our approach we classify image segments, as this allows us to cluster feature points to build up a bag-of-words vocabulary. The segments in this case are those created manually while labelling the data, as in this case we are only interested in the performance of the classifier itself and so a perfect segmentation is assumed.

A dense grid of feature points is computed for each segment, from which we obtain a dense feature descriptor. The grid density, g pixels between grid nodes in both *x* and *y* direction, is chosen empirically by testing values between 2 and 10 pixels. Generally, a denser grid should contain a greater amount of information at the expense of computation time, so a lower number should give better results in most cases. Figure 3.6 illustrates this dense grid overlaid onto an example segment from our data set.

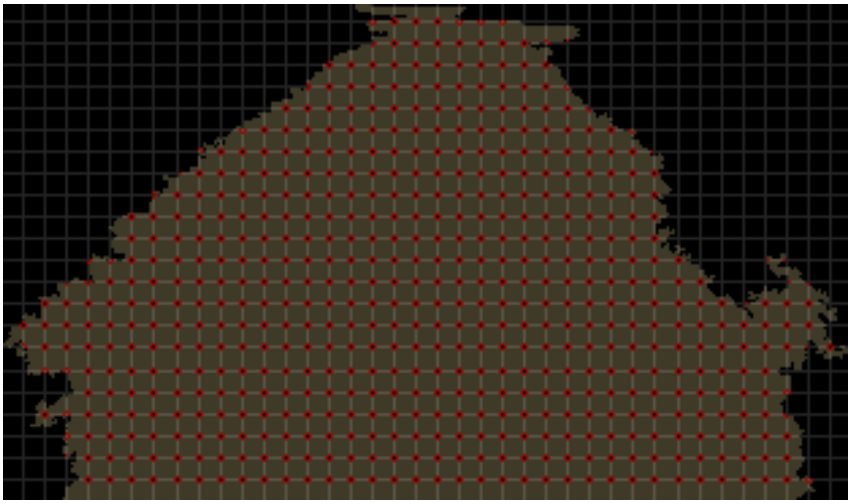

Figure 3.6 An example segment from our data set with a grid overlaid to illustrate feature point distribution. Points highlighted in red are determined to be within segment bounds and subsequently have their local features computed for classification.

The Speeded Up Robust Features (SURF) algorithm [97] is used to create a descriptor for each grid node. A SURF descriptor computes Haar wavelet responses within a square region around the initial point, which are summed to produce a vector describing the intensity distribution of pixels within the region. This results in either a 128 or 64

dimension vector that describes the local texture. Empirically we found a 64 dimension vector to give better results at this task. Every SURF descriptor is computed at the same orientation of 0 radians with a radius of *r* pixels. *r* is chosen empirically after assessing classification results using a range of values from 2 to 20 pixels. Descriptors resulting from this grid-wise feature extraction over the segment are encoded into a fixed length vector for subsequent classification.

We use the histogram encoding, or '*bag-of-words'*, approach detailed in [98]. We first use K-means clustering to create a visual vocabulary of *K* clusters within the 64-dimensional space of our SURF descriptors. The optimum value for *K* is chosen empirically, after testing values from 200 to 1600. For each segment, a histogram is computed accumulating the number of its SURF descriptors assigned to each cluster within the vocabulary.

This histogram is normalised to provide a *K*-dimensional descriptor for the segment, which we use as the input feature vector for an SVM classifier. We train this SVM classifier for a maximum of 20,000 iterations using a radial basis function to perform a grid search over the kernel parameter space.

To ensure enough local gradient information is available at each feature point, we use images with a resolution of 1280 × 720, higher than those used to train the CNN. The memory and time that would be required to train the CNN using images at this resolution would be infeasibly high, however by clustering our features before passing them to the SVM, the size of data it uses to train and classify is constant per sample regardless of image resolution.

Our labelled data set gives us 5664 labelled segments, which we split into 90% for training data and 10% for testing data. We ignore any samples too small to provide at

least 50 feature points, leaving us with between 3000 and 4000 viable segments, depending on the feature grid density used.

3.3 Results

We evaluate our classifiers using two sets of test data: a set of images partially labelled in the same manner as our training data, and a smaller set that are fully labelled (i.e. every single pixel in the image is labelled). The output layer of the CNN assigns a label to every pixel, while the SVM outputs a label for each segment. Figure 3.7 shows some example images with their respective CNN outputs and ground truth annotations.

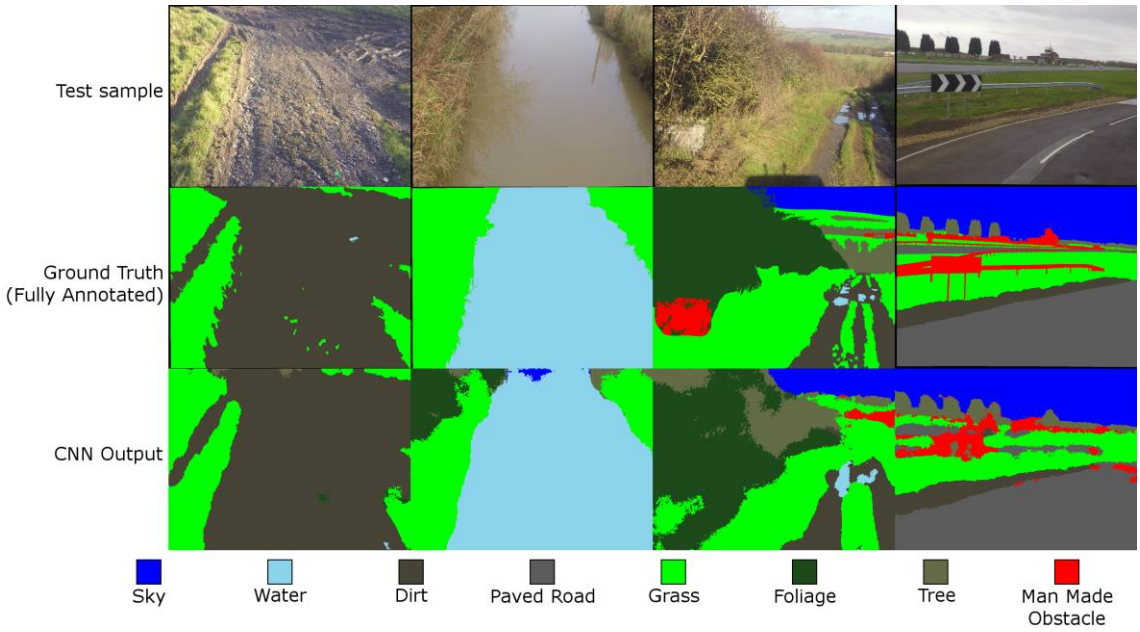

Figure 3.7 Results from the CNN after 30,000 iterations of pre-training and 10,000 iterations of training with the full off-road dataset. The middle row shows the fully annotated test images for comparison

Due to the lack of clearly defined boundaries in some areas of off-road scenes, there exist some pixels could have more than one correct label in terms of true ground truth. This should not have much effect on the partially labelled data, as boundary regions remain largely unlabelled, however this is likely to have a negative effect on classification results when testing against fully labelled data. To limit this effect, when deciding whether a pixel is correctly labelled we search for a match within a 5 pixel radius in the ground truth

image. When testing with partially labelled data, a pixel is only labelled correctly if a match is found at its exact location in the ground truth image.

When discussing the CNN, unless stated otherwise, accuracy is defined as the number of correctly labelled pixels divided by the total number of labelled pixels in the test data. When discussing the SVM, accuracy is defined as the number of correctly labelled segments divided by the total number of labelled segments in the test data.

3.3.1 CNN with Partially Labelled Test Data

First, we compare classification accuracy of the network model after training with different sized subsets of our off-road data set, and with the full data set after undergoing different amounts of pre-training with the Camvid data set. Results are obtained by testing the network model with our partially labelled test data set.

3.3.1.1 Pre-Training Iterations

Iterations Trained	1000	2000	5000	10000	20000	30000
Classification Accuracy	0.31	0.36	0.48	0.68	0.75	0.79

Table 3.1 Accuracy of the Segnet CNN on Camvid test data at the points when snapshots are taken to perform transfer learning

Table 3.1 shows the performance of the network on Camvid test data before any training with off-road data, with accuracy recorded at the six points from which snapshots were taken for subsequent fine-tuning. As the Camvid data is mostly fully labelled, we use the same measure of accuracy as we use with our fully labelled off-road test data set, wherein a label is deemed to be correct if it is within a 5 pixel radius of a similarly labelled pixel in the ground truth image.

These results demonstrate the network has rapid performance improvement over its first 10,000 training iterations, followed by a slower but consistent improvement in performance during later training iterations.

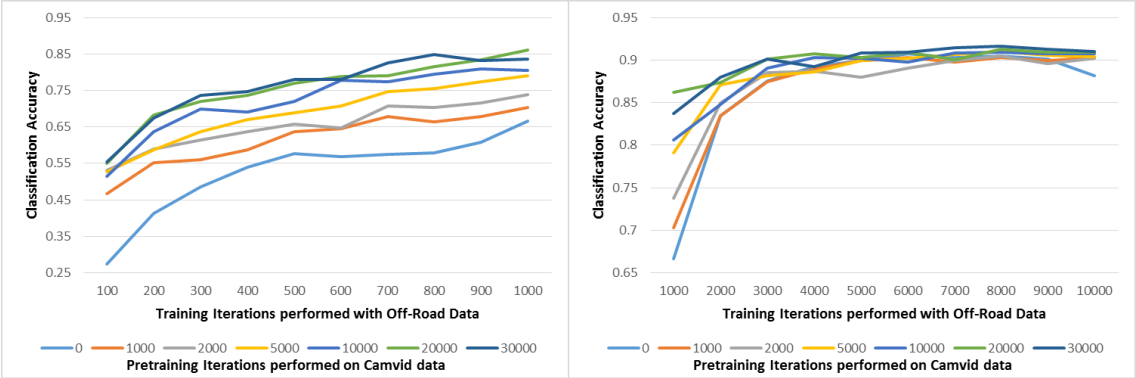

Figure 3.8 Comparison of training progress for networks that have undergone different amounts of pre-training.

Figure 3.8 shows the results achieved by each pre-trained version of the network on our full off-road data set. Each version of the network was trained for 10,000 iterations, with a snapshot taken and accuracy over our test data recorded first at every 100 iterations, then at every 1000 iterations.

The results show that the first few thousand iterations clearly benefit from transfer learning, with the networks that have performed a greater amount of pre-training generally performing better. However, by 5000 iterations of training, even the network initialised with random weights has achieved an accuracy of close to 0.9, beyond which there is very little improvement from any of the networks.

As the training continues, the networks pre-trained for longer give marginally better results. The highest accuracy achieved is 0.917, achieved by the network that pre-trained for 30,000 iterations with Camvid after it had undergone 8000 training iterations with the off-road data. The networks pre-trained for 20,000 and 30,000 iterations show very similar results throughout the training, suggesting a limit to the performance gains that can be achieved by pre-training.

Training was continued up to 20,000 iterations with each network, however this gave no further increase in accuracy and so only the first 10,000 iterations are shown.

It is interesting to note that our results surpass those achieved by their respective networks on the Camvid test data within a few hundred iterations, and then go on to perform significantly better. This could partly result from our data-set containing fewer classes (8 vs 11). Another factor could be our partially labelled test data, which features very few of the class boundary regions that form the most challenging parts in any semantic segmentation problem, however further testing with fully labelled data demonstrates a similar performance increase. It is possible that partially labelled training data could lead to a better performing classifier due to the lack of potentially confusing boundary pixels, although to fully test this we would need to compare these results to those obtained by training an identical network with a fully labelled version of the same data set, which is beyond the scope of this work.

3.3.1.2 Data Set Size

To consider the effect the amount of training data used has on classification, we train networks using five different sized subsets of our training data, containing 140, 70, 35, 17 and 8 images, both with and without pre-training. Figure 3.9 compares results for three of these subsets, each trained for 10,000 iterations.

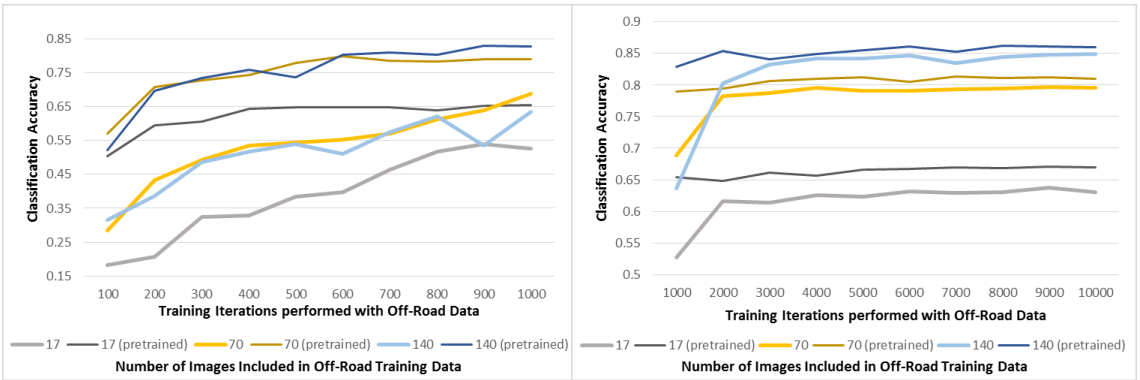

Figure 3.9 Comparing pre-trained and non-pre-trained networks using different sized subsets of our off-road data set.

The effects of transfer learning are similar to those observed when the complete data set is used: for the first 1000 iterations, the benefits of pre-training are clear. However, after just a few thousand more, both pre-trained and un-pre-trained networks have achieved

close to their optimum performance. As training progresses, the pre-trained networks consistently outperform their corresponding non-pre-trained network by a small margin, which generally increases as the dataset size decreases: After 10,000 iterations with a dataset of 140 images, the accuracy of the pre-trained network is just 0.01 better than the un-pre-trained network, while with the dataset of 8 images, this margin increases to 0.09. While pretraining has a large impact early in the training process, after 2000 iterations the effects of training data set size is significantly more pronounced.

3.3.1.3 Per Class Results

		Predicted							
		Sky	Water	Dirt	Paved	Grass	Foliage	Tree	Man Made
Actual	Sky	0.990	0.000	0.000	0.000	0.000	0.001	0.007	0.001
	Water	0.001	0.902	0.093	0.000	0.004	0.000	0.000	0.000
	Dirt	0.000	0.010	0.959	0.004	0.022	0.002	0.003	0.000
	Paved	0.000	0.001	0.028	0.962	0.003	0.000	0.000	0.006
	Grass	0.000	0.000	0.013	0.002	0.937	0.039	0.009	0.000
	Foliage	0.000	0.001	0.012	0.000	0.156	0.679	0.152	0.000
	Tree	0.000	0.000	0.001	0.000	0.025	0.022	0.948	0.004
	Man Made	0.002	0.007	0.020	0.090	0.031	0.031	0.206	0.613

Table 3.2 Confusion matrix of CNN output over our test data set. Values shown are labelled pixels as a proportion of total pixels belonging to ground truth class.

We now discuss in more detail the results from the CNN trained for 10,000 iterations on the full data set after 30,000 iterations of pre-training. This is the network configuration that we would expect to typically perform best, with the highest amount of pre-training and largest data set, and it consistently achieves an accuracy of 0.91 against our partially labelled test data once it has passed 5000 iterations.

Table 3.2 shows the confusion matrix of CNN output, with values represented as proportion of pixels with a given ground truth label. The most common misclassifications are between grass, foliage and trees, which is understandable given their visual similarities. Proportionally to class size, the largest is the 20.6% of pixels containing Man

Made Obstacles that are misclassified as Tree. This is likely because many of the man-made obstacles in the off-road environment, such as fences, posts and gates, are made of wood and so have a similar appearance to trees.

Figure 3.10 plots the precision and recall of each class along with the proportion of the training data set that each class makes up. The foliage class performed worst, likely due to its visual similarity to both grass and trees, while sky gave the best results. Camera exposure was set to capture maximum detail at ground level, so in most instances the sky is much brighter than the rest of the scene, which combined with its lack of high frequency detail and consistent placement at the top of an image makes it easily distinguishable from other classes.

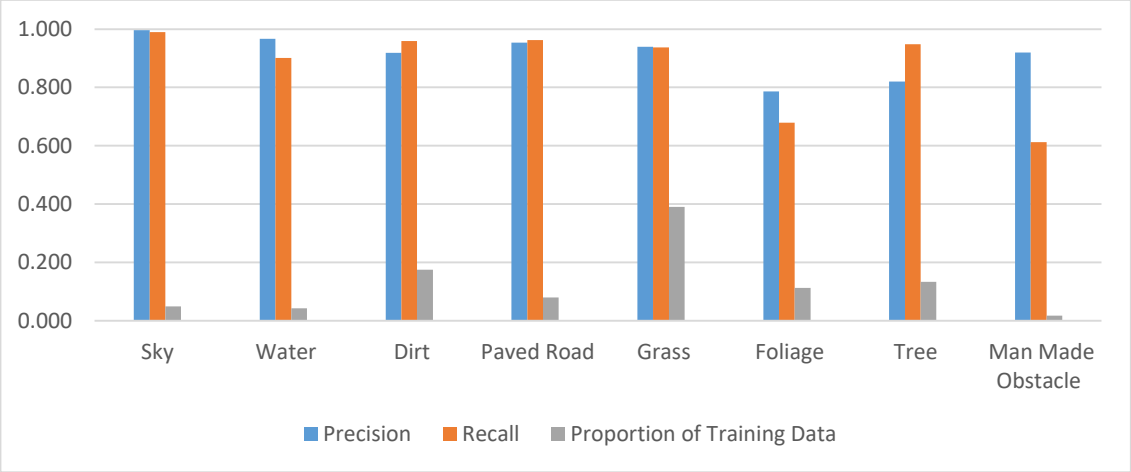

Figure 3.10 Per class statistics for the CNN classifier. As well as class precision and recall, we plot the number of pixels comprising each class within the training data as a proportion of the total number of labelled pixels in the set.

For the most part, classes that achieve high precision also achieve high recall, however man-made obstacle is an exception, with a very high precision (0.92) but lowest overall recall (0.613), meaning very few pixels are misclassified as man-made obstacle, while many pixels which should be labelled man-made obstacle are not. The fact that it is the class with fewest training samples (1.7% of the data set) is likely to have played a part in this, as well as its visual similarity to trees, as discussed above.

There would appear to be a stronger correlation between class frequency and recall than between frequency and precision, suggesting that the network model learns any biases inherent in the training data, despite our decision to weight pixelwise loss as inverse to class frequency during training.

3.3.2 Fully Labelled Test Images

So far we have only discussed the results obtained through testing the CNN classifier against partially labelled data, thus we also test it against a set of fully annotated images to demonstrate that it can achieve similar results when challenging class boundary regions are present.

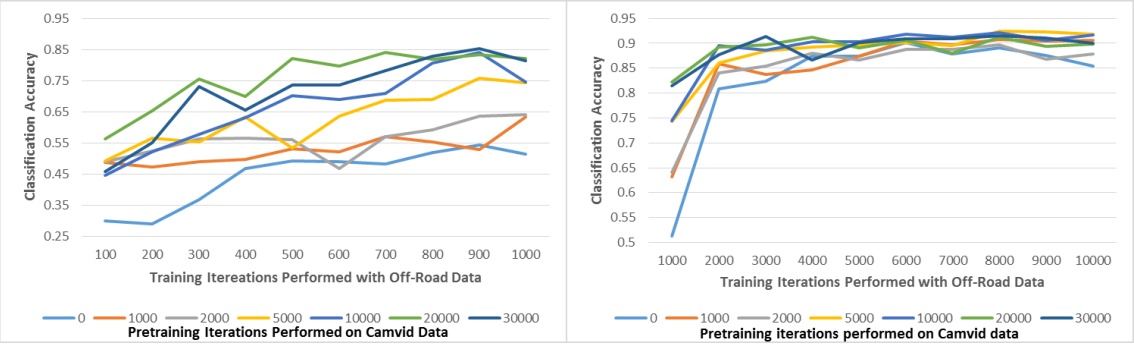

Figure 3.11 Classification results using the fully labelled test set, comparing networks that have undergone different amounts of pre-training on the Camvid urban data set.

Figure 3.11 shows the results obtained, demonstrating that testing with fully labelled images yields results very similar to those obtained using the partially labelled set. The highest accuracy was achieved by the network pre-trained for 5,000 iterations, with an accuracy of 0.924 after 8000 iterations of training with the full off-road data set.

Interestingly, the network snapshots that perform poorly on the partially labelled set (i.e. those that have not yet been through enough training iterations or have only been trained on a small data set) tend to perform worse on the fully labelled images. By contrast, those that perform well on the partially labelled data exhibit less deterioration, and in some cases even demonstrate an improvement in accuracy, when the fully labelled set is used.

This would appear to suggest that a more comprehensively trained network performs much better in class boundary regions.

Another point of note is that with the partially labelled data set, a network that had undergone greater pre-training would almost always perform better, however, when testing with the fully labelled data set, the networks that have undergone 5000 and 10,000 pre-training iterations consistently outperform those with 20,000 and 30,000 iterations, although only by a very small margin, at the later stages of training. This could be because the networks that have undergone more pre-training begin to overfit to the data they were originally trained on. The fact that this only occurs when the fully labelled data is used might suggest that this overfitting only has a noticeable effect when classifying class boundary regions, which are not present in the partially labelled data.

3.3.3 SVM

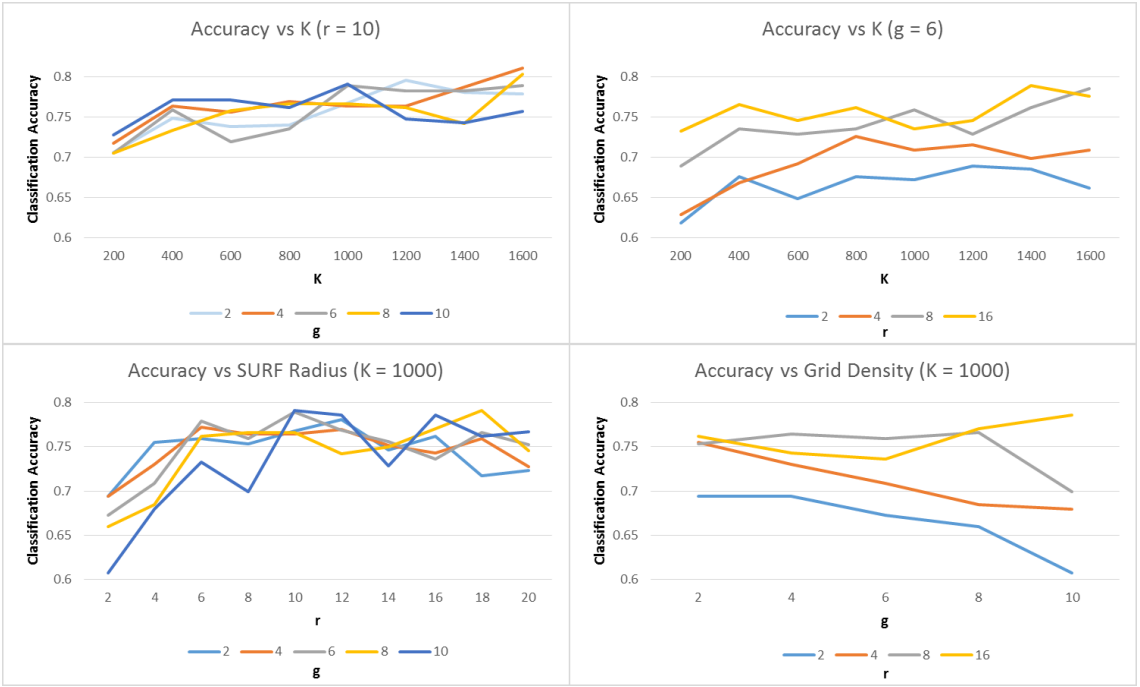

Figure 3.12 Results from SVM classifier using various feature configurations. K represents the number of clusters used for bag-of-words encoding, g is the density of feature grid, i.e. number of pixels in both the x and y direction between feature points, and r is the radius, in pixels, of the area that each feature point takes account of when building its SURF descriptor

For comparison, we test the SVM approach on its ability to classify segments from our off-road data set. The SVM parameters are automatically optimised through cross-

validation, however we test several different configurations for the features that we pass into the classifier. The parameters that we alter are *g*, the number of pixels between feature points in our grid; *r*, the radius in pixels around each feature point that our descriptors take account of; and *K*, the number of clusters used to build our bag-of-words representation. Figure 3.12 shows several comparisons to demonstrate how performance is affected.

We would expect a decrease in *g* to improve results, as a greater amount of detail is being considered. This partly holds true in our results, although not consistently so. As *r* changes, we initially see a consistent improvement in results, which begins to tail off after a while. This is likely because with *r* set too restrictively, each feature point only has access to a limited region of local gradient information. By contrast, with *r* set too large, high frequency detail is lost as the descriptor is built from a greater number of pixels. The optimum value appears to be around $r = 10$. The general trend for *K* is that larger is better, but memory and time constraints make too large a value impractical.

The best result attained by the SVM was an accuracy of 0.813, using the parameters $g = 6$, $r = 12$, $K = 1400$.

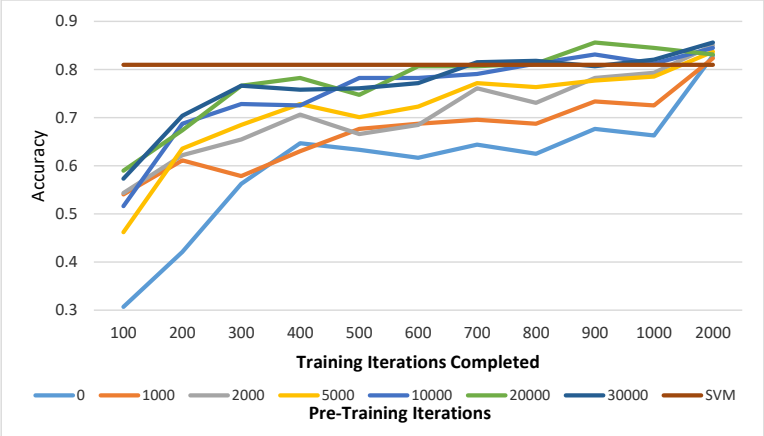

Figure 3.13 Training progress of the CNN after different amounts of pre-training, measured as accuracy on the segment classification task for comparison to the SVM classifier.

To properly compare SVM and CNN performance, we adapted our CNN classifier to label whole segments. This was done by winner-takes-all vote of pixel labels within the

segment. Figure 3.13 shows the segment classification results as the CNN is trained after different amounts of pre-training. The CNNs pre-trained for 10,000 or more iterations all achieve results better than those of the SVM before 1000 iterations have been completed, and by 2000 iterations all, including the CNN that has undergone no pre-training, have surpassed the SVM. After further training, segment classification results are very similar to those for pixel classification, peaking at around 0.91, confirming that the CNN is significantly more effective at this classification task than the SVM. Figure 3.14 shows examples of segments classified by the SURF/SVM approach.

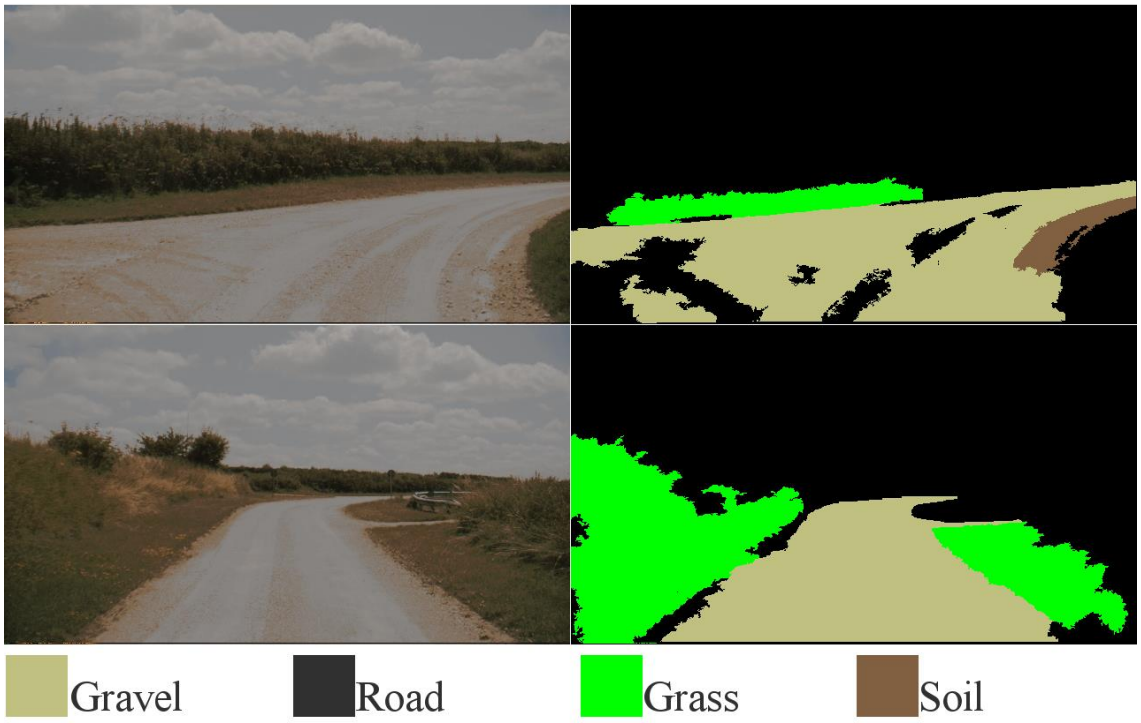

Figure 3.14 Example input and output of our SURF/SVM based approach. Images are segmented before classifier input, and in this case only those segments comprising the traversable surface area are shown with colour-coded class labels.

3.4 Summary

This work demonstrates how an existing deep convolutional neural network classification and segmentation architecture can be adapted to segment and classify off-road scenes. We have shown how quickly the network can learn to classify new kinds of images, and visualised CNN training performance by testing classification accuracy throughout the

cycle, allowing us to compare networks as training progresses to show the effects that transfer learning and data-set size can have on performance.

Notably, we have demonstrated that pre-training is of limited utility when a large data set is used for a long training period. While pre-training was shown to improve performance when smaller data sets were used, results were still well below those observed with larger training data, even without pre-training, suggesting that pre-training is no substitute for an adequately sized data set. Pre-training was also shown to improve results early in the training cycle, although as training continues these effects diminish until a network with no pre-training will almost match the performance of a pre-trained one. In our testing, this happened as early as 5000 iterations, which represents just 5 hours of training. However, our results have shown that networks that have undergone more pre-training tend to perform marginally better, even after many iterations of training, however the results obtained from our fully labelled test set appear to show the opposite effect above 5000 iterations of pre-training, suggesting that there is a limit. With that in mind, it would appear the optimum configuration of CNN for this task, using the Segnet architecture [1] and trained on our full off-road data set, is around 10,000 iterations of pre-training followed by 10,000 iterations of fine-tuning.

These results show that such a CNN can outperform an SVM based classifier using dense gradient features by a significant margin.

Chapter 4

Working Towards Improving Scene Understanding by Adding Depth Information

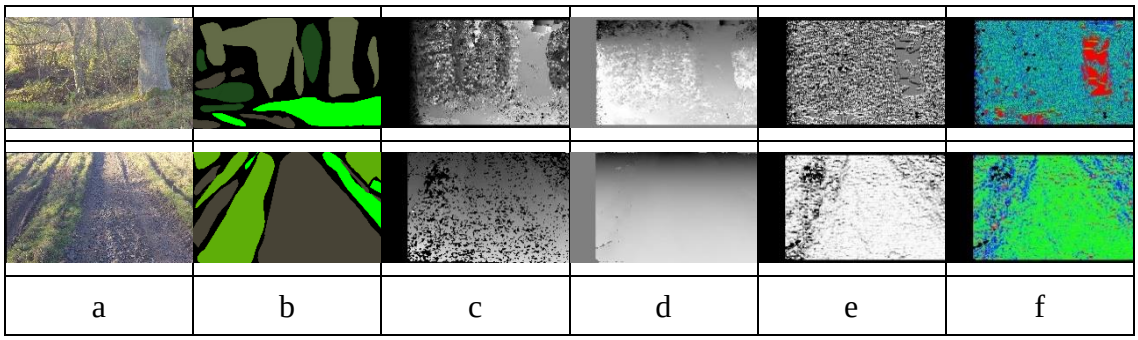

Figure 4.1 Example images from our off-road data set. a. RGB colour image from the left lens of our stereo camera. b. manually labelled ground truth. c. stereo disparity image. d. Height map encoding each pixel height above an assumed ground plane. e. Angle with gravity map, encoding the local surface orientation difference from an assumed gravity vector at each pixel. f. Normal map, encoding local surface orientation at each pixel as a 3 dimensional vector, with the absolute X value represented by the blue channel, Y green and Z red. Depth feature encodings in the top row are computed by ASW while the bottom row are computed by SGBM.

As discussed in Chapter 3, real-time scene understanding for autonomous vehicles is a challenging computer vision task, however recent advances in CNNs have facilitated performance that notably surpasses prior traditional feature driven approaches. In this chapter we discuss our attempts to further build upon these results by supplementing CNN input with depth information obtained from stereoscopic camera. We adapt the same Segnet architecture [1] to perform pixel-wise classification of off-road scenes from multi-channel images that combine RGB colour and depth information derived from stereo disparity. Using two state-of-the-art stereo matching algorithms, we train this CNN with raw disparity data as well as several transformations thereof, visualised in Figure 4.1, and compare results with those of a Segnet model trained on conventional RGB data to investigate the effect this depth information has on segmentation and classification performance. Over a limited set of challenging off-road scene imagery, we find the impact

of this additional depth information to be negligible, with performance roughly equal, or in some cases slightly diminished, to that demonstrated by the model relying on colour data alone.

4.1 Introduction

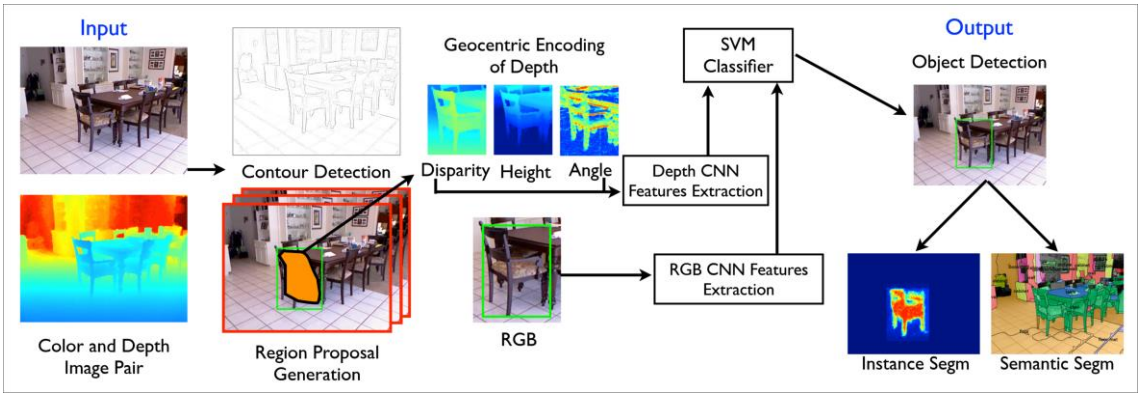

Figure 4.2 An overview of the object detection and segmentation approach of Gupta et al [63]. Colour and depth information are input to separate CNNs to extract relevant features which are then classified by SVM.

Although the majority of scene understanding work only considers conventional 2D colour images (RGB), it has been demonstrated that the addition of 3D depth information (*D*) can improve classification performance in semantic scene understanding tasks, such as in the work of Gupta et al [63]. Gupta's approach, visualised in Figure 4.2, takes a colour and depth image pair and extracts contours using a structured random forest approach to classify pixels (either as contour or not contour) based on local surface normal and pose information present in the depth data and edge information in the colour data. These contours are used to propose region where an object may be present, each of which is then used as input to separate colour and depth CNNs for extraction of features, which are subsequently classified by SVM.

With the increasing prevalence of automotive stereo [99], in this work we examine the relevance of depth within the context of scene understanding for autonomous off-road vehicles. Following our prior scene understanding work, we adapt the Segnet CNN architecture [1] to take either a 4 (RGB-*D*) or 6 (RGB-D_{f1}-D_{f2}-D_{f3}) channel image as input,

comprising an RGB colour image and stereoscopic disparity (normalized depth, D) or derivative features (D_{f1}, D_{f2}, D_{f3}) thereof (namely height above ground, local normal orientation and angle with gravity, following the work of [63]). We train this adapted network architecture to segment and classify off-road scenes over our own off-road stereoscopic data set, comparing the results with those obtained using standard RGB images.

4.2 Approach

Our method comprises a CNN that takes multiple-channel input images and outputs a single class labels for every pixel. Input images consist of 3-channel RGB information with one or more additional channels containing some 3D information derived from stereo disparity.

4.2.1 Segnet

The convolutional neural network architecture we use is nearly identical to that of Segnet [1], as described Chapter 3. We carry out minor alterations so that it can take images with more than 3 channels as input, and adapt the final layer to output the six classes found in our stereoscopic dataset.

For each of our input configurations, we initialise the network with randomised weights and train for 20,000 iterations, an amount empirically found to give the network ample time to achieve optimum performance. We do not perform pre-training in this case as a network with n input channels could only undergo pre-training with n-channel data, and we do not have a suitable larger dataset for this purpose.

We train out network model using stochastic gradient descent (backpropogation) with an initial learning rate of 1×10^{-3} and a momentum of 0.9, as these are the values the original authors used to benchmark the Segnet architecture.

4.2.2 Data

Figure 4.3 An example of a rectified image pair from our off-road data set.

We create our training data from a set of 276 rectified stereo colour image pairs from our own off-road data set, at a resolution of 480 x 360 with a stereo baseline of 400mm, within which we identify 6 pixel class labels, {*sky, water, dirt, grass, foliage, tree*}. An example stereo image pair can be seen in Figure 4.3. We partially label our images in the same manner as detailed in Chapter 3: a polygon is drawn that is entirely contained by, but not touching the edges of, each image segment, so that every pixel within the polygon is considered a member of the chosen class. Again, this is partly because full labelling, even with a small data set, can be very time consuming, but also because of the lack of clear boundaries delineating classes in off-road scenes, for example, when a muddy surface gradually gives way to gravel, or where long grass becomes foliage. Through this technique, roughly 69% of pixels in our data are labelled, with the remaining 31% marked as void so that the CNN ignores them during training. An example labelled image is shown in Figure 4.4.

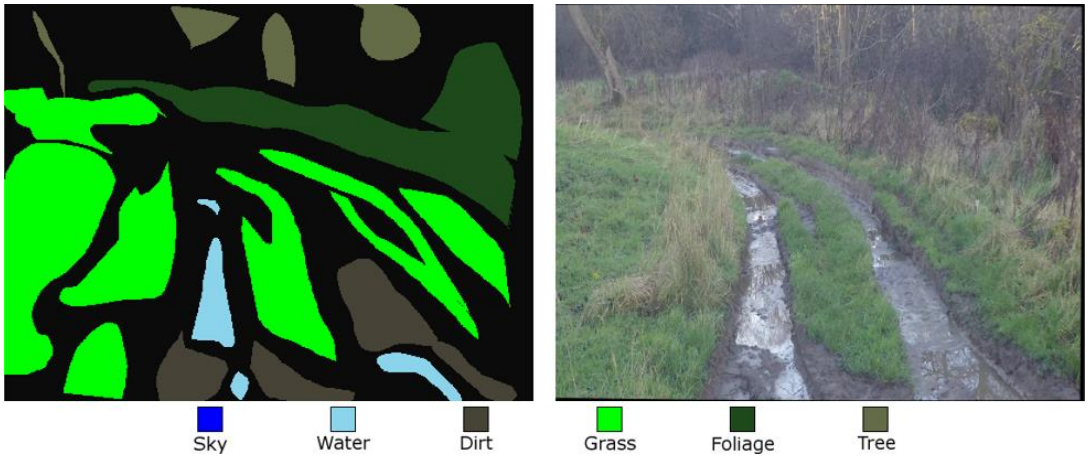

Figure 4.4 An example of a labeled image from our off-road data set.

From this data we create one training set containing just the RGB images from the left lens of our stereo camera, which will be used to set a benchmark. We then create a further 10 sets that each supplement the 3 colour channels with one or more channels containing depth information, the format of which is described in Section 4.2.3.

4.2.3 Depth Features

We compute two sets of stereo disparities, *D*, using Semi Global Block Matching (SGBM) [100] and Adaptive Support Weights (ASW) [101] methods, from which we create maps encoding height above the ground plane *H*, normal orientation *N* and angle with gravity A of each point. We then create training and test sets consisting of RGB-*D* (4-channel), RGB-*H* (4-channel), RGB-N_x-N_y-N_z (6-channel), RGB-*A* (4-channel) and RGB-*D*-*H*-*A* (6-channel) images using both stereo algorithms, giving us a total of 10 different colour-depth configurations for evaluation.

4.2.3.1 Stereo Disparity

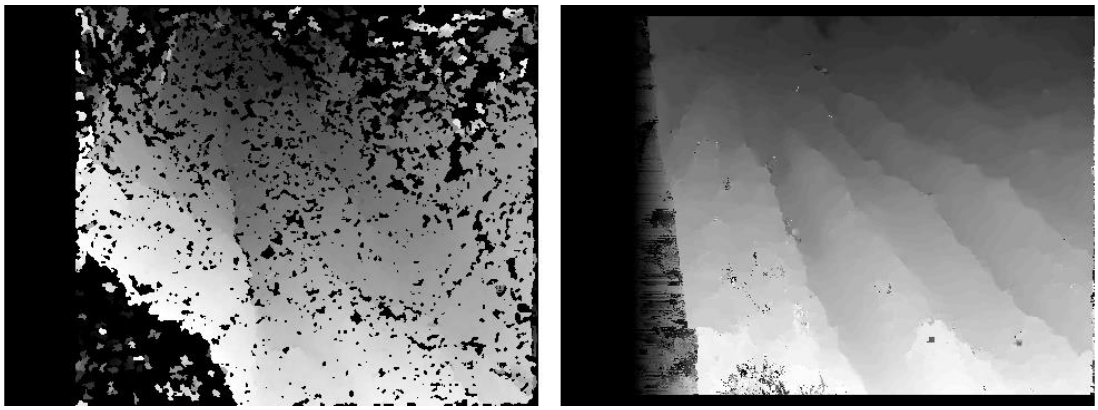

Figure 4.5 Example stereo disparity images created using SGBM (left) and ASW (right) approaches. Brighter pixels represent scene regions that are closer to the camera.

Stereo disparity measures the distance, in pixels, between the horizontal position of a feature in the left and right images of a (rectified) stereo image pair. In a typical stereo camera configuration, a point that is nearer to the camera will have a higher disparity. Our data set contains disparities between 1 and 64, which translates to a theoretical range from 3 metres up to 200 metres in front of the camera. There are a huge array of algorithms designed to compute stereo disparity [102] [103], however we only compare two in this work.

SGBM is widely used in real-time applications due to its combination of accuracy and speed. The algorithm first calculates the cost for every potential disparity match, in our case using the method of Birchfield and Tomasi [104], and then attempts to minimise a global smoothness constraint while aggregating these costs along a one-dimensional path across the image. To aid disparity retrieval in areas of low textural information we match blocks of 5 × 5 pixels, and to reduce noise we filter out any speckles that comprise fewer than 50 pixels. Example disparity images obtained from SGBM and ASW are shown in Figure 4.5, illustrating the differences between the two approaches and the impact that selection of disparity algorithm can have on obtained depth information.

There are many stereo disparity algorithms that have been shown to perform better than SGBM [105] [106], however these are mostly too slow for real-time automotive

applications. However, ASW is an approach that has been shown to be easily parallelisable [107], allowing for real-time performance when run on a GPU. Furthermore, ASW has been empirically shown to provide a greater level of depth texture granularity (i.e. depth detail) than SGBM due to the absence of a global smoothness prior which is central to the premise of SGBM techniques.

To compute ASW disparity, an $n \times n$ local support weight window is computed for each pixel in both the left and right images, in which each pixel is assigned a weight that takes account of its proximity, in both image coordinates and RGB colour space, to the central pixel. Equation(4.1 shows how these weights are calculated, where $w_{p,q}$ is the weight describing the similarity between the pixels at locations q and p, c is the RGB colour value at each pixel, and C and D are empirically chosen constants (here we use C=5, D=10.5 and n=21).

$$w_{p,q} = \exp\left(-\frac{\left|\overrightarrow{c_p - c_q}\right|}{C} - \frac{|\overrightarrow{p-q}|}{D}\right) \tag{4.1}$$

Each pixel p in the left image is then compared to each candidate pixel $\bar{p}$ in the set of potential matches P from the right image. Each pixel q in the support window W_p is compared to the corresponding pixel $\bar{q}$ in the support window $W_{\bar{p}}$, and the colour-space distance between their RGB values, c_q and $c_{\bar{q}}$, is weighted by both the support weight of q in regard to p and the support weight of $\bar{q}$ in regard to $\bar{p}$. These weighted distances are summed to compute the cost of matching p with $\bar{p}$ and a winner takes all approach is used to pick the match with the lowest score, as shown in Equation (4.2.

$$D_p = \underset{\bar{p}\in P}{\operatorname{argmin}}\left(\frac{\sum_{q\in W_p} w_{p,q} w_{\bar{p},\bar{q}} \left|\overrightarrow{c_q - c_{\bar{q}}}\right|}{\sum_{q\in W_p} w_{p,q} w_{\bar{p},\bar{q}}}\right) \tag{4.2}$$

The thinking behind this approach is that when seeking a match for a given pixel, it is more important to compare nearby pixels that are similar in colour to the pixel for which a match is being sought, as they are more likely to be a part of the same object or surface.

An example ASW disparity image is shown in Figure 4.5, adjacent to an SGBM disparity image obtained from the same stereo pair for comparison. The ASW image can be seen to demonstrate greater edge definition than the SGBM image, which relies on a global smoothness prior, while the lack of a confidence threshold means that every pixel is assigned a disparity value whereas many holes can be seen in the SGBM image.

In the case of both SGBM and ASW, we normalise disparity values normalised to the range 0 – 255 to create an 8-bit depth map which forms the *D* channel of our RGB-*D* images. The following depth representations are also derived from these disparity values.

4.2.3.2 Height Above the Ground Plane

From each disparity map we create height map *H*, encoding height above an assumed ground plane for each pixel. Information about vehicle orientation is not available in our data set, and due to the rough nature of off-road environments, fitting a ground plane, as in [63] which focuses on indoor environments, is not likely to be reliable, so we assume a fixed ground plane relative to the camera based on the known dimensions of the vehicle.

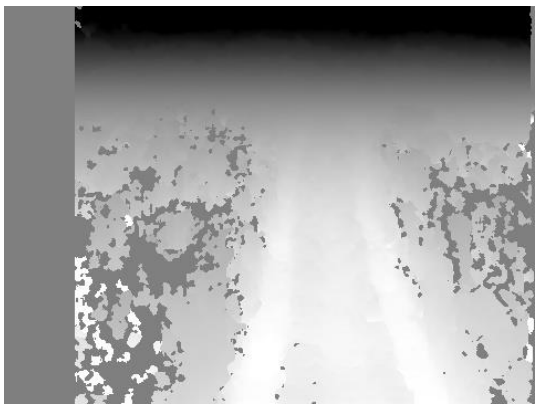

Figure 4.6 An example of a height map, H. Darker pixels are further from the assumed ground plane.

Each pixel height above this assumed ground plane is measured in millimetres, clamped between 0 and 2*h*, where *h* is the height of the camera above the ground, then normalised to 0 to 255 so that an 8-bit image can be produced. We calculate the height at each pixel using Equation (4.3, where *H* is the height map, *h* is the height of the camera above the ground in millimetres, *B* is the baseline of the stereo camera in millimetres, *D* is the

disparity image and *C* contains the pixel coordinates of the optical centre of the left stereo image. An example height map is shown in Figure 4.6.

$$H_{x,y} = \max\left(0, \min\left(2h, h - \frac{B}{D_{x,y}}(y - C_y)\right)\right) \tag{4.3}$$

4.2.3.3 Normal Orientation

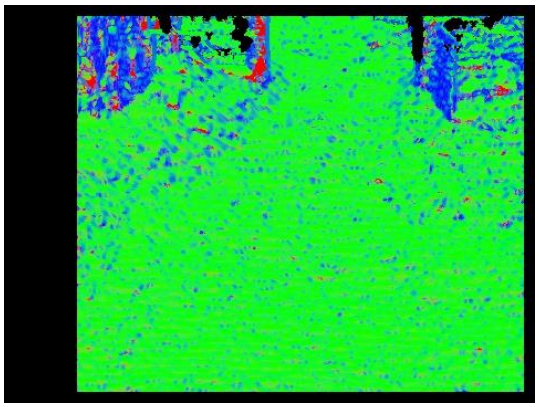

Figure 4.7 An example of a normal image, N. The blue channel represents the X value, green Y and red Z.

A normal map *N* encodes local orientation information at each point in an image. We compute one for each stereo disparity map, using 3 points from a 5 × 5 region around each pixel to give us a 3-dimensional vector describing the orientation of an imagined plane encompassing those three points. First, we compute the real-world coordinates of each point relative to the camera, *P*, as shown in Equation (4.4, where *B* is the stereoscopic baseline in millimetres, *f* is the left camera focal length in pixels, *D* is the disparity image and *C* contains the pixel coordinates of the optical centre of the left image.

$$P_{x,y} = \left(\frac{B}{D_{x,y}}(x - C_x), \frac{B}{D_{x,y}}(y - C_y), f\frac{B}{D_{x,y}}\right) \tag{4.4}$$

The normal, *N*, can thus be derived at a given point by calculating the cross product of two vectors that connect three surrounding points, as shown in Equation (4.5.

$$N_{x,y} = \left(P_{x-2,y+2} - P_{x,y-2}\right) \times \left(P_{x+2,y+2} - P_{x,y-2}\right) \tag{4.5}$$

We multiply the absolute value of each vector component by 255 to create an image comprising three 8-bit channels encoding the *X*, *Y* and *Z* values of the local normal. An

example normal map is shown in Figure 4.7, where the red channel corresponds to the Z value, green to Y, and blue to X (i.e. the ground mostly appears green, the surfaces of obstacles that face towards the camera appear red, and vertical surfaces that line the sides of the vehicle path appear blue).

4.2.3.4 Angle with Gravity

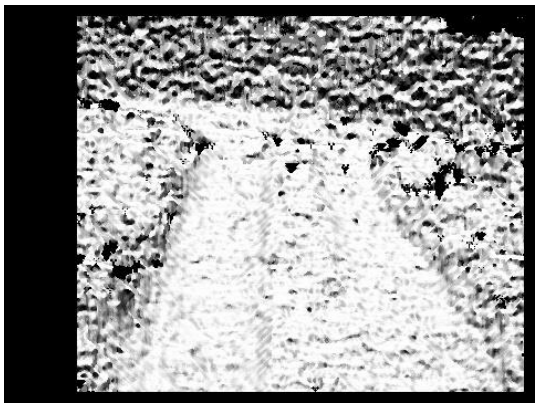

Figure 4.8 An example of an angle-with-gravity image, A.

In [63], angle with gravity was found to be a useful CNN input feature when classifying objects in RGB-D images of indoor scenes. We derive such a map, A, from each of our normal orientation maps, N, by computing the dot product of each normal vector with a gravity vector to give us the cosine of the angle between the two, as shown in Equation(4.6. Due to a lack of either vehicle orientation information or a reliable ground plane in the off-road environment, we assume a gravity vector of (0, -1, 0) relative to the camera. Each value is then multiplied by 255 to create an 8-bit image. An example of such an image can be seen in Figure 4.8.

$$A_{x,y} = N_{x,y} \cdot (0, -1, 0) \tag{4.6}$$

4.3 Results

We assess the trained network models on a test data set consisting of 29 images labelled in the same manner as our training data. As each model trains, we take a snapshot after every 1000 iterations which we assess using this test data so that we can analyse how

classification performance improves as training progresses. Our primary metric for assessing classification performance is accuracy, defined as the number of correctly labelled pixels divided by the total number of labelled pixels in the test data.

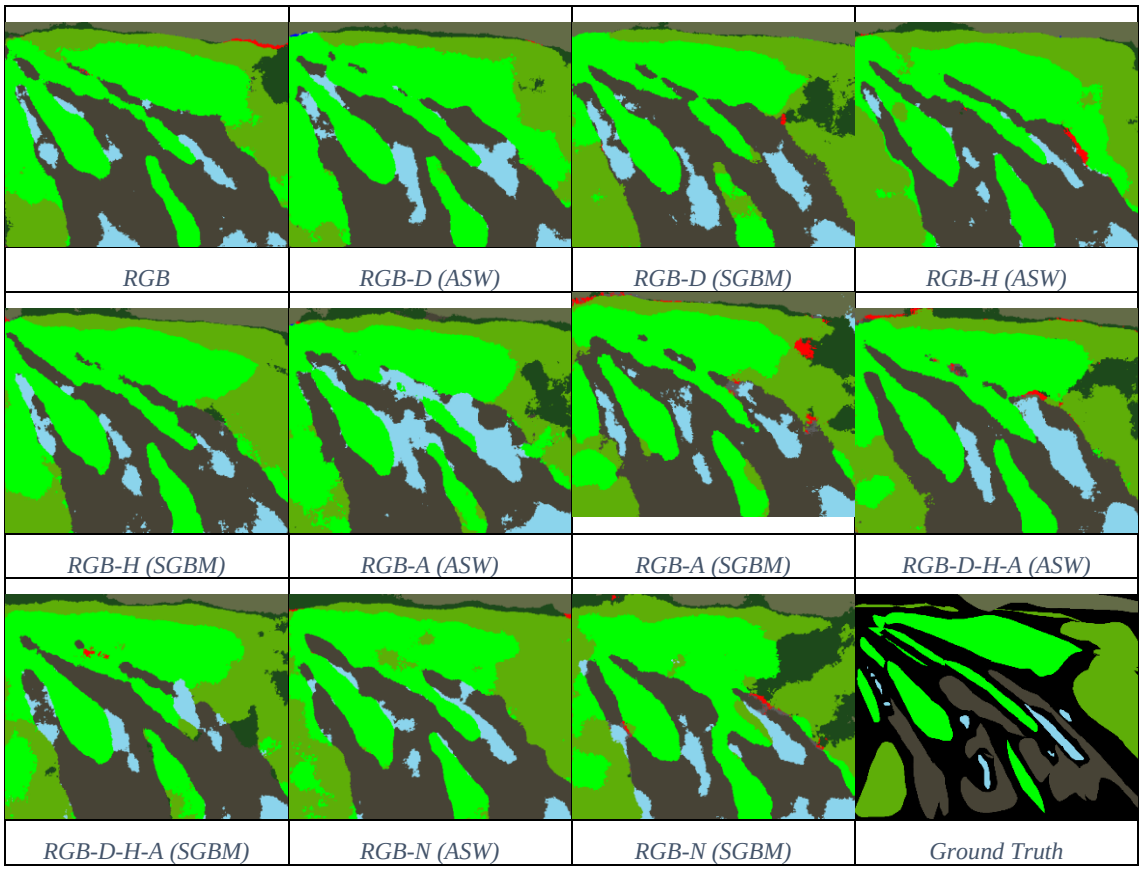

Figure 4.9 Example output for the same image from each of our models after 20,000 training iterations. The differences between model outputs appear very small, a hypothesis supported by our quantitative evaluation.

Table 4.1 shows the best test accuracy observed with each of our data sets, demonstrating very similar results in all cases. RGB-*H* data derived from SGBM disparity is the only configuration to outperform RGB colour data, albeit by a very small margin, suggesting that overall depth data is of limited utility in addressing this particular scene classification problem. With all data sets apart from RGB-*A*, classification performance was better with SGBM-derived depth features than with those derived from ASW, suggesting that disparity smoothness is more important than boundary definition. Figure 4.9 shows the output from each of our trained models for a single input image from our test data set, demonstrating how similarly all the models perform.

Data	RGB	RGB-D		RGB-H		RGB-A		RGB-D-H-A		RGB-N	
Disparity	-	ASW	SGBM	ASW	SGBM	ASW	SGBM	ASW	SGBM	ASW	SGBM
Accuracy	0.87	0.86	0.87	0.84	**0.88**	0.87	0.84	0.84	0.84	0.82	0.83

Table 4.1 Best overall class accuracy observed from each trained network model over our off-road data set.

4.3.1 RGB

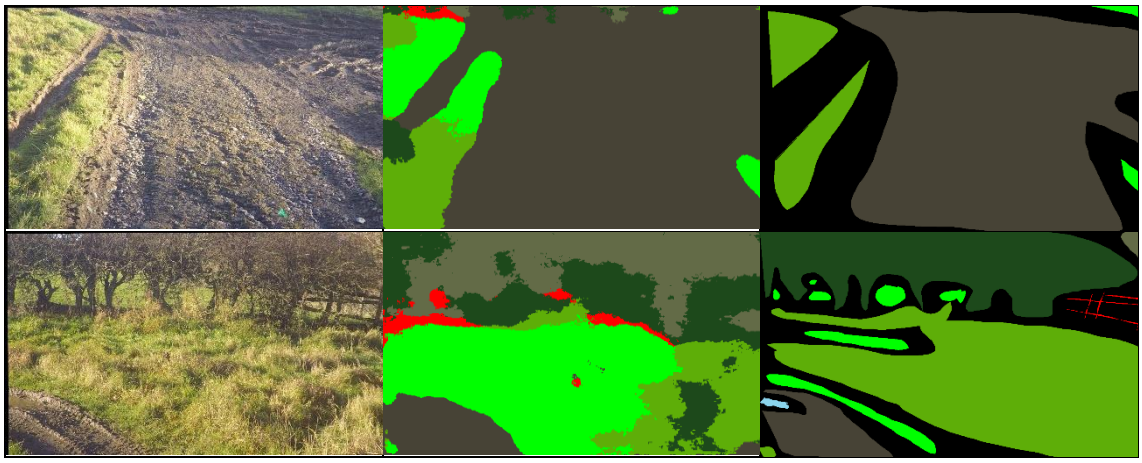

Figure 4.10 Example output from the CNN model using RGB input. Left - input image, middle - output, right - ground truth.

Figure 4.10 shows example RGB data along with corresponding CNN output and ground truth. Figure 4.11 plots test accuracy of the network model trained and tested on RGB images containing no depth information, which we use as a baseline for comparison with our other configurations. The peak accuracy achieved in this case is 0.87, which comes after just 6000 iterations of training, however it then settles at around 0.86 for the remainder of the training, suggesting that this result is an anomaly. This result is slightly worse than that observed in our earlier work described in Chapter 3, in which an accuracy of 0.91 was achieved, however that network model had undergone pretraining with an urban road scene data set and a slightly larger off-road data set was used in that case.

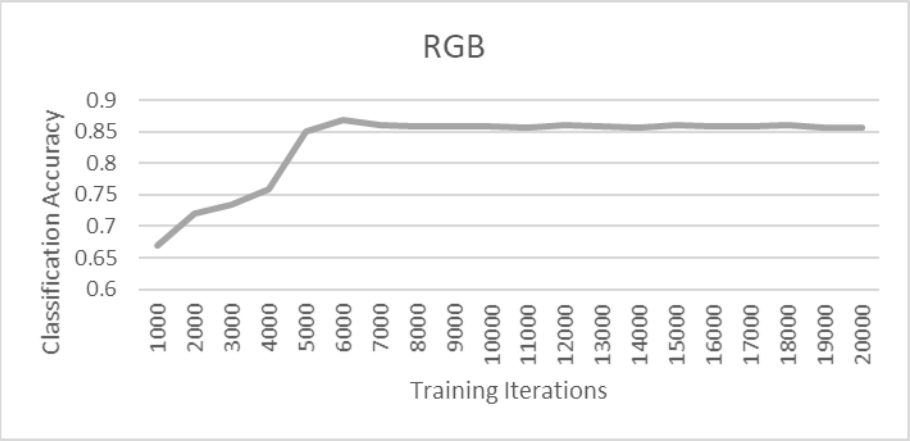

Figure 4.11 Training progress of the CNN using RGB images. Peak performance was reached after about 5000 training iterations.

4.3.2 RGB-D

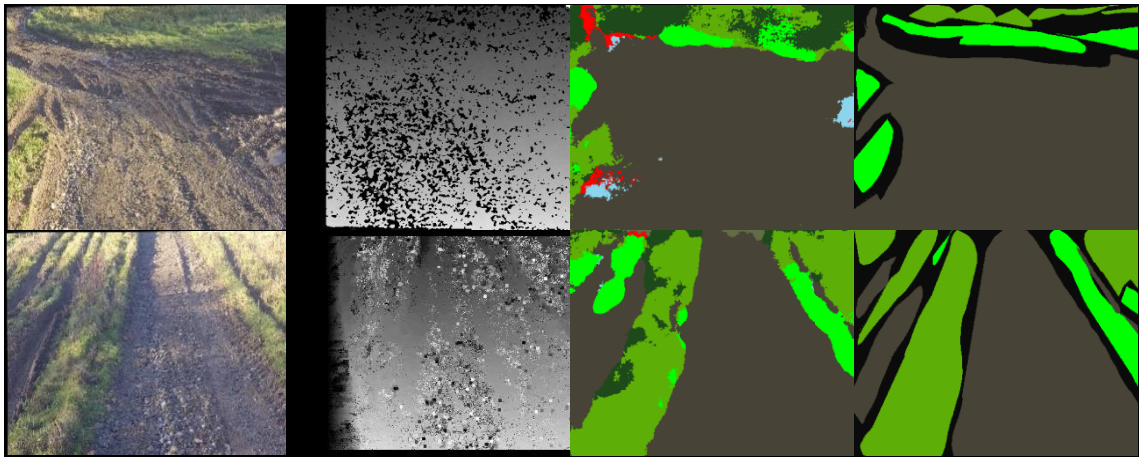

Figure 4.12 Example output from the CNN model using RGB-D input. Left to right - input RGB image, depth image, output, ground truth. Top row uses SGBM disparity, bottom row uses ASW disparity.

Figure 4.12 shows example RGB-*D* data along with corresponding CNN output and ground truth. Figure 4.13 shows the results obtained from the network model trained to classify RGB-D images, in which the three colour channels (*R*, *G* and *B*) are supplemented by an additional channel encoding stereo disparity computed with either the ASW or SGBM algorithm (*D*). The extra information contained in the RGB-*D* images does not appear to make much difference, with both networks achieving performance very similar to that of the model trained with RGB images. Neither model achieves a result better than the 0.87 reached by RGB after 6000 iterations, however for the remainder of training the ASW disparity does marginally outperform RGB, while SGBM is marginally worse.

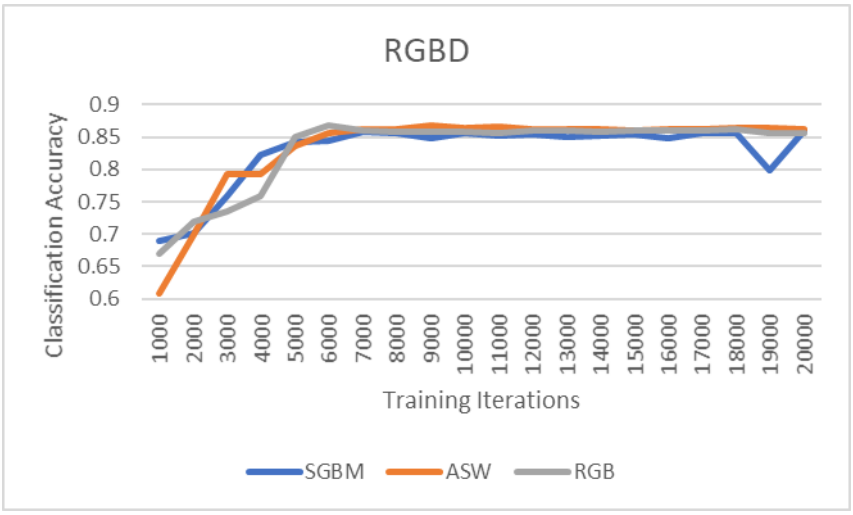

Figure 4.13 Training progress of the network models using RGB-D images, with disparity computed by both SGBM and ASW methods. RGB results are also shown for comparison.

4.3.3 RGB-H

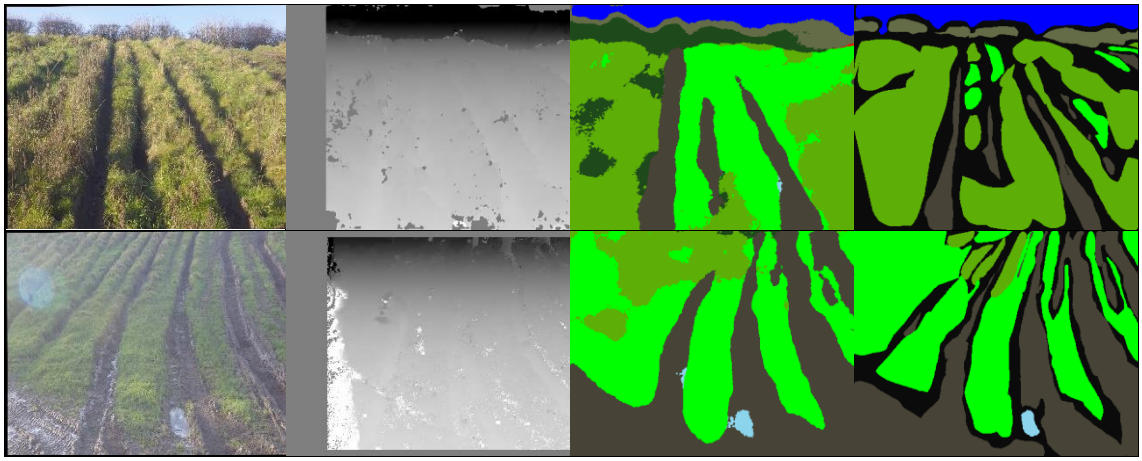

Figure 4.14 Example output from the CNN model using RGB-H input. Left to right - input RGB image, height image, output, ground truth. Top row uses SGBM disparity, bottom row uses ASW disparity.

Figure 4.14 shows example RGB-*H* data along with corresponding CNN output and ground truth. Figure 4.15 shows the classification accuracy when using RGB-*H* images, where the fourth channel encodes each pixel height above an assumed ground plane, derived from either SGBM or ASW disparity. The effect of the addition of height data on performance is more pronounced than in the case of RGB-*D*, although still small. SGBM derived height appears to outperform ASW derived height, which would not perhaps be expected given the relative depth texture characteristics of the two algorithms. After 14,000 iterations, the RGB-*H* imaged derived from SGBM disparity achieves a classification accuracy of 0.88, the highest reached by any configuration during these

experiments, and goes on to consistently outperform the RGB CNN by 0.02-0.03, while ASW performs 0.02-0.03 worse than RGB.

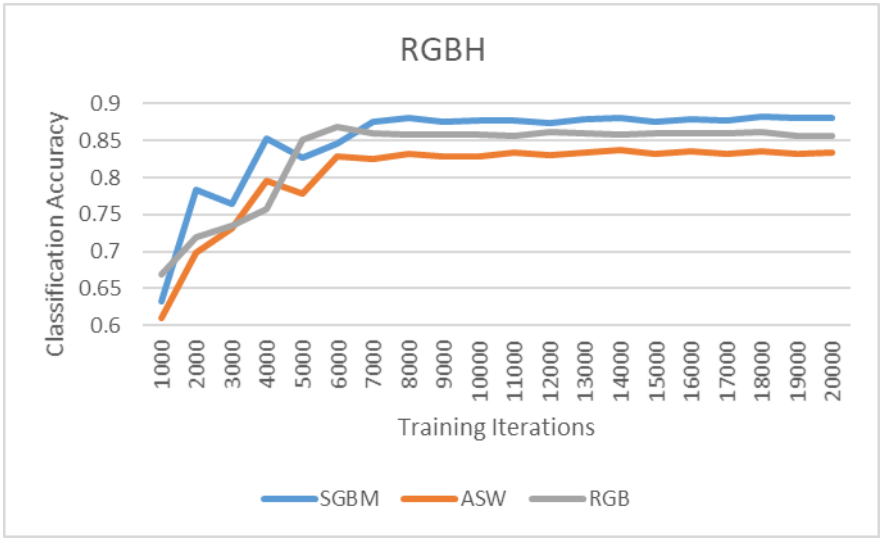

Figure 4.15 Training progress of the network models using RGB-H images, with height derived from both SGBM and ASW algorithms. RGB results are also shown for comparison.

4.3.4 RGB-N

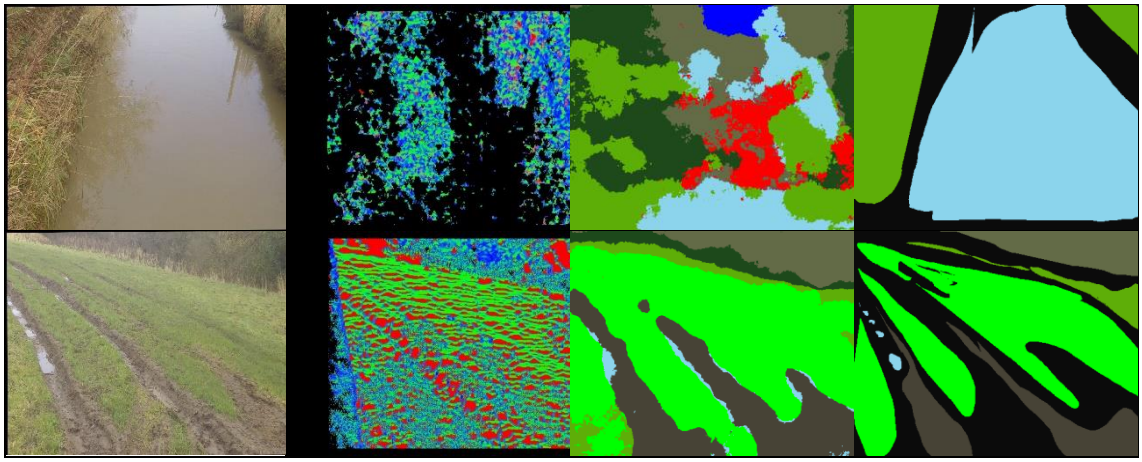

Figure 4.16 Example output from the CNN model using RGB-N input. Left to right - input RGB image, normal image (blue = X, green = Y, red = Z), output, ground truth. Top row uses SGBM disparity, bottom row uses ASW disparity.

Figure 4.16 shows example RGB-*N* data along with corresponding CNN output and ground truth. Results from RGB-N_x-N_y-N_z, that is 3 colour channels and 3 channels encoding the local normal orientation at each point, are shown in Figure 4.17. In this experiment, the network trained using images containing normal data derived from SGBM disparity consistently demonstrates an accuracy in the range of 0.03-0.04 below that of RGB, while ASW performs 0.04-0.05 worse than RGB.

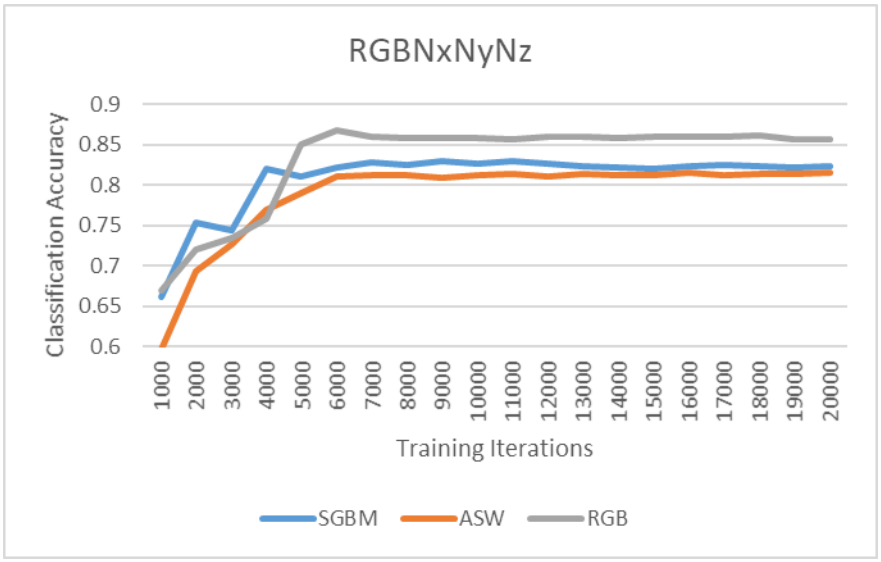

Figure 4.17 Training progress of the network models using RGB-N_x-N_y-N_z images, with normal vectors derived from both SGBM and ASW algorithms. RGB results are also shown for comparison.

4.3.5 RGB-A

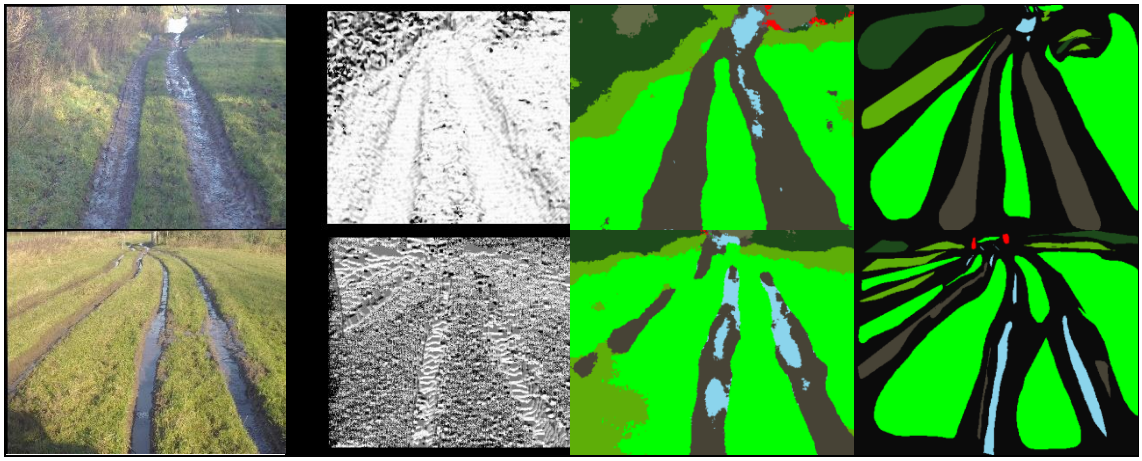

Figure 4.18 Example output from the CNN model using RGB-A input. Left to right - input RGB image, angle with gravity image, output, ground truth. Top row uses SGBM disparity, bottom row uses ASW disparity.

Figure 4.18 shows example RGB-*A* data along with corresponding CNN output and ground truth. Figure 4.19 plots results obtained using RGBA images, where *A* encodes the cosine of the angle between gravity and local normal at each point. The images derived from ASW perform best in this case, although with only marginal gains compared to the model trained with RGB images. SGBM consistently achieves an accuracy between 0.02 and 0.03 below that of RGB in this experiment.

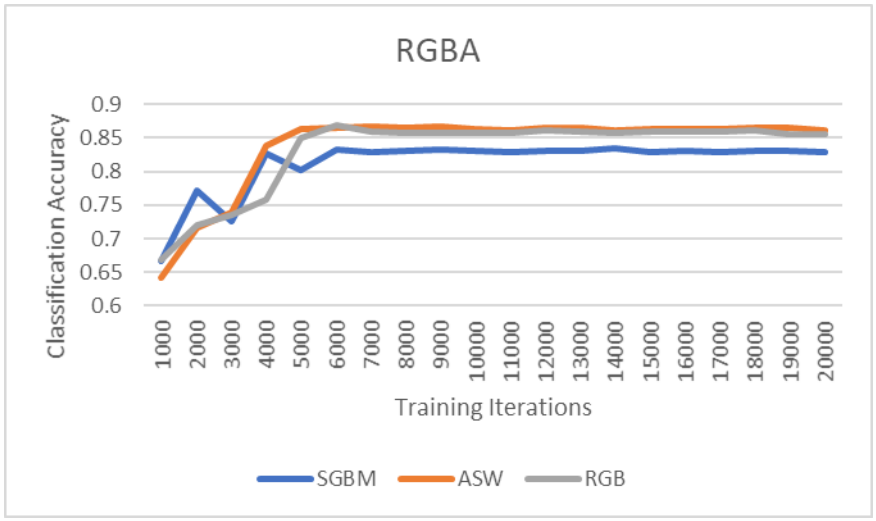

Figure 4.19 Training progress of the network models using RGB-A images, with angle with gravity derived from both SGBM and ASW algorithms. RGB results are also shown for comparison.

4.3.6 RGB-D-H-A

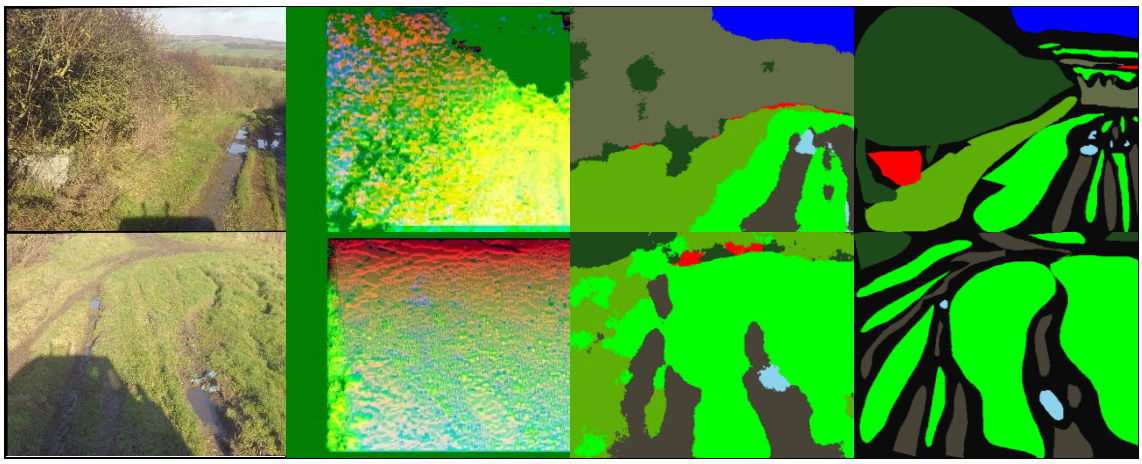

Figure 4.20 Example output from the CNN model using RGB-D-H-A input. Left to right - input RGB image, D-H-A image (blue = D, green = H, red = A), output, ground truth. Top row uses SGBM disparity, bottom row uses ASW disparity.

Gupta et al. [63] showed promising results from combining disparity, height and angle with gravity into 3-channel images for object detection in indoor environments using a CNN, so we combine our 3 similar channels with our 3 RGB channels into a set of RGB-*D*-*H*-*A* images, results from which are shown in Figure 4.21, and example input and output is shown in Figure 4.20. The *D*, *H* and *A* channels obtained from SGBM perform best here, seeing gains of around 0.01 over RGB, while ASW consistently performed between 0.02 and 0.03 below RGB.

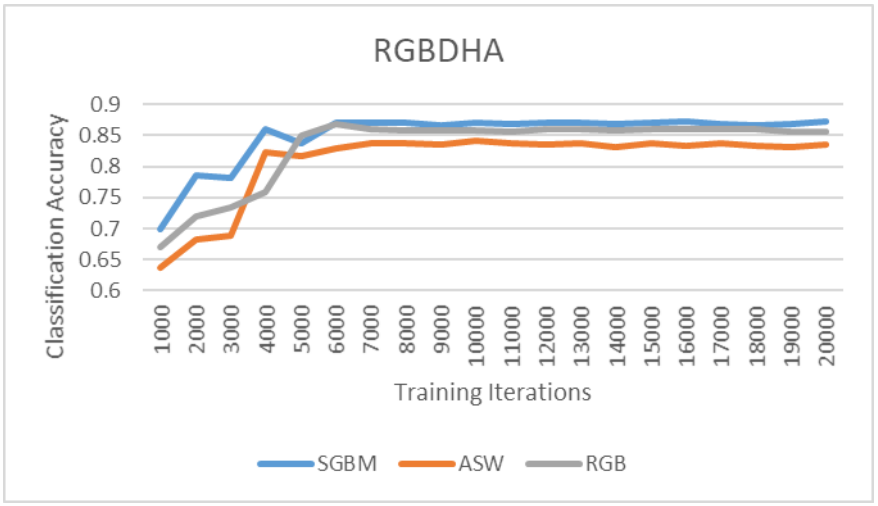

Figure 4.21 Training progress of the network models using RGB-D-H-A images, derived from both SGBM and ASW disparity. RGB results are also shown for comparison.

4.4 Summary

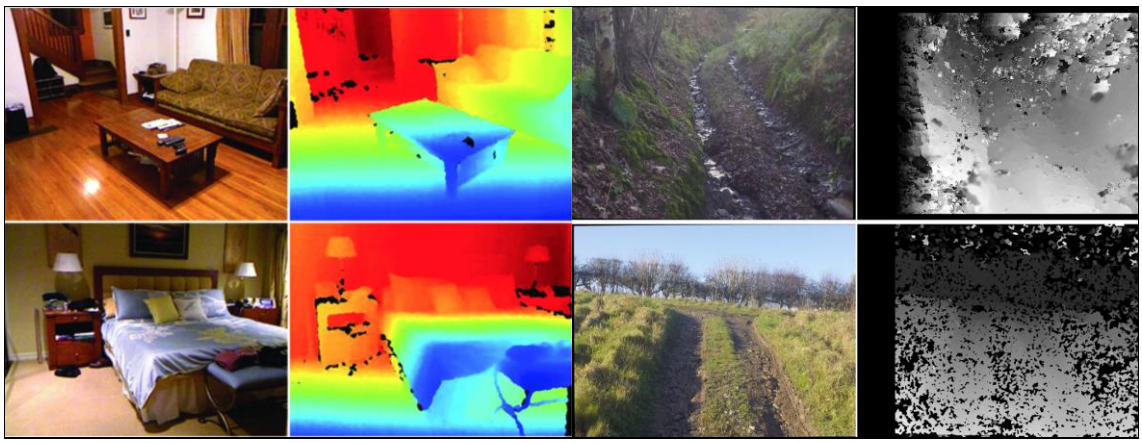

Figure 4.22 Example colour and depth images from the NYU Depth data set [88](left) and from our off-road data set (right).

Our results would appear to show that the additional information contained in stereo disparity can only provide marginal improvement to CNN classification accuracy over standard RGB images in this particularly challenging problem of off-road scene understanding. This is partly due to the efficacy of the Segnet architecture at classification from conventional colour images (RGB), with results so good to begin with that it becomes ever harder to gain additional performance. Another potential consideration is the quality of disparity information used, as we focus our attention on real-time applications and so more accurate, but slower, stereo disparity algorithms were not investigated. Our results would appear to contradict those of prior work, such as [63],

which demonstrated that depth information can improve classification performance in indoor environments. However, the scene structure inherent in such environments does not exist in our data set, and it may be the case that depth data does not contain meaningful information able to aid classification of the type of cluttered, natural environments where off-road driving takes place. Figure 4.22 illustrates this with example depth images from our data set as well as from the NYU Depth data set [88]: the difference in quality of the depth information obtained by structured light depth sensor in indoor environments compared to stereoscopic camera and realtime disparity algorithm in unstructured environments is clear, as is the difference in scene structure. The different classes present in the NYU data, such as table, bed, and sofa, can be clearly visually distinguished in the depth images, however the classes present in our data, such as grass, bushes, and dirt, are not visually obvious within our depth images. We believe this is a major reason why this work does not appear to support the conclusion of Gupta et al [63] that depth information improves classification performance.

We also show that performance gains through transforming disparity information before inputting it into a CNN are either non-existent or very small. Further consideration of what features to extract from disparity could potentially lead to further improvements. However, such features are usually derived solely from the disparity data itself, and thus contain no new information. As such, they should not aid CNN classification as in theory a neural network should optimise during training to automatically find the most effective transformation of its input data. For example, if local orientation information were the most effective transformation of raw disparity, a CNN fed disparity images should adjust its parameters during training such that they approximate the calculation used to derive orientation from disparity.

Chapter 5

Off Road Path Prediction

In this chapter we discuss the problem of identifying a viable vehicle path through an off-road environment. This is an especially challenging problem due to the unstructured nature of off-road environments, along with visually indistinct obstacles and the need to assess terrain parameters that might affect levels of grip or cause a vehicle to become immobile.

While previous chapters have focused on semantic labelling of off-road scene features, this chapter investigates techniques for planning the route a vehicle should take through such a scene. First, we reimplement and evaluate an existing end-to-end learning approach [2] that has demonstrated state of the art on-road results. We evaluate the performance of this method over our own off-road dataset, identifying several limitations. We then attempt to address these limitations with our own proposed visual path prediction approach, in which a CNN is trained to segment an image from a forward-facing camera into path and non-path areas.

Figure 5.1 A frame from a video demonstration of the Nvidia end to end autonomous driving approach [108].

5.1 End to End Learning

We reimplement and evaluate the End to End Learning approach of Bojarski et al [2] in an off-road environment. A CNN is trained to predict steering inputs using data comprising images from a forward-facing camera and corresponding ground truth vehicle control inputs made by a human driver. Figure 5.1 shows a frame from a video of the approach being demonstrated [108] on a road vehicle driving through a test course, in this case navigating a scenario set up to look like road works.

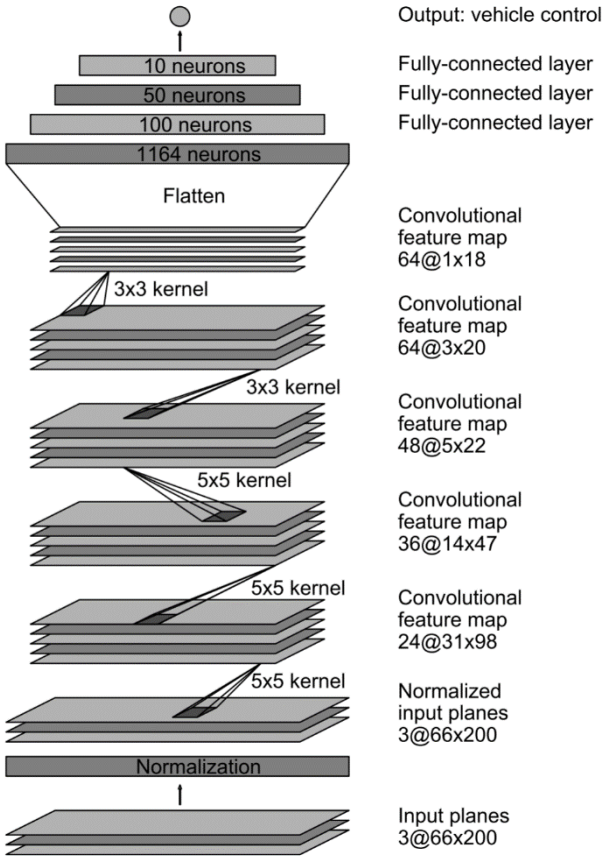

Figure 5.2 The CNN architecture proposed by Bojarski et al [2] for end to end learning.

5.1.1 Implementation

Figure 5.2 shows the End to End Learning network architecture as implemented by the original authors, comprising five convolution layers, three of which use a 5 × 5 kernel, the final two of which use a 3 × 3 kernel, followed by three fully-connected layers and a regression output. We use as input a single colour image of dimensions 256 × 136 and output is a single floating-point value {-1→1} that represents normalized steering wheel

angle (i.e. the amount by which the vehicle steering wheel has been turned as a proportion of the maximum rotation that can be applied in either direction). Some examples from our data set are shown in Figure 5.3. The original authors used a significantly smaller input size of 200 × 66 to enable real-time performance on a mobile device, however we found that too much useful information was lost when performing such a drastic downsampling of our off-road dataset.

We train this network for 100,000 iterations using ~20,000 images taken from our off-road data, each with an associated ground-truth normalised steering wheel angle, using a batch size of 6 images. Loss is computed as the mean squared error between predicted and ground-truth steering wheel angle across each batch, and backpropagated by stochastic gradient descent with momentum = 0.9. The initial learning rate is set to 2×10^{-4} and a step function is used to decrease this as training progresses.

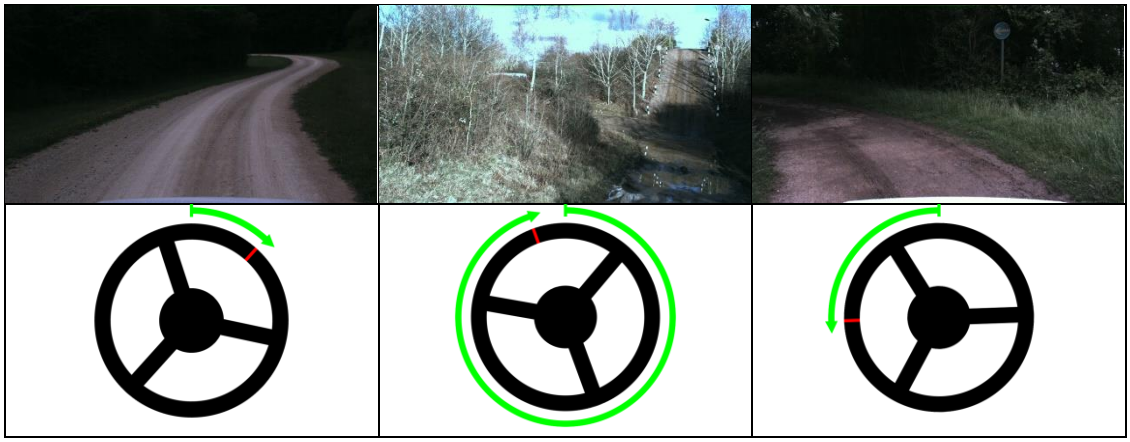

Figure 5.3 Example images from our off-road data set along with a visualisation of corresponding ground truth steering angle.

5.1.2 Evaluation

The original authors of the End to End Learning method we are implementing did not conduct a quantitative evaluation of their approach, instead relying on the subjective measurement of the number of times a human passenger felt it necessary to override the driving decisions of the system [2]. We address this by performing a quantitative analysis in which we evaluate performance through measurement of the mean error between

predicted and ground truth steering wheel angle. Over our test dataset, mean error between predicted and ground truth normalized steering wheel angle was 0.097. Unnormalised, this translates to an average error of 52.6° at the steering wheel, which, given the steering ratio of the vehicle, translates to approximately 3.5° of steering direction error at the road wheels.

Figure 5.4 plots predicted steering angle against ground truth for our test data set, demonstrating a clear bias towards straight-ahead within the data which appears to be amplified in the predictions - despite samples existing right up to the full steering extents, no prediction demonstrated a magnitude of greater than 0.5. This suggests that the network optimised to a local minimum whereby it could minimize error by constraining its predictions close to the mean. This could potentially be addressed by modifying the training data so that a greater proportion of samples demonstrate sharp turns, or by modifying the training loss function so that loss is weighted proportionally to the distance a given sample lies from the mean. However, this does demonstrate an additional limitation of this approach that is especially significant in an off-road environment and that we aim to address with our own approach.

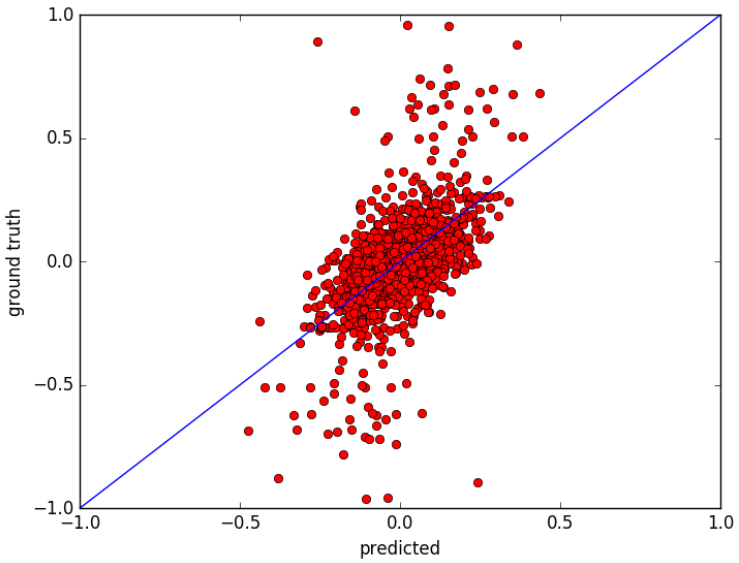

Figure 5.4 Steering angle values output by the End to End CNN for our test data plotted against ground truth.

5.2 Visual Path Prediction

In the End to End Learning approach discussed above [2], a neural network is fed an image from a vehicle mounted camera and trained to predict the steering input a human driver would make at the time the image was captured. We have identified four major limitations with this method: a) only immediate driving inputs are considered, with no thought as to how the vehicle path might change over time; b) driving inputs are learned for the characteristics of a specific vehicle, and so to apply the technique to a new vehicle with different steering characteristics, for example a tracked vehicle, would require the system be retrained on data from that vehicle; c) steering inputs do not necessarily relate consistently to the movement of the vehicle, especially in off-road environments where effective traction may be severely limited; d) the strong bias towards straight-ahead likely to be present in any real-world training data appears to become amplified in network output.

In this work, we address these limitations by proposing a visual end-to-end autonomous driving approach, whereby a CNN is trained to map future vehicle path directly to pixels in an image from a forward-facing camera. Training data is created automatically by using stereoscopic visual odometry [109] to track the motion of a human-driven vehicle through a sequence of images and then mapping this motion into the image space of the initial frame such that pixels that the vehicle traverses are labelled as '*path*'. This addresses the identified limitations of existing end-to-end autonomous driving approaches [2] [19] [39], whereby only immediate driver input is predicted, by predicting a path that takes account of future changes in direction and does not rely on a direct link to driver inputs. Furthermore, the output of this process could be combined with semantic scene understanding, as discussed in Chapters 3 and 4, to semantically label path pixels and create awareness of upcoming terrain characteristics that can be used to setup vehicle

parameters, such as suspension stiffness and gear ratio, for optimum traction and passenger comfort.

Subsequently, we use our automatically labelled data to train three state-of-the-art CNN architectures, each originally designed to perform a different segmentation or classification task. We also create a test dataset in the same manner, which we use to carry out a quantitative analysis of the performance of our approach using the three architectures.

5.2.1 Approach

The problem we are solving is the prediction of the path that a human driver would take through an off-road environment from a single image of that environment taken by a forward-facing vehicle-mounted camera. Our approach involves the automatic creation of training-data, whereby labelling of image pixels is automated using visual odometry to track vehicle motion, then using this data to train a CNN to map future vehicle path to image pixels, and by extension real-world location.

5.2.1.1 Automated Dataset Creation

Figure 5.5 An off-road vehicle being driven as part of our data collection process. A stereoscopic camera rig attached to the roof of the vehicle was used to capture data in a wide variety of off-road environments.

Our training data comprises individual colour images captured by a forward-facing vehicle-mounted camera and the corresponding automatically labelled binary ground truth images. Images were captured by stereoscopic camera mounted on the roof of an off-road vehicle, shown in Figure 5.5, at a rate of 10 frames per second. The 3D transformation matrix between each consecutive pair of stereo frames is computed using the visual odometry approach of Geiger et al [109] such that the transformation between any pair of frames within a sequence can be obtained. As the stereo depth data is only required for the creation of this ground-truth data, a deployed version of this system would require only a monocular camera.

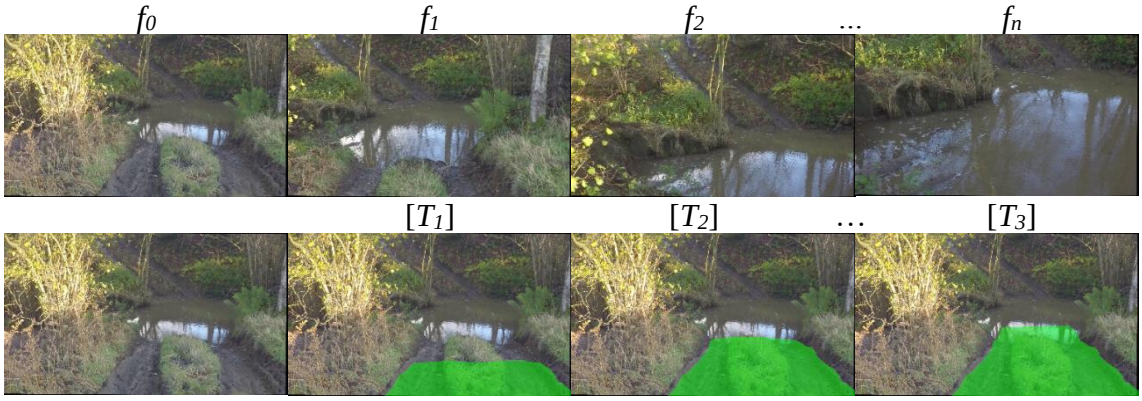

Figure 5.6 An example image sequence from our data set: starting from frame f_0, transformation matrices $[T_1]$ to $[T_n]$ are computed for camera position in n subsequent frames, from which vehicle footprint can be calculated and translated back into image space so that path pixels can be labelled. Top row contains original frames f_0 to f_n, while bottom row shows aggregate computed footprint at each frame overlaid onto f_0.

To select the frames that will form our dataset, we begin at the start of a sequence and look for the first frame containing movement, f_0, for which we create a label image L of matching dimensions with every pixel labelled as '*not path*'. Subsequent transformation matrices, T_1, T_2, … T_n are used to calculate the location and orientation of the camera, and by extension the vehicle footprint, in respective frames, f_1, f_2, … f_n, relative to the camera position in f_0. We then reproject this vehicle footprint back into camera space at f_0, labelling any pixel $L_{x,y}$ contained within as '*path*'. This process, illustrated in Figure 5.6, continues until the magnitude of the global transformation vector between the camera positions in f_0 and f_n is greater than distance threshold D, at which point the process is

started again using the frame midway between f_0 and f_n as the new starting point. Empirically we found D = 20m to be an appropriate value, as larger distances tended to cause noticeable propagation of transformation errors during the stereo visual odometry process.

We collected two sets of data, each using a different stereo camera (firstly a pair of GoPro Hero 4 [110] cameras placed on a 3D printed mount with a stereo baseline of 400mm, then a Carnegie Robotics MultiSense S21 [111], with a baseline of 210mm), vehicle and location to ensure variability within the data, and mirrored versions of every image were added to ensure an equal frequency of left and right turns. In total, our dataset comprises ~1000 RGB images of dimensions 512 × 288 along with corresponding binary ground truth images of the same dimensions. We use a 90/10 split to divide our data into training and test sets. The challenges associated with capturing off-road data, along with the lack of publicly available datasets, are significant areas for future consideration.

5.2.1.2 Network Architectures

We use our data to train three CNN architectures: Segnet [1], Fully Convolutional Network (FCN) [80], and u-net [81], each of which represents a slightly different approach to the concept of an encoder-decoder architecture for pixelwise labelling and segmentation.

The design of Segnet, discussed further in Chapter 3, was motivated by segmentation and classification of road scenes for autonomous driving. It is a symmetrical architecture comprising an encoder based on the VGG16 [53] network, with fully connected layers removed, followed by a mirror-image decoder. The encoder consists of 13 convolution layers, each using a 3 × 3 kernel followed by batch normalization and rectified linear units, interspersed with 5 max pooling layers, each downsampling its input by a factor of 2. The decoder replaces the max pooling layers with upsampling layers that use the

corresponding max-pooling indices from the encoding phase to restore some of the spatial information that would otherwise be lost to pooling. The final layer of the network is a softmax classifier of dimensions $c \times w \times h$ where c is the total number of classes (in this case 2: {*path*, *not path*}), and w and h are the dimensions of the original input image.

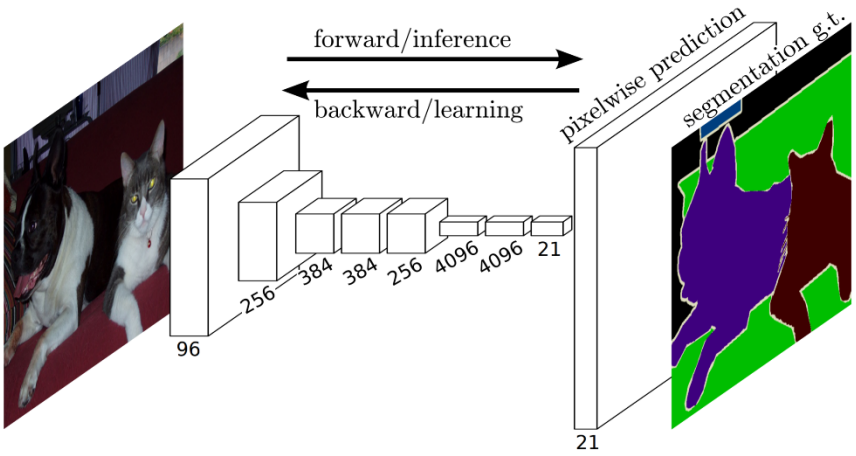

Figure 5.7 An example of an FCN for performing a semantic segmentation task [80]

FCN, shown in Figure 5.7, was designed primarily for object segmentation and classification, and while versions have been implemented based on various network architectures, we use a version based on VGG16 for better comparison with Segnet. The fully connected layers originally used in VGG16 are replaced by convolution layers with large receptive fields, and dropout is used to reduce the risk of overfitting. A 1×1 convolution is used to predict class likelihoods at each scale during the decoder phase, which are then concatenated with the corresponding max-pooling output before upscaling. Output is again a softmax classifier of dimensions $c \times w \times h$.

U-net, shown in Figure 5.8, was initially motivated by segmentation of medical imagery, and is a more compact architecture than Segnet or FCN, comprising eighteen 3×3 convolutions and 4 each of pooling and upsampling operations. The output of each upsampling operation is concatenated with the input of the corresponding pooling operation in order to retain spatial information that would otherwise be lost through downsampling. Again, Output is a softmax classifier of dimensions $c \times w \times h$. All three

networks are implemented in Caffe [112], and in the case of Segnet and FCN we use models made publicly available by the original authors [1] [80].

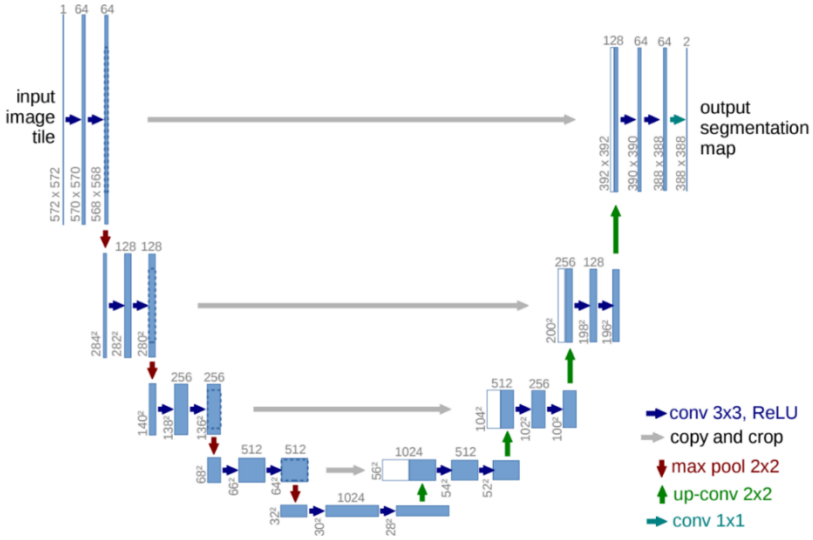

Figure 5.8 U-net CNN architecture [81].

The networks are pretrained on an off-road semantic segmentation task, as detailed in Chapter 3, with pixels initially being labelled as one of eight classes: sky, water, dirt, paved road, grass, foliage, tree, man-made obstacle. The final output layer of the network is then changed to only output two classes (path, not path) and the model is fine-tuned on our path prediction dataset. The motivation behind this is that a network trained to identify individual classes within a scene may be better equipped to differentiate between surfaces that are drivable, for example dirt or paved road, and those that are not, for example tree or sky. In order to test this hypothesis, we also evaluate the networks without this pre-training step, instead initialising weights randomly.

Our training data is passed through each network in batches of six images, with the final softmax classification error computed at every pixel to give a training loss value for the entire batch, which is propagated backwards through the network using stochastic gradient descent with momentum (we set momentum = 0.9 for all three network architectures). For Segnet we use a step function to decrease learning rate over training epochs, while the learning rate is fixed for u-net and FCN, as we attempt to replicate the

work of the original authors as best as possible in each case. The initial learning rate is selected empirically to enable parameter fine-tuning without overfitting: for both Segnet and u-net the value selected for initial loss was 1×10^{-3}, while for FCN it was 1×10^{-10}. A weight decay of 5×10^{-4} is also used in all three cases. Training continues until training loss is minimized such that no further gains are observed: this happened after 60,000 iterations for Segnet, 30,000 iterations for u-net, and 100,000 iterations for FCN, most likely due to the lower learning rate.

5.2.1.3 Post Processing

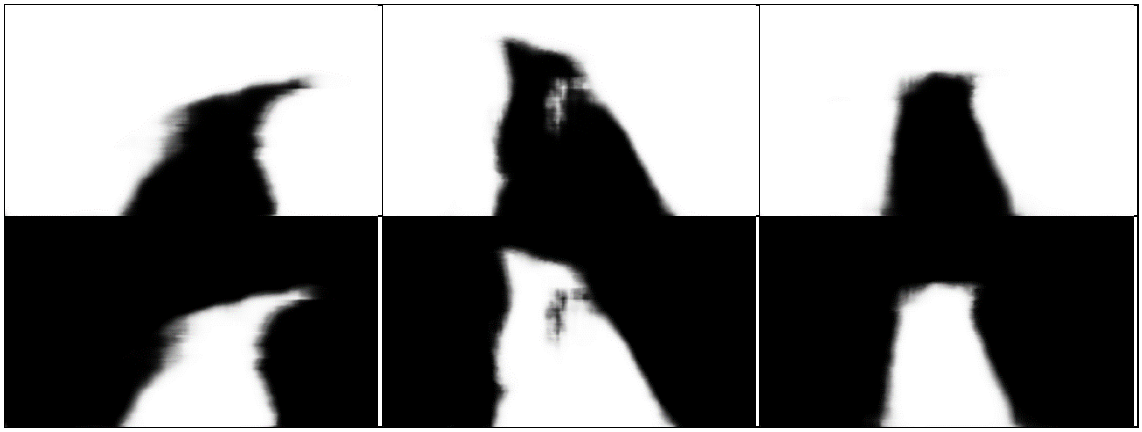

Figure 5.9 Example CNN output - top row shows C1, bottom row shows corresponding C0 before post-processing.

CNN output, shown in Figure 5.9, is an array of dimensions $2 \times w \times h$, where one channel contains a map of confidence values C_0 $\{0 \rightarrow 1\}$ that express the likelihood that a given pixel belongs to the class '*path*', and the other channel contains a map C_1 $\{0 \rightarrow 1\}$ expressing the confidence that each pixel belongs to the class *'not path'*, such that C_0 and C_1 are the inverse of each other and $C_{0,x,y} + C_{1,x,y} = 1$. As C_1 only contains information that can easily be inferred from C_0, it can be discarded.

An additional post-processing step is applied to C_0 to create the final path confidence map for evaluation against ground truth data. Firstly, we use the stereo disparity available with our source data to compute the distance from the camera to each pixel location in the image, and any pixel further than the distance threshold D = 20m is set to 0, as these

pixels will have been ignored during the ground truth creation process. We then convolve the image with a Gaussian kernel of $\sigma = 6$ to smooth out any high-frequency noise.

Next, we set the confidence values of all pixels that are disconnected from the main path segment to 0. For the purpose of this step, we use a very low path confidence threshold δ and set all pixels where $C_0 < \delta$ to 0. Empirically, we found a value of $\delta = 0.025$ to give the best results. If the image contains multiple disconnected path segments, we determine which to consider the actual path by finding the pixel where $C_{0,x,y} > \delta$ with the smallest Euclidean distance from $C_{0,w/2,h}$ (i.e. the central pixel of the bottom row of the image), and performing a flood fill operation starting at that point that treats pixels with a value of 0 as component boundaries. Any pixel that is outside of the component filled by this operation is set to 0.

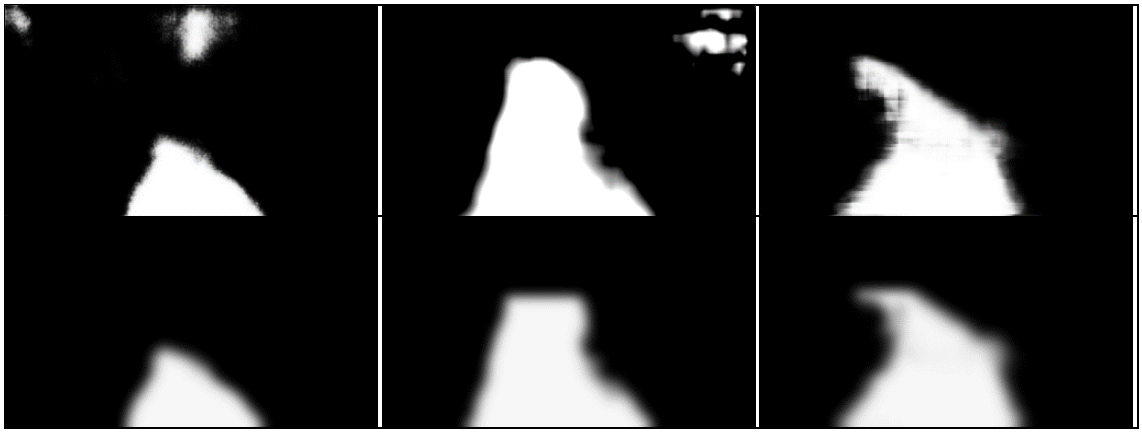

Figure 5.10 Example output path confidence maps, before (top row) and after (bottom row)post processing. In addition to smoothing the output with Gaussian kernel, unconnected path segments have been removed.

Some examples of output confidence maps before and after post-processing are shown in Figure 5.10 to demonstrate the effects of this step.

5.2.1.4 Evaluation Methodology

We evaluate the performance of the three trained networks, both with and without the post processing steps detailed above, using the 10% of our dataset that was set aside for testing.

In all cases we threshold the output path confidence map such that any pixel that satisfies the condition $C_{0,x,y} > 0.5$ is labelled '*path*', and any that does not is labelled '*not path*'. We compare this thresholded image to the ground truth data to compute the following four metrics across the entire test dataset:

Accuracy expresses the proportion of total image pixels that are classified correctly as either '*path*' or '*not path*', as shown in equation (5.1 (*tp* = true positives (the number of pixels correctly labelled as '*path*'), *tn* = true negatives (the number of pixels correctly labelled as '*not path*'), *fp* = false positives (the number of pixels incorrectly labelled as '*path*', and *fn* = false negatives (the number of pixels incorrectly labelled as '*not path*').

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{5.1}$$

Precision expresses the proportion of those pixels that are labelled '*path*' in the output image that are also labelled as such in the ground truth data, as shown in equation (5.2.

$$Precision = \frac{tp}{tp + fp} \tag{5.2}$$

Recall expresses the proportion of the pixels that are labelled '*path*' in the ground truth data that are correctly classified as such in the output image, as shown in equation (5.3.

$$Recall = \frac{tp}{tp + fn} \tag{5.3}$$

Intersection over union (IoU) expresses the number of pixels that are correctly classified as '*path*' as a proportion of the total number of pixels that are either labelled as such in the output image or labelled as such in the ground truth data, as shown in equation (5.4.

$$IoU = \frac{tp}{tp + fp + fn} \tag{5.4}$$

In all four cases, values are expressed in the range {0→1} with a higher value demonstrating superior performance.

5.2.2 Results

Our results are shown in Table 5.1 with illustrative examples shown in Figure 5.11 based on an evaluation over our test dataset. The top three rows show good results from FCN, Segnet and u-net respectively. The penultimate row shows an example of a poor result obtained from u-net, in this case likely caused by a combination of shadow and water on the ground. The bottom row shows an example where the track forks in front the vehicle creating two valid paths, although our ground truth only includes the path that the vehicle originally took.

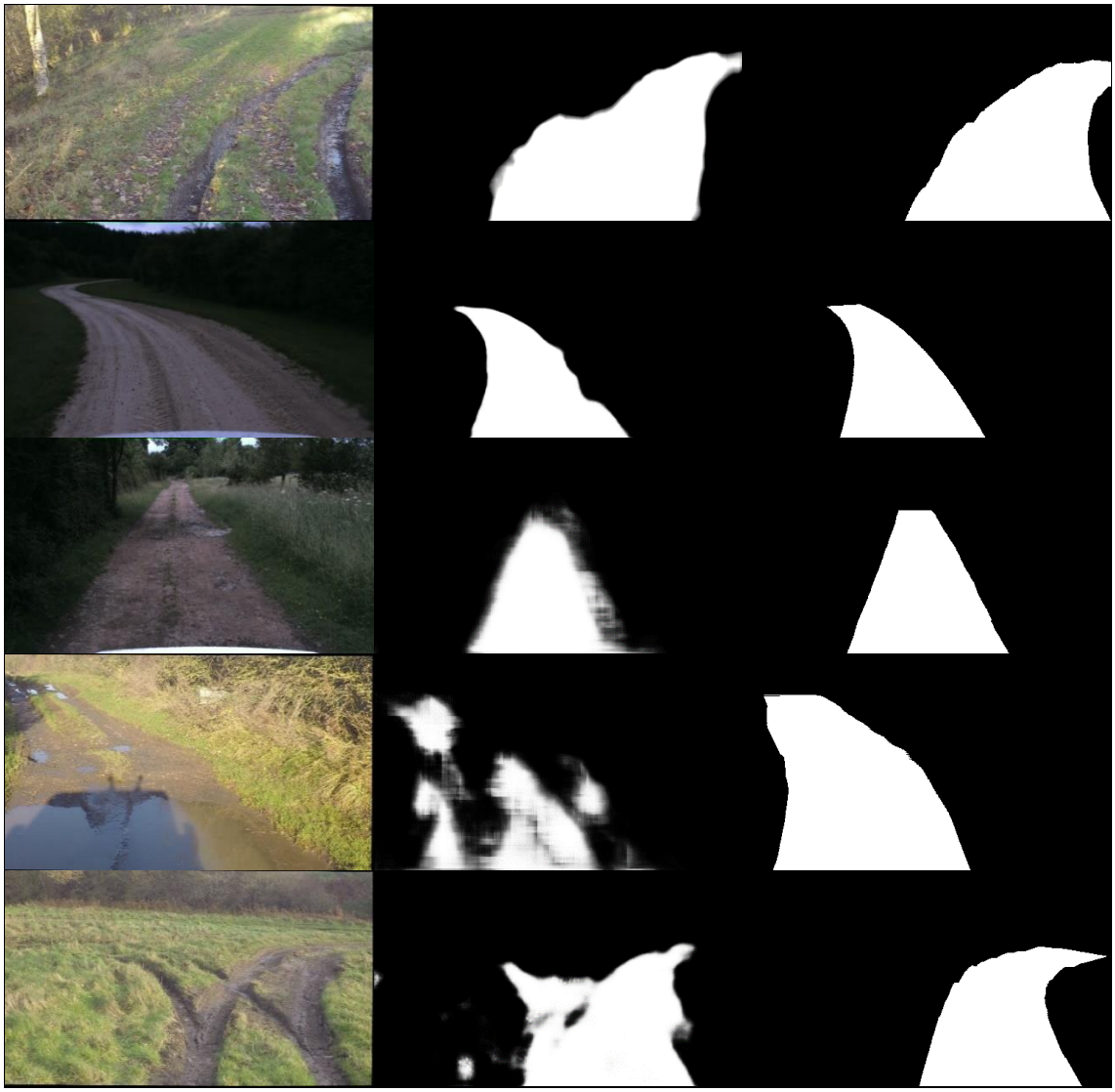

Figure 5.11 Example images from our test data set. Left: input, centre: raw CNN output prior to post-processing, right: ground truth

Detailed results are shown in Table 5.1. In terms of accuracy, the performance was similar across all three network types, with little impact from either pre-training or post-processing - the randomly initialised Segnet and FCN models both demonstrated an accuracy of 0.94, while u-net performed marginally better at 0.95. Segnet performance appears to benefit slightly from pre-training while FCN and u-net were negatively affected. Post-processing also raised accuracy slightly for Segnet, however no change was observed for the other two networks.

While these figures may appear highly impressive, accuracy alone is of limited utility in this case as it takes account of every pixel across the entire image when certain image regions will consistently contain only one label across the entire dataset, for example the top half of the image is almost exclusively '*not path*' in most cases. To demonstrate this, we compared every output path confidence map to every non-corresponding ground truth image (i.e. those derived from different input images) and found the mean accuracy in this case to be 0.84. This implies that there could be a risk that a CNN optimize to a local minimum whereby it outputs a single average path that demonstrates high accuracy across the entire training dataset, however our results show that this did not happen.

Recall again demonstrated very similar performance from Segnet and u-net, while FCN performs slightly worse. A small improvement is demonstrated by the Segnet model that has undergone pre-training, however in all other cases both pre-training and post-processing have a negative impact on recall. In most cases the opposite is true of precision, which increases slightly with post processing – from 0.84 to 0.85 in the case of FCN, from 0.86 to 0.88 in the case of Segnet and from 0.86 to 0.89 in the case of u-net. This is because the post-processing step will have removed or decreased the confidence value of more path pixels than it added or increased. U-net is the only network for which precision appears to be negatively affected by pre-training, dropping to 0.82.

Regarding intersection over union, performance is again similar between Segnet and u-net, however u-net output would appear to benefit the most from post-processing, with its IoU improving from 0.75 to 0.77 (Table 5.1). Again, FCN was the worst performer (0.72), and neither the results from it nor from Segnet showed any improvement with post-processing. Pre-training appears to have only slightly improved the performance of Segnet, while having a negative impact on FCN and u-net. We believe IoU to be the most useful metric for measuring performance at this task as it takes account of both false positives and false negatives while ignoring true negatives, which make up a significant proportion of the data and are part of the reason accuracy is so high: in the experiment described above where path confidence maps were compared against non-corresponding ground truth images, the mean IoU was only 0.41.

Network	Pretraining	Post Processing	Accuracy	Recall	Precision	IoU
SegNet	No	No	0.94	0.85	0.86	0.75
SegNet	Yes	No	**0.95**	**0.86**	0.87	0.76
SegNet	No	Yes	**0.95**	0.85	0.88	0.76
FCN	No	No	0.94	0.84	0.84	0.72
FCN	Yes	No	0.93	0.71	**0.89**	0.65
FCN	No	Yes	0.94	0.82	0.85	0.72
U-net	No	No	**0.95**	**0.86**	0.86	0.75
U-net	Yes	No	0.93	0.81	0.82	0.69
U-net	No	Yes	**0.95**	0.85	**0.89**	**0.77**

Table 5.1 For each network type used we display the results obtained from the raw output of a model initialised with random weights, along with our attempts to improve on these by pre-training on a semantic segmentation task and post-processing the output data in the manner detailed above.

5.3 Summary

In this chapter we have proposed an approach to off-road path prediction that combines a novel method for automatically creating labelled training data with state of the art convolutional neural network architectures designed for pixelwise labelling and segmentation tasks.

We created our own off-road dataset which we used to train networks based on the Segnet [1], Fully Convolutional Network [80], and u-net [81] architectures. These networks were then evaluated using our testing dataset, and all three demonstrated good performance. Overall, the best result was obtained from u-net output, which considering its advantages in terms of speed and memory usage would make it ideally suited for deployment as part of an autonomous vehicle solution.

Our approach addresses several limitations with existing end-to-end driving methods, extending the work of Bojarski et al [2], which itself built upon the work of Pomerleau [19] and Muller [39], in several ways. Firstly, our approach predicts the vehicle path up to a set distance, while existing approaches only consider immediate control inputs. Secondly, our approach is not tied to any specific vehicle or steering apparatus, whereas existing approaches are only applicable to the vehicle used to generate training data, as any change to steering ratio, turning circle, or steering method, for example on a tracked vehicle, would require different control inputs. By removing the direct link between CNN output and vehicle controls, we also believe that our approach is better suited to off-road environments where traction may be limited, and vehicle movement might not necessarily correlate with steering direction. Finally, by implementing one of these approaches ourselves, we demonstrated an additional limitation, in its proclivity for predicting low-magnitude steering inputs.

5.3.1 Future Work

One aspect we did not explore with this work is temporal consistency – the data used is derived from video sequences, and so an additional constraint that filters predicted path across consecutive frames would likely improve performance. Another potential approach for achieving this could be the modification of CNN architecture such that some

information from previous frames is retained to inform future predictions. The size and variability of available data is another area for consideration.

Chapter 6

Conclusions

In this chapter we conclude our findings and discuss the contributions made to the state of the art. We outline the limitations of the work, the implications and impact to industry, and possible future directions building upon this research.

Despite a rapidly growing body of research into autonomous road vehicles [3] [4], there is comparatively little work investigating the significantly more challenging problem of autonomous driving in off-road environments. The work described here has gone some way to addressing this, demonstrating two important capabilities in key areas of this problem domain: semantic segmentation of off-road scenes and path planning through unstructured environments.

We have demonstrated that an existing deep convolutional neural network classification and segmentation architecture [1] can be adapted to segment and classify off-road scenes, concluding that such an architecture can learn to successfully classify new kinds of images even with minimal training time and data through the technique of transfer learning. We then investigated the effects of the addition of several encodings of stereoscopic depth information to CNN input data, concluding that only marginal improvement in scene understanding performance resulted. We can conclude, contrary to popular belief, that stereo depth sensing does not strongly contribute to semantic scene understanding in the less structured off-road environment.

We have also investigated the application of end-to-end learning approaches to the problem of off-road path planning, evaluating an existing approach [2], which uses data generated by a human driver to train a CNN to predict steering inputs. After identifying

several limitations with this approach, we addressed these with our own proposed approach, in which a CNN is trained to predict vehicle path visually within an input image from data generated by a human driver: firstly, our approach predicts the vehicle path up to a set distance, while existing approaches only consider immediate control inputs. Secondly, our approach is not tied to any specific vehicle or steering apparatus, whereas existing approaches are only applicable to the vehicle used to generate training data, as any change to steering ratio, turning circle, or steering method, for example on a tracked vehicle, would require different control inputs. By removing the direct link between CNN output and vehicle controls, we also believe that our approach is better suited to off-road environments where traction may be limited, and vehicle movement might not necessarily correlate with steering direction. Finally, by implementing one of these approaches ourselves, we demonstrated an additional limitation, in its proclivity for predicting low-magnitude steering inputs.

6.1 Contributions

In this work we have demonstrated several key contributions towards the development of off-road autonomous vehicles.

In Chapter 3 we demonstrated a successful approach to off-road scene understanding, achieving state of the art performance by adapting and retraining a CNN architecture designed for semantic segmentation of urban road scenes [1] for the kind of unstructured environments encountered off-road. This builds on existing work in the area of autonomous on-road driving [1] by demonstrating that this approach is applicable to much more challenging off-road environments.

In Chapter 4 we attempted to improve our semantic segmentation performance with the addition of depth information, however we found this to be of limited utility when performing scene understanding tasks in off-road environments. This would appear to

contradict existing work [63] that demonstrated positives results using a similar approach in structured indoor environments.

In Chapter 5 we built on our semantic segmentation work to focus on binary path segmentation, using similar encoder-decoder networks and proposing a novel automated data annotation process to overcome the problem of time consuming image labelling. We implemented an existing end-to-end learning approach for autonomous driving [2], and demonstrated several limitations with such approaches when focusing on off-road environments, which we addressed with our approach.

Overall, we demonstrated how state-of-the-art deep learning techniques can be used to segment and classify off-road scenes to inform the decision-making process of autonomous off-road vehicles. Despite the challenges inherent in off-road environments, we demonstrated that good multi-class semantic segmentation and binary path segmentation results can be achieved with only monocular imagery.

6.1.1 Industrial Impact

As on-road autonomous driving technology matures, it is inevitable that the automotive industry will look for ways to apply similar techniques in off-road environments. Agriculture [6], military [5] and planetary exploration [7] are all areas that such technology could potentially save lives and reduce costs. Scene understanding and path planning are two of the core problems that form part of the processing pipeline of such an autonomous vehicle, and in this work we have successfully demonstrated proposed solutions to both.

6.1.2 Limitations

The data used for training and evaluation of this work was severely limited in both quantity and variety, due to the lack of publicly available off-road data sets and limited resources that could be devoted to the acquisition and annotation of data. All data used

was captured in rural environments in England under similar weather conditions, and so it is unlikely that the CNN models trained using this data would be capable of generalising well to other environment types without further training. If similar methods were to be deployed on an autonomous off-road vehicle platform, training and evaluation would need to be undertaken with data from as wide a variety of environmental, weather and lighting conditions as possible to ensure good generalisation.

Speed and computational complexity is another limitation that would affect the suitability of this work for deployment. Training and evaluation was undertaken on a Nvidia Geforce Titan XP [113], a high-end GPU with significantly greater capabilities than what could feasibly be deployed on a vehicle. Using this hardware scene understanding was able to run at ~10fps using the Segnet architecture, while path prediction using unet was marginally quicker, however on a device that meets the space and power requirements of a mobile platform, a significant drop in performance would likely be observed. Although off-road vehicles typically move at a lower speed than on-road vehicles and obstacles tend to be stationary so high-speed performance is not as important.

6.2 Further Work

We have demonstrated the capability to semantically segment off-road scenes, as well as to visually predict vehicle path through unstructured environments. Despite these successes, there remain areas in which this work could be built upon.

6.2.1 Data

As discussed in Section 6.1.2, limits to the quantity and variability of data used for this work limit the ability of CNN models to learn to generalise. A larger data set, comprising data acquired in a wide variety of locales, under varying weather and lighting conditions could be used to expand upon this work to demonstrate that a robust model capable of generalising to many types of off-road environment can be trained.

6.2.2 Temporal Consistency

Both scene understanding and path prediction approaches use individual frames extracted from video sequences as input, and so the addition of a technique to increase consistency of predictions across subsequent frames could improve accuracy. This could be implemented as a post-processing step, whereby predictions that are consistent with those seen in previous frames are given priority over those that differ, or built into the CNN architecture as a recurrent step which retains data from previous frames to inform subsequent decisions, such as in [114].

Bibliography

[1] V. Badrinarayanan, A. Kendall and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence,* vol. 39, no. 12, pp. 2481-2495, 2017.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller and J. Zhang, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316,* 2016.

[3] H. Badino, U. Franke and D. Pfeiffer, “The stixel world-a compact medium level representation of the 3d-world,” in *Joint Pattern Recognition Symposium,* 2009.

[4] A. Ess, T. Müller, H. Grabner and L. J. Van Gool, “Segmentation-Based Urban Traffic Scene Understanding.,” in *British Machine Vision Conference,* 2009.

[5] D. Braid, A. Broggi and G. Schmiedel, “The TerraMax autonomous vehicle,” *Journal of Field Robotics,* vol. 23, no. 9, pp. 693-708, 2006.

[6] H. Fang, R. Fan, B. Thuilot and P. Martinet, “Trajectory tracking control of farm vehicles in presence of sliding,” *Robotics and Autonomous Systems,* vol. 54, no. 10, pp. 828-839, 2006.

[7] P. Corke, D. Strelow and S. Singh, “Omnidirectional visual odometry for a planetary rover,” in *Intelligent Robots and Systems,* 2004.

[8] I. Gheorghe, W. Li, T. Popham, A. Gaszczak and K. J. Burnham, “Key learning features as means for terrain classification,” *Advances in Systems Science,* pp. 273-282, 2014.

[9] L. Lu, C. Ordonez, E. G. Collins and E. M. DuPont, “Terrain surface classification for autonomous ground vehicles using a 2D laser stripe-based structured light sensor,” in *Intelligent Robots and Systems*, 2009.

[10] R. Manduchi, A. Castano, A. Talukder and L. Matthies, “Obstacle detection and terrain classification for autonomous off-road navigation,” *Autonomous robots,* vol. 18, no. 1, pp. 81-102, 2005.

[11] A. Krizhevsky, I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012.

[12] K. He, X. Zhang, S. Ren and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015.

[13] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny and G. Hoffmann, “Stanley: The robot that won the DARPA Grand Challenge,” *Journal of field Robotics,* vol. 23, no. 9, pp. 661-692, 2006.

[14] A. B. Hillel, R. Lerner, D. Levi and G. Raz, “Recent progress in road and lane detection: a survey,” *Machine vision and applications,* vol. 25, no. 3, pp. 727-745, 2014.

[15] T. NVIDIA, “K1: A new era in mobile computing,” *Nvidia, Corp., White Paper,* 2014.

[16] H. Kong, J.-Y. Audibert and J. Ponce, “General road detection from a single image,” *IEEE Transactions on Image Processing,* vol. 19, no. 8, pp. 2211-2220, 2010.

[17] V. Subramanian, T. F. Burks and A. Arroyo, “Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation,” *Computers and electronics in agriculture,* vol. 53, no. 2, pp. 130-143, 2006.

[18] D. B. Gennery, “Traversability analysis and path planning for a planetary rover,” *Autonomous Robots,* vol. 6, no. 2, pp. 131-146, 1999.

[19] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” in *Advances in neural information processing systems,* 1989.

[20] M. Bertozzi and A. Broggi, “GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection,” *IEEE transactions on image processing,* vol. 7, no. 1, pp. 62-81, 1998.

[21] Institute of Advanced Motorists, “Licensed to Skill - Contributory Factors in Road Accidents Great Britain 2005 - 2009,” 2016. [Online]. Available: https://www.iamroadsmart.com/docs/default-source/research-reports/report_licensed-to-skill-v2.pdf.

[22] SAE On-Road Automated Vehicle Standards Committee, “Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems,” *SAE International,* 2014.

[23] Volvo Cars, “Your Guardian Angel,” 2018. [Online]. Available: https://www.volvocars.com/intl/cars/new-models/xc60/stories/autonomous-driving.

[24] Meredes-Benz, “Intelligent Drive next Level as part of Driving Assistance package,” [Online]. Available: https://www.mercedes-benz.com/en/mercedes-benz/innovation/with-intelligent-drive-more-comfort-in-road-traffic/.

[25] Car Magazine, “How did Audi make the first car with Level 3 autonomy?,” 2017. [Online]. Available: https://www.carmagazine.co.uk/car-news/tech/audi-a3-level-3-autonomy-how-did-they-get-it-to-market/.

[26] Renault, “Autonomous Vehicles,” 2017. [Online]. Available: https://group.renault.com/en/innovation-2/autonomous-vehicle/.

[27] Waymo, “Technology,” [Online]. Available: https://waymo.com/tech/.

[28] R. Labayrade, J. Douret and D. Aubert, “A multi-model lane detector that handles road singularities,” in *Intelligent Transportation Systems Conference,* 2006.

[29] K. Kluge and G. Johnson, “Statistical characterization of the visual characteristics of painted lane markings,” in *Intelligent Vehicles Symposium,* 1995.

[30] H.-C. Choi, J.-M. Park, W.-S. Choi and S.-Y. Oh, “Vision-based fusion of robust lane tracking and forward vehicle detection in a real driving environment,” *International Journal of Automotive Technology,* vol. 13, no. 4, pp. 653-669, 2012.

[31] Intel, “Intel Completes Offer For Mobileye,” 2017. [Online]. Available: https://newsroom.intel.com/press-kits/mobileye/.

[32] Intel, “The Evolution of EyeQ,” 2017. [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq-chip/.

[33] Nvidia, “Nvidia Drive - Scalable Platform for Autonomous Driving,” [Online]. Available: http://www.nvidia.com/en-us/self-driving-cars/drive-px/. [Accessed February 2018].

[34] Nvidia, “NVIDIA AI Technology Will Process Massive Volumes of Sensor Data, Enabling Vehicles to Anticipate and Respond to Dynamic Driving Situations,” 2017. [Online]. Available: https://nvidianews.nvidia.com/news/nvidia-and-toyota-collaborate-to-accelerate-market-introduction-of-autonomous-cars. [Accessed February 2018].

[35] Nvidia, “Tesla and Nvidia,” [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/partners/tesla. [Accessed February 2018].

[36] Nvidia, “Go, Autonomous Speed Racer, Go! NVIDIA DRIVE PX 2 to Power World’s First Robotic Motorsports Competition,” [Online]. Available: http://blogs.nvidia.com/blog/2016/04/05/roborace/. [Accessed February 2018].

[37] Nvidia, “CES 2016: NVIDIA DRIVENet Demo - Visualizing a Self-Driving Future (part 5),” 2016. [Online]. Available: https://www.youtube.com/watch?v=HJ58dbd5g8g. [Accessed February 2018].

[38] C. Caraffi, S. Cattani and P. Grisleri, “Off-road path and obstacle detection using decision networks and stereo vision,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, no. 4, pp. 607-618, 2007.

[39] U. Muller, J. Ben, E. Cosatto, B. Flepp and Y. L. Cun, “Off-road obstacle avoidance through end-to-end learning,” in *Advances in neural information processing systems*, 2006.

[40] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics,* vol. 5, no. 4, pp. 115-133, 1943.

[41] A. G. Ivakhnenko and V. G. Lapa, “Cybernetic predicting devices,” Purdue University Lafayette Ind School of Electrical Engineering, 1966.

[42] P. J. Werbos, “Applications of advances in nonlinear sensitivity analysis,” *Systems Modelling and Optimization,* pp. 762-770, 1982.

[43] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning,* vol. 20, no. 3, pp. 273-297, 1995.

[44] L. Breiman, “Random forests,” *Machine learning,* vol. 45, no. 1, pp. 5-32, 2001.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition*, 2009.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla and M. Bernstein, “Imagenet large scale visual recognition

challenge," *International Journal of Computer Vision,* vol. 115, no. 3, pp. 211-252, 2015.

[47] F. Perronnin, Y. Liu, J. Sánchez and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Computer Vision and Pattern Recognition,* 2010.

[48] J. Hu, L. Shen and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507,* 2017.

[49] A. Karpathy, "What I learned from competing against a ConvNet on ImageNet," 2014. [Online]. Available: http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/. [Accessed February 2018].

[51] L. v. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research,* vol. 9, no. Nov, pp. 2579-2605, 2008.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed and D. Anguelov, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[54] 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.

[55] G. Larsson, M. Maire and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648,* 2016.

[56] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167,* 2015.

[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[58] R. Girshick, “Fast r-cnn,” *arXiv preprint arXiv:1504.08083,* 2015.

[59] J. Ba, V. Mnih and K. Kavukcuoglu, “Multiple object recognition with visual attention,” *arXiv preprint arXiv:1412.7755,* 2014.

[60] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney and K. Saenko, “Translating videos to natural language using deep recurrent neural networks,” *arXiv preprint arXiv:1412.4729,* 2014.

[61] S. Gupta, P. Arbeláez, R. Girshick and J. Malik, “Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation,” *International Journal of Computer Vision,* vol. 112, no. 2, pp. 133-149, 2015.

[62] B.-s. Kim, P. Kohli and S. Savarese, “3D scene understanding by voxel-CRF,” in *Computer Vision, IEEE International Conference on*, 2013.

[63] S. Gupta, R. Girshick, P. Arbeláez and J. Malik, “Learning rich features from RGB-D images for object detection and segmentation,” in *European Conference on Computer Vision*, 2014.

[64] P. Sturgess, K. Alahari, L. Ladicky and P. H. Torr, “Combining appearance and structure from motion features for road scene understanding,” in *British Machine Vision Conference,* 2009.

[65] L.-J. Li, R. Socher and L. Fei-Fei, “Towards total scene understanding: Classification, annotation and segmentation in an automatic framework,” in *Computer Vision and Pattern Recognition,* 2009.

[66] Y. Kang, K. Yamaguchi, T. Naito and Y. Ninomiya, “Multiband image segmentation and object recognition for understanding road scenes,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 12, no. 4, pp. 1423-1433, 2011.

[67] B. Demir and S. Erturk, “Hyperspectral image classification using relevance vector machines,” *IEEE Geoscience and Remote Sensing Letters,* vol. 4, no. 4, pp. 586-590, 2007.

[68] K. Van De Sande, T. Gevers and C. Snoek, “Evaluating color descriptors for object and scene recognition,” *IEEE transactions on pattern analysis and machine intelligence,* vol. 32, no. 9, pp. 1582-1596, 2010.

[69] S. Lazebnik, C. Schmid and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *Computer vision and pattern recognition, IEEE computer society conference on*, 2006.

[70] P. Filitchkin and K. Byl, “Feature-based terrain classification for littledog,” in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 2012.

[71] J. Tighe and S. Lazebnik, “Superparsing,” *International Journal of Computer Vision,* vol. 101, no. 2, pp. 329-349, 2013.

[72] A. Bosch, A. Zisserman and X. Muñoz, “Scene classification via pLSA,” in *European conference on computer vision,* 2006.

[73] P. Jansen, W. van der Mark, J. C. van den Heuvel and F. C. Groen, “Colour based off-road environment and terrain type classification,” in *IEEE Intelligent Transportation Systems,* 2005.

[74] D. Kim, J. Sun, S. M. Oh, J. M. Rehg and A. F. Bobick, “Traversability classification using unsupervised on-line visual learning for outdoor robot navigation,” in *Robotics and Automation, IEEE International Conference on,* 2006.

[75] B. Upcroft, C. McManus, W. Churchill, W. Maddern and P. Newman, “Lighting invariant urban street classification,” in *Robotics and Automation, IEEE International Conference on,* 2014.

[76] I. Tang and T. P. Breckon, “Automatic road environment classification,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 12, no. 2, pp. 476-484, 2011.

[77] C. Szegedy, A. Toshev and D. Erhan, “Deep neural networks for object detection,” in *Advances in neural information processing systems,* 2013.

[78] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition,* 2016.

[79] C. Farabet, C. Couprie, L. Najman and Y. LeCun, “Learning hierarchical features for scene labeling,” *IEEE transactions on pattern analysis and machine intelligence,* vol. 35, no. 8, pp. 1915-1929, 2013.

[80] J. Long, E. Shelhamer and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition,* 2015.

[81] O. Ronneberger, P. Fischer and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention,* 2015.

[82] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang and P. H. Torr, “Conditional random fields as recurrent neural networks,” in *Proceedings of the IEEE international conference on computer vision,* 2015.

[83] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, “Pyramid scene parsing network,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[84] S. Valipour, M. Siam, M. Jagersand and N. Ray, “Recurrent fully convolutional networks for video segmentation,” in *IEEE Winter Conference on Applications of Computer Vision,* 2017.

[85] M. Siam, S. Valipour, M. Jagersand and N. Ray, “Convolutional Gated Recurrent Networks for Video Segmentation,” in *IEEE International Conference on Image Processing,* 2017.

[86] W. Byeon, T. M. Breuel, F. Raue and M. Liwicki, “Scene labeling with lstm recurrent neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[87] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[88] N. Silberman and R. Fergus, “Indoor scene segmentation using a structured light sensor,” in *Computer Vision Workshops, IEEE International Conference on*, 2011.

[89] N. Silberman, D. Hoiem, P. Kohli and R. Fergus, “Indoor segmentation and support inference from rgbd images,” in *European Conference on Computer Vision,* 2012.

[90] S. Gupta, P. Arbelaez and J. Malik, “Perceptual organization and recognition of indoor scenes from RGB-D images,” in *Computer Vision and Pattern Recognition, IEEE Conference on*, 2013.

[91] M. S. Pavel, H. Schulz and S. Behnke, “Recurrent convolutional neural networks for object-class segmentation of RGB-D video,” in *Neural Networks, International Joint Conference on*, 2015.

[93] K. Chatfield and A. Zisserman, “Visor: Towards on-the-fly large-scale object category retrieval,” in *Asian Conference on Computer Vision,* 2012.

[94] J. M. Alvarez, T. Gevers, Y. LeCun and A. M. Lopez, “Road scene segmentation from a single image,” in *European Conference on Computer Vision,* 2012.

[95] G. J. Brostow, J. Fauqueur and R. Cipolla, “Semantic object classes in video: A high-definition ground truth database,” *Pattern Recognition Letters,* vol. 30, no. 2, pp. 88-97, 2009.

[96] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura and R. M. Summers, “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning,” *IEEE transactions on medical imaging,* vol. 35, no. 5, pp. 1285-1298, 2016.

[97] H. Bay, T. Tuytelaars and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*, 2006.

[98] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos,” in *International Conference on Computer Vision*, 2003.

[99] O. K. Hamilton, T. P. Breckon, X. Bai and S.-i. Kamata, “"A foreground object based quantitative assessment of dense stereo approaches for use in automotive environments,” in *International Conference on Image Processing*, 2013.

[100] H. Hirschmuller, “Accurate and efficient stereo processing by semi-global matching and mutual information,” in *Conference On Computer Vision and Pattern Recognition*, 2005.

[101] K.-J. Yoon and I.-S. Kweon, “Adaptive support-weight approach for correspondence search,” *Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, no. 4, pp. 650-656, 2006.

[102] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International journal of computer vision,* vol. 47, no. 1-3, pp. 7-42, 2002.

[103] F. Mroz and T. P. Breckon, “An empirical comparison of real-time dense stereo approaches for use in the automotive environment,” *EURASIP Journal on Image and Video Processing,* vol. 2012, no. 1, p. 13, 2012.

[104] S. Birchfield and C. Tomasi, “A pixel dissimilarity measure that is insensitive to image sampling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 4, pp. 401-406, 1998.

[105] Y. Zhan, Y. Gu, K. Huang, C. Zhang and K. Hu, “Accurate image-guided stereo matching with efficient matching cost and disparity refinement,” *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 26, no. 9, pp. 1632-1645, 2016.

[106] M. G. Mozerov and J. van de Weijer, “Accurate stereo matching by two-step energy minimization,” *IEEE Transactions on Image Processing,* vol. 24, no. 3, pp. 1153-1163, 2015.

[107] J. Kowalczuk, E. T. Psota and L. C. Perez, “Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences,” *IEEE transactions on circuits and systems for video technology,* vol. 23, no. 1, pp. 94-104, 2013.

[108] Nvidia, “Nvidia AI Car Demonstration,” 2016. [Online]. Available: https://www.youtube.com/watch?v=-96BEoXJMs0. [Accessed February 2018].

[109] A. Geiger, J. Ziegler and C. Stiller, “Stereoscan: Dense 3d reconstruction in real-time,” in *Intelligent Vehicles Symposium*, 2011.

[110] GoPro Inc., “GoPro Hero 4 Black,” [Online]. Available: http://www.gopro.com. [Accessed September 2017].

[111] Carnegie Robotics LLC, “MultiSense Stereo Compact & Accurate 3D Data Collection,” 2014. [Online]. Available: http://files.carnegierobotics.com/products/MultiSense_S21/MultiSense_Stereo_brochure.pdf. [Accessed April 2018].

[112] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

[113] Nvidia, “Nvidia Titan Xp,” [Online]. Available: https://www.nvidia.com/en-us/titan/titan-xp/. [Accessed April 2018].

[114] S. Tripathi, Z. C. Lipton, S. Belongie and T. Nguyen, “Context Matters: Refining Object Detection In Video With Recurrent Neural Networks,” *arXiv preprint arXiv:1607.04648,* 2016.

[115] L. V. D. Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning*, vol. 9, pp. 2579-2605, 2008.