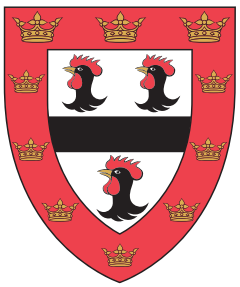
Learning Birds-Eye View Representations for Autonomous Driving



Thomas Edward Roddick

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Jesus College

March 2021

I would like to dedicate this thesis to my parents and to my brother, Alistair, who kept me motivated throughout it all.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Thomas Edward Roddick
March 2021

Learning Birds-Eye View Representations for Autonomous Driving

Thomas Edward Roddick

Over the past few years, progress towards the ambitious goal of widespread fully-autonomous vehicles on our roads has accelerated dramatically. This progress has been spurred largely by the success of highly accurate LiDAR sensors, as well the use of detailed high-resolution maps, which together allow a vehicle to navigate its surroundings effectively. Often, however, one or both of these resources may be unavailable, whether due to cost, sensor failure, or the need to operate in an unmapped environment. The aim of this thesis is therefore to demonstrate that it is possible to build detailed three-dimensional representations of traffic scenes using only 2D monocular camera images as input. Such an approach faces many challenges: most notably that 2D images do not provide explicit 3D structure. We overcome this limitation by applying a combination of deep learning and geometry to transform image-based features into an orthographic birds-eye view representation of the scene, allowing algorithms to reason in a metric, 3D space. This approach is applied to solving two challenging perception tasks central to autonomous driving.

The first part of this thesis addresses the problem of monocular 3D object detection, which involves determining the size and location of all objects in the scene. Our solution was based on a novel convolutional network architecture that processed features in both the image and birds-eye view perspective. Results on the KITTI dataset showed that this network outperformed existing works at the time, and although more recent works have improved on these results, we conducted extensive analysis to find that our solution performed well in many difficult edge-case scenarios such as objects close to or distant from the camera.

In the second part of the thesis, we consider the related problem of semantic map prediction. This consists of estimating a birds-eye view map of the world visible from a given camera, encoding both static elements of the scene such as pavement and road layout, as well as dynamic objects such as vehicles and pedestrians. This was accomplished using a second network that built on the experience from the previous work and achieved convincing performance on two real-world driving datasets. By formulating the maps as an occupancy grid map (a widely used representation from robotics), we were able to demonstrate how predictions could be accumulated across multiple frames, and that doing so further improved the robustness of maps produced by our system.

Acknowledgements

One thing that my amazing supervisor Roberto instilled in me from the very start of my graduate studies was that a PhD truly is a unique opportunity, to have such freedom to learn and discover the things that interest you. It has also been such an incredible personal experience, and there are so many people I'd like to thank for being a part of that and helping me along the way.

First of all to my supervisor, Roberto Cipolla. I cannot reiterate enough how grateful I am to have had this opportunity and for all your guidance and support over the years. In particular I'm so thankful for all the opportunities you gave me to meet and interact with amazing people, to attend summer schools in Sicily and for all your advocacy and generosity.

Another of Roberto's great skills is bringing together a research group of hugely talented people, and I've learned so much from working alongside them. I wanted to say thank you especially to Ignas Budvytis, Alex Kendall, Marvin Teichmann, James Charles and everyone else I've had the fortune to work with over the years. Thank you for all the sharing of ideas and tea-room discussions in front of a whiteboard. I could not have completed my PhD without the incredible support staff of Cambridge University Engineering Department: in particular Rachel Fogg, Phill Richardson, Peter Grandi and Raf Czlonka. I also wanted to add a quick thanks to Geoff Parks and Tim Wilkinson, who've indulged me for so long my dream of remaining a student in Cambridge forever.

Finally I would like to thank everyone I've had the fortune of getting to know at Jesus College. It's been my home not just for my PhD but for the past ten years of my life, and so many of my happiest memories have been made there. The list of great friends who've made my time there so incredible is too long to print in full, but just a few names who've been there for me throughout my PhD include: Chris Jones, Lisa Vickers, Yani Ioannis, Taylor Saunders-Wood, John Carpenter, Alexandra Forrester, Emma Findlay, and Tom and Jessica Powell. Thank you so much to my incredible partner Vanessa Knight, I couldn't have got there without your love and support. I also need to give special mention to Benjamin Biggs and Pip Liggins, who have been the most amazing housemates, lab partner, travel buddies and friends I could hope for. Thank you so much to everybody who has made this PhD the most incredible experience of my life.

Table of contents

List of figures

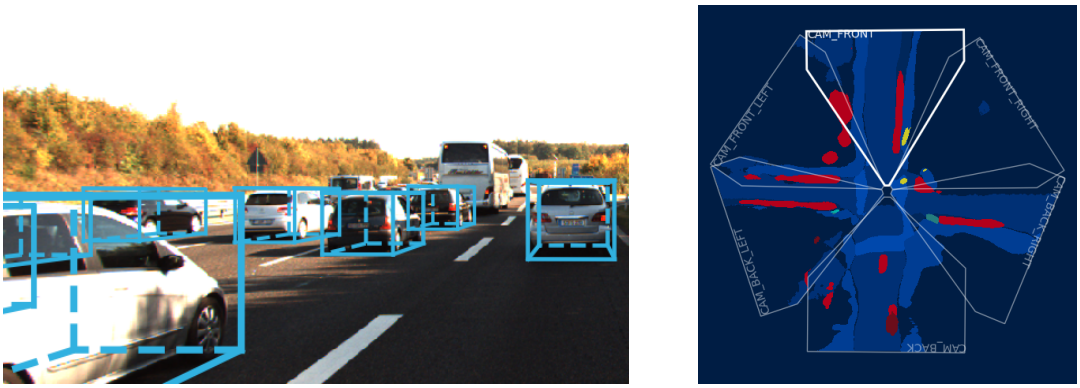
List of tables

Chapter 1

Introduction

Autonomous driving epitomises one of the most difficult and important challenges of robotics and embodied artificial intelligence: the ability to navigate autonomously in a complex, dynamic and human-populated world (Alatise and Hancke, 2020). Over recent years the problem has received a huge resurgence of interest, spurred both by significant advances in computer vision and deep learning technologies (Krizhevsky et al., 2012; Ren et al., 2015) as well as the huge benefits to society a solution is expected to bring (Herrmann et al., 2018). In this fast-paced environment, it is vital that the next generation of autonomous vehicles are safe and can exhibit an awareness of their surroundings (Wang, Huang, et al., 2020).

To that end, this thesis focuses on the problem of autonomous vehicle perception. In particular, we are interested in the specific setting where the primary source of information available to a vehicle is monocular images captured from an on-board camera system. This represents one of the most challenging scenarios for an autonomous system since raw camera data provides neither explicit information about the 3D structure of the scene nor semantic knowledge of the objects and obstacles it contains (Yurtsever et al., 2020). Deep-learning-based solutions have shown huge promise in solving image-based scene understanding tasks such as depth prediction (Eigen et al., 2014; Fu, Gong, et al., 2018), semantic segmentation (Badrinarayanan et al., 2017; Long et al., 2015) and 3D object detection (Ren et al., 2015; Redmon, Divvala, et al., 2016; Liu et al., 2016). Nevertheless, the underlying two-dimensional image representation represents a challenging modality to work with on account of the inherent perspective projection, which distorts the true three-dimensional nature of the scene. This thesis advocates a unique paradigm for solving tasks involving 3D scene understanding: to transform perspective image-based features into an orthographic, birds-eye view of the scene. This approach exploits the fact that road scenes are typically planar and allows a deep neural network to reason about the 3D structure of the scene whilst avoiding the intense computational costs of volumetric scene



(a) 3D object detection (b) Semantic map prediction

Figure 1.1. This thesis focuses on two key autonomous driving problems: 3D object detection and semantic map prediction. In both cases, we assume only monocular camera sensor data is available.

representations (Maturana and Scherer, 2015; Wu et al., 2015). Through the investigations described in this thesis, we show that such an approach improved performance compared to existing methods in solving monocular 3D scene understanding tasks.

This thesis focuses on solving two specific subtasks within the setting of autonomous vehicle perception and 3D scene understanding. The first problem we approach is that of 3D object detection; that is, determining the pose, object type, and dimensions of all specified objects in the scene (Geiger, Lenz, and Urtasun, 2012). Objects such as cars, pedestrians and bicycles are represented as a three-dimensional cuboid that must be accurately localised in space: an essential prerequisite for many downstream tasks such as object tracking (Chiu et al., 2020) and trajectory prediction (Chandra et al., 2020). While this representation captures many of the dynamic objects in the scene, it is difficult to incorporate amorphous surfaces such as the road, pavement and surroundings. Therefore, the second problem we tackle is that of semantic map prediction, which aims to build a complete birds-eye view representation of the scene. This representation incorporates both dynamic objects such as pedestrians and vehicles, as well as the layout of the road and other static elements.

In solving these two tasks, we adopt convolutional neural networks (CNNs) (Fukushima and Miyake, 1982; LeCun, Bengio, et al., 1995) as our primary tool of choice. We develop novel network architectures that build on existing ideas from the literature, which have shown that CNNs are powerful tools in reasoning about both the semantics and 3D geometry of the scene (Eigen et al., 2014; Badrinarayanan et al., 2017). However, a recurring theme in this thesis is that, where possible, existing knowledge about geometry and semantics should be built into these networks explicitly, rather than requiring them to be learned from scratch. This philosophy informs the use of the pinhole camera model (Hartley and Zisserman, 2003)

to transform image features into a birds-eye view representation, as well as the use of the Bayesian Filtering algorithm (Moravec and Elfes, 1985) to build large-scale maps over time. These topics are addressed in depth in Chapters 3 and 4. This thesis is organised as follows:

- The remainder of Chapter 1 motivates the work discussed in this thesis, including the significance of autonomous driving to society; the necessity to develop algorithms that operate on monocular images; and the reasoning behind the use of the birds-eye view representation, which underpins much of the subsequent research.

- Chapter 2 presents an overview of the relevant prior art within the field, covering general scene understanding topics within computer vision and autonomous driving, as well as literature surrounding the two core topics of this thesis: object detection and automatic map generation.

- Chapter 3 introduces the first task addressed by this thesis: that of detecting and localising 3D objects in video. The chapter describes a novel neural network component: the Orthographic Feature Transform; for mapping image-based representations to the birds-eye view. This component is employed as part of a neural network architecture called the OFTNet for 3D object detection. The OFTNet was evaluated on the pioneering KITTI autonomous driving dataset, and detailed analysis is presented in order to understand the network's underlying sources of error.

- Chapter 4 addresses the second main topic of semantic map prediction from monocular images. The approach of the work in this chapter was centred around a widely used framework called Bayesian occupancy grid mapping, which is outlined in Section 4.2. The map prediction problem was addressed using a second network architecture, building on many ideas from the OFTNet, but incorporated a novel transformer layer called the Dense Transformer. Two main strands of evaluation are considered in this chapter. The first focuses on map predictions from a single camera over a local spatial area. The second discusses building larger-scale maps by combining predictions using the Bayesian occupancy grid method.

- Chapter 5 considers the overall findings of the research presented in this thesis; their place within the modern research context as the field has progressed; and future avenues for research based on these results.

1.1 Motivation

1.1.1 Why autonomous driving?

Achieving fully autonomous driving has emerged as a major research goal across both academia and industry, and forms the central motivating application for this thesis. The widespread interest in this technology stems from the potentially seismic impact it may have on society, both in terms of the direct wellbeing and safety benefits to drivers and road users, as well as wider socio-economic issues. These impacts fall into four main categories:

Safety In the UK, road traffic accidents were responsible for 1,752 deaths and 25,945 serious injuries during 2019 (UK Department for Transport, 2019) and were respectively the second and third most prevalent causes of death among children and adults aged 5-19 and 20-35 (Office for National Statistics, 2020). Globally, the mortality from road accidents rises to 1.35 million deaths annually (World Health Organization, 2018). Autonomous driving technology would never be able to eliminate all of these deaths, but studies in the US have found that approximately 94% of accidents were caused by human error (National Highway Traffic Safety Administration, US Department of Transportation, 2015; National Highway Traffic Safety Administration, US Department of Transportation, 2008). Despite this huge potential to save lives, existing autonomous vehicles have not yet achieved the same level of safety as human drivers. A study by Dixit et al. (2016) found that autonomous vehicles (AVs) experienced an accident once every 48,000 miles, compared with once every 2.08 million miles for human drivers (although accidents were typically less severe for AVs). Therefore, improving the safety record of AVs (including through improved perception systems as discussed in this thesis) is a vital research imperative.

Environmental Autonomous vehicles also offer huge potential to reduce the environmental cost of transportation (Kopelias et al., 2020). The improved connectivity and control offered by AVs can reduce emissions and energy use by enabling driving behaviours such as intelligent speed adaptation, which avoids stop-start traffic flow; eco-driving, to optimise braking and acceleration; and platooning, which reduces the aerodynamic drag of a convoy of vehicles (Barth et al., 2014). Widespread uptake of autonomous technology could also offer macro-scale benefits, such as reducing traffic congestion due to vehicles travelling closer together at higher speeds; and lighter vehicle design as safety becomes less of a concern (Anderson et al., 2014). However, these potential environmental benefits should be considered with caution, as any net impacts will likely depend on the reaction of consumers. There is justified concern that the widespread adoption of autonomous vehicles will lead

to a considerable increase in total vehicle miles travelled as transportation becomes more convenient, with an accompanying rise in energy usage and emissions (Miller and Heard, 2016). On the other hand, AVs lend themselves to ridesharing and on-demand ownership models which would allow vehicles to be deployed more efficiently and reduce the size of the global vehicle fleet (Miller and Heard, 2016).

Social One possible outcome of autonomous driving is wider access to transport to the disabled, elderly, and other groups currently precluded from travelling independently (Ohnemus and Perl, 2016). Improved physical mobility has significant implications for social mobility, access to services and personal wellbeing (Herrmann et al., 2018, Chp. 34; Claypool et al., 2017). Widespread autonomy may also profoundly affect the design of towns and cities, as fewer inner-city areas need to be devoted to parking and may instead be reserved for parks and other public spaces (Johnsen et al., 2018).

Economic The UK Centre for Connected and Autonomous Vehicles (CCAV) forecasts the global market for autonomous vehicles to be worth £907bn globally, including up to £52bn in the UK, by 2035. It predicts this would create at least 9900 jobs in the UK AV technology industry and supply chain (Transport Systems Catapult, 2017). Such economic benefits, however, must be balanced against the potential disruption to other industries, such as the freight and taxi industries, which may suffer significant displacement due to the proliferation of autonomous technologies (Crayton and Meier, 2017).

In addition to the potential societal benefits of autonomous driving, it also represents an attractive topic from a research perspective. Many of the unsolved problems in this area exemplify the frontier of remaining challenges in the broader fields of computer vision and robotics. In particular, this thesis predominantly focuses on how to build detailed representations of the 3D world. This problem remains an open question across many domains: autonomous cars, unmanned aerial vehicles, indoor domestic robotics, and augmented reality systems all rely on the accurate understanding of their three-dimensional surroundings. Breakthrough in any one of these fields may be largely transferable across others.

Simultaneously, the challenges encountered by autonomous vehicles are particularly acute. They are required to operate within highly complex environments, navigate independently, and obey relevant traffic restrictions and interact with other dynamic and often unpredictable agents. They are subject to considerable constraints on computational power, communication bandwidth, energy use and processing time. Safety remains a paramount concern, meaning that algorithms must push the boundaries of accuracy, and when they do fail, they must do

so elegantly. The hope is that these requirements represent an upper bound for many other computer vision and robotics applications, and that achievements in this domain may help drive progress in the field as a whole.

Finally, from a pragmatic perspective, the current surge of public and commercial interest in autonomous driving has offered unique research opportunities. Many of the deep learning techniques applied in this thesis are constrained by the vast amounts of data required to train them. Over the duration of this PhD, companies and organisations including Waymo, Argo AI, Motional, Ford, Audi, Honda, Daimler, Karlsruhe Institute of Technology and others have released large-scale, publicly-available multi-modal datasets to help accelerate the progress of autonomous driving technology. These datasets have provided a unique test-bed to cultivate novel computer vision algorithms, particularly those which are data-constrained, in the hope that such developments can be passed on to other application areas as and when data becomes available.

1.1.2 Why monocular cameras?

Existing autonomous vehicles are generally equipped with an extensive array of sensing apparatus used to perceive their surroundings. A typical sensor suite comprises a range of different sensor types that offer complimentary benefits. Commonly deployed sensors include:

- **RADAR**, a low-cost active sensor capable of detecting objects at long ranges, but with limited accuracy and resolution.
- **LiDAR (Light detection and ranging)**, which provides highly accurate point distance measurements at medium ranges.
- **Ultrasound**, used to detect the proximity of objects at close range.
- **Monocular cameras**, passive sensors which can capture colour information at potentially high resolutions, but provide no automatic measurements of distance.
- **Stereo camera pairs**, consisting of two twinned cameras which can reconstruct 3D information from images by comparing the visual disparity.
- **IMU (Inertial measurement unit)**, a proprioceptive sensor which measures acceleration and orientation.
- **GNSS (Global navigation satellite system) receiver**, used to approximately localise the vehicle against a global coordinate frame.

At the start of this PhD, the vast majority of research into autonomous driving had focused on LiDAR as the primary perception modality. LiDAR has many advantages over other sensors: they are very high accuracy (can measure distances up to an accuracy of < 2cm), work in low-light conditions, and directly produce a 3D representation of the world with no need for further preprocessing. As a result, few existing works had explored the potential of other sensing modalities.

The start of this PhD also coincided with rapid advancement in computer vision algorithms, spurred on by the resurgence of deep neural networks for image classification (Krizhevsky et al. (2012)) and subsequent powerful deep learning architectures that followed (He, Zhang, et al. (2016), Huang, Liu, et al. (2017), and Ren et al. (2015)). Particularly relevant to this thesis were novel neural-network-based algorithms trained for geometric tasks such as pose regression (Kendall, Grimes, et al., 2015) and monocular depth estimation (Eigen et al., 2014). These networks demonstrated that it was possible to obtain accurate measurements of distance and reason about the world's 3D structure purely from monocular images. These developments raised an unavoidable question: could many of the achievements of a LiDAR-based system be replicated using a simple, low-cost camera? Indeed, aside from cost cameras offer additional benefits that LiDAR systems do not: for example, a typical Velodyne HDL-64E LiDAR sensor is effective up to a range of approximately 120m (although the sparsity of the points often limits their usefulness beyond 80-100m) and has a vertical resolution of 64 points (Geiger, Lenz, and Urtasun, 2012). Meanwhile, a typical inexpensive CMOS camera can have a resolution in the thousands of pixels, and its range is limited only by the physical sensor size. Additionally, images produced by a camera sensor capture colour information and are much more amenable to processing by existing algorithms, unlike LiDAR sensors which generate unstructured point cloud outputs.

Some prominent public figures have gone as far as suggesting that once image-based algorithms approach LiDAR-based systems' accuracy, cameras may ultimately replace LiDAR on autonomous platforms completely (McFarland, 2019). Even if this never comes to fruition, an autonomous vehicle that can operate effectively using cameras alone remains an important goal for redundancy reasons. No sensor can ever be relied on entirely, and having vehicles which can fall back on camera-only systems may be essential to bring AV safety in line with that of human drivers. Even if a failure is only partial, LiDAR and cameras have many complementary failure modes: cameras perform better in adverse weather such as rain or snow, while LiDAR is better suited to low-light conditions. Chapter 4 presents a framework which provides a natural mechanism for combining these two modalities if the results from a single sensor are uncertain.

Given the potential benefits of using monocular cameras over LiDAR, the essential requirement for sensor redundancy, and the relative lack of previous research in this field, there was a strong incentive to explore further the capacity of image-only systems. This thesis was therefore devoted to the study of scene understanding in autonomous driving from purely monocular cameras.

1.1.3 Why birds-eye view representations?

Whilst monocular images present many advantages as outlined above, they nonetheless represent a challenging modality to work with in the context of autonomous driving. A camera image is a projection of the 3D world onto a 2D plane. This means that much of the scene's structure, for example, the absolute distance between two objects, is discarded. Such information is vital for autonomous vehicles to infer the positions of obstacles or decide where to travel to next. Works such as Eigen et al. (2014) and Fu, Gong, et al. (2018) have shown that deep neural networks have the capability to recover some of this structure; using the known sizes of features in the scene to infer the distances of each pixel from the camera. The resulting 2.5D representation is called a depth or range image and encodes both colour and depth information about the scene. However, such a representation presents challenges for an autonomous agent trying to reason about its surroundings. In particular, perspective projection means that points which are close to each other in the depth image are not necessarily adjacent in the world. Such representations are also challenging from a learning point of view. For a given 3D object such as a car, its appearance and depth profile in the range image will vary dramatically depending on its position and orientation relative to the camera. As a result, a learning-based method such as a deep neural network will have to learn a different representation of the object for every possible configuration in the scene.

The ideal representation for an autonomous system would be one which closely mirrors the true 3D structure of the world, allowing the system to make informed judgements about absolute distances and speeds. Such a representation can easily be obtained from the range image in the form of a point cloud, where each pixel from the image is represented by a point in 3D space. These point clouds closely resemble those created directly by a range-measuring sensor such as LiDAR or RADAR. However, point clouds are inherently unstructured, making them difficult to process with existing convolutional neural networks (with some exceptions, see Qi, Su, et al. (2017)). Consequently, a common alternative to point clouds is to represent the world in the form of a voxel grid. A voxel grid is a 3D structure analogous to an image which partitions the world into cube-shaped cells called voxels. Because voxel grids are structured, they can easily be processed by convolutional neural networks. Unfortunately, this comes with a substantial computational footprint, as both complexity and memory usage

scales cubically with the voxel grid's size and resolution. In practice, this means that voxel grids are typically only feasible for small spatial volumes and low-resolution representations. Moreover, this representation is inherently inefficient, as the vast majority of voxels represent regions of empty space.

Fortunately, the types of scenes encountered in autonomous driving (and most ground-based robotics applications in general) exhibit a unique structure that can be exploited. Since all objects and agents in the scene are constrained by gravity, the structure and motion of the world perpendicular to the ground plane have little bearing on an autonomous agent's decisions. The vast majority of information relevant to driving can be summarised by a single plane, representing a 2D orthographic projection of the world along the direction of gravity. This projective space is referred to as the birds-eye view.

The birds-eye view representation shares many of the advantages of 3D point clouds: It is metric, so distances in the birds-eye view space are representative of distances in the real world (neglecting vertical distance, which does not inform path planning and decision making). The space is translationally equivariant, meaning that features remain the same regardless of their position on the birds-eye view. This property implies that objects' size and appearance remain constant, regardless of their distance from the sensor. Learning-based algorithms can therefore exploit strong priors about the world and frees up representational capacity. At the same time, we can choose to represent the birds-eye view as a structured image-like representation. This allows us to exploit many of the advancements in image processing conferred by deep convolutional neural networks. It is inherently efficient since every point on the birds-eye view must correspond to at least one point on the world's surface. It also does not suffer from the voxel grid's extreme scaling issues, allowing larger areas to be represented and in finer detail.

Given the factors discussed above, the birds-eye view seems to be the natural representation for 3D tasks. The challenge was how to obtain this representation in cases where full depth information was not available. Many previous works had focused on either building a birds-eye view from LiDAR (where full 3D information was available), or performing scene understanding from images alone. This PhD's unique characteristic was the aim to build birds-eye view representations directly from monocular images. This objective motivated the remaining chapters of this thesis.

1.2 Publications

The material discussed in this thesis is largely drawn from work which was later published in the form of two conference papers, in collaboration with other coauthors. Specifically, the

3D object detection research described in Chapter 3 contains results and material which was first introduced in:

> Thomas Roddick, Alex Kendall, and Roberto Cipolla (2019). "Orthographic feature transform for monocular 3d object detection". In: *Proceedings of the British Machine Vision Conference*

The material in Chapter 4 on semantic map prediction relates to the publication:

> Thomas Roddick and Roberto Cipolla (2020). "Predicting Semantic Map Representations from Images using Pyramid Occupancy Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11138–11147

All work discussed in this thesis was conducted by the author unless explicitly stated otherwise in the text.

Chapter 2

Literature Review

This thesis draws on a wide body of prior research across the fields of computer vision, robotics, and 3D scene understanding. This literature review will broadly mirror the structure of the thesis and introduce relevant related work for our two main tasks: 3D object detection and semantic map prediction. We begin by introducing the problem of 2D object detection (Felzenszwalb, Girshick, McAllester, and Ramanan, 2009), which defined many key concepts relevant to Chapter 3 such as bounding boxes, anchor boxes, and the distinction between single-stage and proposal-based architectures (Zhao et al., 2019). We then turn our attention to research which focuses on 3D object detection, specifically in the case where 3D information such as LiDAR points are available, which is assumed for most of the key works in the field (Ku, Mozifian, et al., 2018; Chen, Ma, et al., 2017; Qi, Liu, et al., 2018). Finally we consider works which, similarly to own, aim to recover 3D object bounding boxes using only monocular images as input. For the problem of semantic map prediction, we first review the various representations used to encode spatial information, including topological maps (Kuipers, 1977), continuous metric maps (Leonard and Durrant-Whyte, 1991a), and probabilistic occupancy grids (Elfes et al., 1990). We then discuss an important subset of mapping algorithms which address the task of Simultaneous Localisation and Mapping (SLAM) (Leonard and Durrant-Whyte, 1991b), and in particular those which, like our method, incorporate semantic information into their mapping pipeline. We consider the importance of mapping for downstream tasks such as trajectory prediction and motion planning (Masehian, 2015), and discuss how the requirements of these systems motivates our choice of representation in Chapter 4. Finally, we describe works which have leveraged advances in deep learning technology to learn to generate map-like representations from overhead or forward facing camera images.

2.1 Object Detection

2.1.1 2D Object Detection

Image-based object detection is one of the most fundamental problems in computer vision. It extends the problem of image classification by requiring an algorithm to not only determine the semantic category of an object of interest, but also to accurately localise it within the image. Generally, the location of the object is specified using a tightly-fitting axis-aligned bounding box, which also encodes the object's apparent size and aspect ratio. Unlike the problem of image classification, any number of instances of objects of the same or different object categories may be present within a given image.

The taxonomy of object detection algorithms largely consists of two-main families: two-stage methods which make use of initial bounding box proposals that are subsequently refined; and single-stage methods which predict bounding boxes directly from image features. We begin by reviewing early approaches to object detection before discussing both proposal-based and single-stage methods in depth.

Early methods

Most early object detection algorithms consisted of a two-stage approach: firstly extracting a dense set of features such as SIFT (Lowe, 2004), HOG (Dalal and Triggs, 2005) or Haar-wavelets (Mallat, 1989); and then subsequently applying a learning-based classifier such as Adaboost (Freund and Schapire, 1995) or support vector machines (SVMs) (Cortes and Vapnik, 1995) to classify regions in a sliding-window manner. The most well-known example is the seminal face detection algorithm of Viola and Jones (2001), which used Haar-like rectangular filters to extract features, followed by a cascade of boosted classifiers. This work was notable for being the first real-time object detector, introducing the integral image representation as a means of evaluating large numbers of rectangular features rapidly. Viola and Jones were preceded by Papageorgiou et al. (1998), who also used Haar-like input features together with a linear SVM classifier.

At this time, convolutional neural networks (CNNs) were already widely used in domain-specific object detection tasks, such as for handwritten text recognition (Matan et al., 1992; Delakis and Garcia, 2008), hand tracking (Nowlan and Platt, 1995), and face detection (Vaillant et al., 1994; Garcia and Delakis, 2004; Osadchy et al., 2007). However prior to the resurgence of deep learning after 2012 (Krizhevsky et al., 2012), the dominant general object detection algorithms were based on deformable parts-based models (DPMs) (Divvala et al., 2012), which were inspired by the pictoral structure models of Fischler and Elschlager

(1973) developed in the 1970s. Perhaps the most prominent DPM model was the approach of Felzenszwalb, Girshick, McAllester, and Ramanan (2009), which consisted of a set of part filters based on HOG features, arranged in a deformable star configuration about a central root filter. Latent SVMs (Andrews et al., 2003) enabled learning of the individual part models without direct supervision. Subsequent works improved the accuracy (Felzenszwalb, Girshick, and McAllester, 2010) and runtime (Yan, Lei, et al., 2014) of DPM-based approaches. Whilst DPMs remained popular in the early 2010s, CNN-based methods quickly came to dominate the object detection literature following the introduction of Region-CNN (Girshick et al., 2014).

Two-stage methods

A defining moment in the history of computer vision was the revolutionary neural-network-based classification algorithm of Krizhevsky et al. (2012), which demonstrated that convolutional neural networks could vastly outperform alternative algorithms for visual object recognition. A natural response therefore was to adapt such methods to object detection. Unfortunately, applying a deep neural network such as AlexNet exhaustively across all image locations, as was common in earlier CNN-based object detectors (Matan et al., 1992; Vaillant et al., 1994; Delakis and Garcia, 2008), proved to be prohibitively computationally expensive. A solution was to apply the CNN selectively to a small subset of image locations which were likely to contain objects, using an inexpensive region proposal algorithm such as selective search (Uijlings et al., 2013) or EdgeBox (Zitnick and Dollár, 2014). This was the rationale behind the Region-CNN (R-CNN) architecture of Girshick et al. (2014), inspired by work from Gu et al. (2009), which took initial region proposals from selective search and classified them using AlexNet.

In spite of the reduced computation from using region proposals however, R-CNN was still too slow to be useful in practical scenarios. To overcome this limitation, He, Zhang, et al. (2015) proposed to share convolutional features across all regions through the use of a spatial pyramid pooling layer (Lazebnik et al., 2006) which transforms arbitrarily-sized feature map regions into a fixed-size feature vector. This shared computation formed the basis for Fast R-CNN by Girshick (2015), which used a simplified version of spatial pyramid pooling called RoI-pool. Girshick further improved on R-CNN by adding a supervised box regression layer which refined the approximate bounding boxes produced by the selective search algorithm. The final component of the algorithm not to make use of the shared convolutional features was the initial region proposal stage; this was later addressed by Ren et al. (2015) which replaced selective search with a region proposal network (RPN) for generating regions at

multiple different scales. Their network: Faster R-CNN, provides the base architecture for many of the 3D object detection algorithms discussed in Section 2.1.2.

Since the emergence of Faster R-CNN as the dominant object detection architecture, various refinements have been proposed. One limitation of the RPN network is it is only able to propose regions with a relatively limited range of scales (three different sizes and three different aspect ratios). A long-established solution to this problem was to run the algorithm on images at multiple resolutions and aggregate the resulting bounding boxes (Adelson et al., 1984). A much more efficient alternative was proposed by Lin, Dollár, et al. (2017), which involved generating bounding boxes at multiple stages of the CNN feature hierarchy. To overcome the fact that features at early stages of the CNN have a high spatial resolution but fairly low-level semantic content, Lin, Dollár, et al. introduced the Feature Pyramid Network (FPN), in which low-level feature maps with strong semantics were upsampled to augment the features generated at earlier stages of the network. A very similar concept was proposed by Shrivastava et al. (2016), who focused on improving just the final layer feature representation.

One of the major strengths of the Faster R-CNN and its derivatives is its versatility. Performing additional downstream tasks such as instance segmentation (He, Gkioxari, et al., 2017), multi-person pose estimation (He, Gkioxari, et al., 2017) and 3D shape reconstruction (Gkioxari et al., 2019) can easily be accomplished by simply appending additional task-specific prediction branches alongside those used for classification and bounding box regression. It is partly as a result of this flexibility that has made Faster R-CNN the basis of many of the 3D object detection architectures which are the focus of the remainder of this chapter.

Single-stage methods

Whilst region-based neural networks like R-CNN proved to be highly successful, the machinery involved in prediction with multiple stages was extremely complex, and prior to the work of Dai et al. (2016) training such systems end-to-end proved impossible. A much simpler paradigm was to predict object classes and bounding boxes from deep network features in a sliding-window fashion. Such an approach is inherently efficient due to the convolutional nature of the CNN features. This technique had already been widely used in early CNN-based detectors (Matan et al., 1992; Vaillant et al., 1994; Delakis and Garcia, 2008), but perhaps the first modern incarnation was the OverFeat algorithm of Sermanet et al. (2013), which densely classified and regressed bounding boxes at every location of an image. OverFeat however was fairly limited the number and scales of objects that it could detect, since all predictions were generated from a single downsampled feature map. This

problem was overcome by the Single Shot MultiBox Detector (SSD) of Liu et al. (2016), which used a very large number of anchor boxes generated at multiple different scales within the network. SSD was significant in being the first single-stage network to outperform the more widespread proposal-based approaches whilst operating at significantly higher frame rates. Analogously to the addition of a feature pyramid to Faster R-CNN by Lin, Dollár, et al. (2017), Fu, Liu, et al. (2017) improved SSD by upsampling low-resolution features to incorporate greater semantic content. Shen et al. (2017) showed that a similar architecture could be trained completely from scratch without the ubiquitous ImageNet pretraining step using DenseNet-inspired dense connections (Huang, Liu, et al., 2017).

Another well-known example of the single-stage approach is the “You Only Look Once" (YOLO) (Redmon, Divvala, et al., 2016; Redmon and Farhadi, 2017; Redmon and Farhadi, 2018) family of architectures. The original YOLO model of Redmon, Divvala, et al. (2016) resembles the earlier MultiBox algorithm of Erhan et al. (2014), using fully-connected layers to predict a fixed number of bounding boxes for each image. Subsequent iterations replaced these fully-connected layers with convolutional layers as well as additional improvements including a more powerful feature extraction network called DarkNet (Redmon and Farhadi, 2017; Redmon and Farhadi, 2018). YOLO and its derivatives emphasised speed over accuracy, typically achieving below state-of-the-art performance but with the ability to run at speeds of up to 155 frames per second (Redmon, Divvala, et al., 2016).

Despite the development of powerful single-stage networks such as SSD and YOLO, proposal-based methods continued to dominate the field of 2D object detection. Perhaps the strongest challenge to this monopoly was posed by Lin, Goyal, et al. (2017). They suggested that the limitations of single-stage methods were down to the severe class imbalance arising from the fact that the vast majority of image locations do not contain an object. Lin, Goyal, et al. overcame this with a novel focal loss function, and their single-stage architecture RetinaNet significantly outperformed existing proposal-based methods and offered a better speed-accuracy trade-off. The impressive performance of RetinaNet, combined with the simplicity and efficiency of the single-stage approach, provided considerable motivation for the OFTNet architecture discussed in Chapter 3.

2.1.2 3D Object Detection from LiDAR

In autonomous driving, simply knowing the 2D size and location of an object in a image is typically insufficient. In order to make decisions about where to move next, an autonomous vehicle must know the approximate position, dimensions, and direction of travel of other dynamic objects in the scene. The 3D object detection task is a challenging variant of the traditional object detection problem, where it is necessary to obtain a tightly-fitting 3D cuboid

to every object in the scene. Given the severity of the problem, the vast majority of prior works have focused on the case where accurate depth measurements are available using LiDAR or other range sensors. Various different approaches are available.

Perhaps the most conceptually straightforward approach is to apply machine learning techniques directly to the 3D point clouds produced by LiDAR sensors. Point clouds have traditionally been viewed as a difficult modality to work with given their lack of structure and inherent ordering. The pioneering PointNet network of Qi, Su, et al. (2017) showed that direct processing on point clouds could be achieved through the use of simple symmetric operations such as max pooling. Several 3D object detection algorithms adopt this approach, most notably the Frustrum PointNet (Qi, Liu, et al., 2018), which applies a PointNet network to the subset of points which project to a previously detected 2D bounding box. Despite the successes of PointNet-based architectures however, this class of architectures is still relatively immature compared to extensively-studied convolutional neural networks which have enjoyed considerable success on more structured inputs. Consequently most algorithms which accept point clouds as inputs directlty, such as PointPillars (Lang et al., 2019) and VoxelNet (Zhou and Tuzel, 2018), use this approach only as a preprocessing step, before converting to a more structured representation.

The simplest way to represent a point cloud in a more structured fashion is to discretise the world into a regular grid of 3D regions called voxels. A voxel is marked as occupied if it contains one or more points in the point cloud, empty otherwise. Works to adopt this representation include Voting for Voting (Wang and Posner, 2015), Vote3Deep (Engelcke et al., 2017), SECOND (Yan, Mao, et al., 2018) and 3D FCN (Li, 2017; Li, Zhang, et al., 2016). These approaches are however limited by the extreme computational and memory requirements of voxel grids, which scales cubicly with the size and resolution of the grids. Engelcke et al. (2017) and Yan, Mao, et al. (2018) sought to overcome this limitation by employing sparse 3D convolutions within a hierarchical voting scheme, exploiting the fact that the vast majority of voxels are empty. Given a fixed convolutional kernel size however, the sparsity of the voxel grid decreases as the number of convolutions increases, making this approach unsuitable in modern ultra-deep neural networks.

Most works therefore opt to project the 3D point cloud to a more efficient '2.5D' representation, which preserves the 3D information about the scene but can be processed much more efficiently using standard convolutional networks. One natural approach is to construct an image where the horizontal and vertical pixel coordinates represent the azimuth and elevation angles respectively of each LiDAR ray, and the greyscale colour value represents the distance of the corresponding point from the camera (Li, Zhang, et al., 2016; Minemura et al., 2018). While this closely reflects the way LiDAR point clouds are generated, performing

convolutions in this space ignores much of the underlying 3D structure, since points which are close in the range image need not be close in 3D (You et al., 2019).

As a result, the most popular representation for 3D object detection from LiDAR methods is an orthographic projection of the point cloud onto a birds-eye view plane (Yu, Westfechtel, et al., 2017; Yang, Luo, et al., 2018; Beltrán et al., 2018; Chen, Ma, et al., 2017; Ku, Mozifian, et al., 2018; Wirges et al., 2018). In this representation, each location on the birds-eye view image corresponds to a vertical column of points in 3D space. To retain vertical information, the colour channels of the birds-eye view image encode information such as the maximum, minimum and mean height of points within each column (Yu, Westfechtel, et al., 2017); average reflection intensity (Yang, Luo, et al., 2018; Beltrán et al., 2018); point density (Chen, Ma, et al., 2017; Ku, Mozifian, et al., 2018; Wirges et al., 2018); and others. An alternative interpretation on the birds-eye view approach, adopted by the “Fast and Furious" network of Luo et al. (2018), is to construct a full 3D voxel grid as described above, but then treat the vertical dimensions as the colour channels of a 2D image. This allowed Luo et al. to process the voxel grid as efficiently as if it were an image, without losing vertical information. The success of the above works illustrate the effectiveness of the birds-eye view representation, which forms the central motivation for this thesis.

2.1.3 Image-based 3D Object Detection

Localising 3D objects from images represents a considerably more challenging task, since unlike dedicated range sensors like LiDAR and RADAR, images do not provide explicit measurements of distance. Approximations to a LiDAR-like point cloud representation can be obtained using stereo imaging, where the pixel intensities of a pair of rectified images are compared to obtain disparity estimates for each pixel (Szeliski, 2010, Chp. 12). This technique formed the basis of the 3DOP algorithm of Chen, Kundu, Zhu, et al. (2015), which spans the 3D space with hypothesised bounding box candidates, then scores each proposal based on the parts of the stereo point cloud which falls within each box. More recent work by Li, Chen, et al. (2019) applies a Faster-R-CNN-like (Ren et al., 2015) framework directly to the pair of stereo images, while You et al. (2019) run LiDAR-based object detectors such as AVOD (Ku, Mozifian, et al., 2018) and MV3D (Chen, Ma, et al., 2017) on the stereo point clouds, noting special dispensations that should be made to the underlying stereo algorithm to facilitate this application.

In many scenarios, paired stereo imagery may not be available, for example in the surround-view ring cameras mounted on many recent autonomous platforms (Caesar, Bankiti, et al., 2019; Chang et al., 2019; Sun, Kretzschmar, et al., 2020). Computing stereo disparity is also computationally expensive, and may stretch computational resources. As a result,

predicting 3D bounding boxes directly from monocular images remains an important research goal.

The main challenge behind monocular 3D object detection, however, is the lack of explicit 3D information. To overcome this, several works have taken advantage of constraints imposed by the projective camera geometry to lift 2D bounding boxes to 3D. For example, Mousavian et al. (2017) note that the projection of an object's 3D bounding box should fit tightly within the bounds of the corresponding 2D detection. Starting from pre-detected 2D windows, they train a CNN to regress the dimensions and orientation of each object and then solve the system of linear constraints to infer the 3D coordinates which best match the 2D window. This approach is also followed by Gustafsson and Linder-Norén (2018), who adopt a slightly different parametrisation for the 3D boxes. Payen de La Garanderie et al. (2018) meanwhile adapt the geometric constraints of Mousavian et al. for panoramic images. Both Gustafsson et al. and Payen de La Garanderie et al. employ style transfer techniques (Zhu, Park, et al., 2017) from synthetic images to expand the available training data.

In addition to constraints provided by 2D bounding boxes, assumptions about the structure of the world can help resolve the 3D positions of objects. Novak (2017) for example assumes all objects are constrained to a fixed ground plane, and uses homography constraints to infer the distance of the bounding box vertices from the camera. The Mono3D algorithm of Chen, Kundu, Zhang, et al. (2016) meanwhile uses assumptions about the ground plane to generate a large number of 3D bounding boxes proposals which densely span the visible 3D space. Each candidate box is projected into the image using known camera parameters, and is scored against a set of image-based metrics to eliminate boxes which are unlikely to contain objects. Mono3D provides the main baseline which is used in Chapter 3.

A number of authors have argued that the 3D bounding box representation used by the above methods is overly simplistic and ignores the underlying 3D geometry of the scene, and instead advocate explicitly modelling the 3D shape of objects (Zeeshan Zia et al., 2014; Chabot et al., 2017). Zeeshan Zia et al. (2013) propose to represent cars and other vehicles as a simple wireframe mesh, which forms the basis of their deformable parts model. Zeeshan Zia et al. (2015) extend this model further, incorporating joint reasoning about all objects in the scene. A similar methodology is adopted by the Deep MANTA architecture of Chabot et al. (2017), which uses an iterative CNN to predict a set of 2D keypoints corresponding to vertices on an annotated template mesh. A 3D-2D template fitting procedure is then used to lift the 2D keypoints to 3D. Kundu et al. (2018) develop a fully parametric PCA shape model of objects such as cars and motorcyles, and use a differentiable rendering process to fit the models to 2D silhouettes. Xiang et al. (2015) by contrast employ a non-parametric approach, representing objects as 3D voxel occupancy grids. They then train a detector

to recognise patterns of voxels which have similar appearance, taking into account factors such as occlusion and clipping due to the camera's field of view. While the above methods represent a powerful way of obtaining not just information about an objects' pose, but also its shape; they are typically too slow for real-time applications, and rely on annotated databases of 3D CAD models which limits their use to a few well-defined object categories.

2.2 Maps for autonomous driving

2.2.1 Map representations

Maps for use in robotics and autonomous driving may take a variety of different forms. The IEEE Standard for Robotic Map Data Representation for Navigation defines a broad hierarchy for classifying different types of map used for robotics (Yu and Amigoni, 2014). At the top level of the hierarchy, maps may be subdivided into two broad categories: metric maps and topological maps. Metric maps aim to capture accurate geographical information about the scene, including the locations of landmarks and obstacles. Topological maps, on the other hand, describe the connectivity between locations, representing the environment as a graph of places and the relationships between those places (Kuipers, 1977; Kuipers and Byun, 1991). Topological maps are most commonly used for high-level navigation and route planning, such as planning a path between cities, navigating between rooms in a complex building, or choosing lanes at an intersection (Qiao et al., 2019). Many works have shown how topological maps can be built on the fly, either directly from sensor readings and robot odometry (Shatkay and Kaelbling, 1997), or as a post-processing step applied to a pre-computed metric map (Thrun and Bücken, 1996a; Thrun and Bücken, 1996b). Another common paradigm is to combine metric and topological maps: using topological maps for high-level navigation and localisation and metric maps for lower-level path planning and obstacle avoidance. Perhaps the most notable example of this strategy is the Spatial Semantic Hierarchy of Kuipers (2000) as well as work by Chatila and Laumond (1985).

In this thesis, we are predominantly interested in the metric mapping paradigm. The definition of a metric map is a map where, for any pair of elements (such as points, grid cells or features) a and b, a given metric distance metric $d(a,b)$ can be evaluated. This makes metric mapping more applicable in the context of road scene understanding, obstacle avoidance and short-term trajectory planning. Within this subcategory, the IEEE Standard (Yu and Amigoni, 2014) makes a further distinction between continuous and discrete metric map representations. Continuous metric maps are composed of collections of basic geometric primitives. Popular representations include points (Leonard and Durrant-Whyte, 1991a),

lines (Ayache and Faugeras, 1989), polygons (Chatila and Laumond, 1985) and cylinders (Tao et al., 2010). The point-based representation is particularly common in visual mapping techniques such as visual simultaneous localisation and mapping (SLAM) and structure from motion (see Section 2.2.2), where map landmarks often correspond to corners or point-features in an image (Davison, 2003). Continuous representations have the advantage of being compact since they typically represent a sparse description of the environment. They are also well suited to the application of probabilistic techniques such as Kalman filtering (Kalman, 1960) and particle filters (Del Moral, 1997). However, applying motion planning techniques to continuous metric maps directly is often challenging, and further post-processing is usually required.

The second type of metric map representation, and the one we focus on in this thesis, are discrete metric maps. By far the most common discrete metric map representation in robotics is the *occupancy grid map*, first proposed by Moravec and Elfes (1985) in 1985. Occupancy grid maps divide the world into regular 2D grid cells or 3D voxels (Martin and Moravec, 1996) with a fixed resolution. Each grid cell is associated with a value indicating the probability that an obstacle occupies the cell. This provides a natural probabilistic framework, formalised by Moravec (1989), for accumulating map features over time and combining maps computed by different sensors (Nuss, Thom, et al., 2014). This approach is more robust to sensor noise than many of the geometric mapping approaches described above (Elfes, 1989). Occupancy grids also are simple to incorporate into a motion planner. For example, Elfes et al. (1990) described how an A* path planning algorithm could be applied directly to an occupancy grid representation, with occupancy probabilities representing the risk of entering a given grid cell.

One of the drawbacks of the occupancy grid framework is that it typically assumes the world is static and cannot easily incorporate information about dynamic objects such as cars and vehicles. This has led to the development of dynamic occupancy grid maps (DOGMa) algorithms, including those of Nuss, Reuter, et al. (2018), Gindele et al. (2009), and Danescu et al. (2011).

2.2.2 Simultaneous localisation and mapping

Perhaps the most widely-studied application of map-making in robotics concerns localisation. For a robot to keep track of its location in an unseen environment, it needs to maintain some representation of the places it has already visited. This symbiosis between localisation and map creation has led to the emergence of a major sub-field of robotics called *Simultaneous Localisation and Mapping* (SLAM). The term SLAM was first coined by Leonard and Durrant-Whyte in 1991; however Thrun (2002) traced the field back to the work of Carl

Friedrich Gauss in mapping the orbits of celestial bodies (Gauss, 1809). The first modern incarnation of the SLAM problem was proposed in a series of papers by Smith, Self and Cheeseman, which used Extended Kalman Filters (EKF) (Daum, 2015) to jointly estimate the ego-pose and positions of a set of landmark points in the environment under uncertainty (Smith, Self, et al., 1986; Smith and Cheeseman, 1986; Smith, Self, et al., 1990). This approach builds a map incrementally as the vehicle travels through the environment. Other notable developments in simultaneous localisation and mapping include Graph-SLAM (Lu and Milios, 1997), which solved the offline mapping problem; the sparse extended information filter (SEIF) (Thrun, Liu, et al., 2004) which adapted Graph-SLAM to the online mapping case; and FAST-SLAM (Montemerlo, 2003), which improved the scalability of EKF-SLAM using particle filters (Smith, 2013). The above approaches were general and could be applied to any number of different sensor modalities. In this thesis, we are principally concerned with mapping from monocular cameras, for which many specialised algorithms have been proposed (Davison, 2003; Mur-Artal and Tardós, 2015; Engel et al., 2014; Mur-Artal, Montiel, et al., 2015). Monocular SLAM is also closely related to the problems of structure from motion (SfM) (Tomasi and Kanade, 1992; Hartley and Zisserman, 2003) and bundle adjustment (Triggs et al., 1999) in computer vision. These approaches all rely on sequences of monocular images in order to triangulate feature points across time. One of the attractive features of the mapping approach described in this thesis is that structure could be inferred from just a single image, although this initial prediction could be refined with information from subsequent frames.

Most of the above SLAM approaches have focused on building geometric maps of the scene, either in the form of simple geometric primitives (Moutarlier and Chatila, 1990; Leonard and Durrant-Whyte, 1991b), 3D point clouds (Davison, 2003) and occupancy grids (Elfes, 1989). As the field of SLAM has matured, there has been a growing interest in more semantic map representations. In early works, semantic information was principally used to identify dynamic objects, which can cause traditional SLAM algorithms to fail. For example, Wang, Thorpe, et al. (2003) proposed SLAM with detection and tracking of moving objects (DATMO). Meanwhile, Wolf and Sukhatme (2005) maintained separate occupancy grid maps for dynamic and static objects, an approach which we adopt in Chapter 4. Similarly, Vineet et al. (2015) built a semantic mesh representation of a road scene and used the semantic labels to subtract dynamic instances. As well as aiding with removing moving objects, various works have shown that incorporating semantic information into SLAM and structure from motion can improve the accuracy of the produced map (Bao and Savarese, 2011; Yu, Liu, et al., 2018). In particular Salas-Moreno et al. (2013) and Yang and Scherer (2019) showed that applying SLAM at the level of semantically-meaningful objects, rather than simple

features, was able to improve localisation in repetitive environments, as well as producing richer and more detailed maps. Furthermore, many works have combined SLAM with semantic segmentation to build complete 3D semantic reconstructions of the environment, for example works by Hermans et al. (2014), Hane et al. (2013), Koppula et al. (2011), Pham et al. (2015), and Valentin et al. (2013). These works often adopted a conditional random field-based (Lafferty et al., 2001; Krähenbühl and Koltun, 2011) refinement step to jointly optimise semantic labels and 3D structure.

Unlike many of the works described above, our mapping approach in Chapter 4 does not treat localisation as a primary objective. However, the semantic map representation we describe could be used to localise the ego-vehicle within a global coordinate system using a technique called map-matching, which registers a locally predicted map against a larger global map. Several algorithms have been proposed, including those which used hill-climbing algorithms (Simmons et al., 2000), iterative closest point (ICP) search (Bosse and Zlot, 2008) and Gaussian Markov random fields (Thrun and Liu, 2005).

2.2.3 Mapping for motion planning and prediction

Even since the early days of mobile robotics, map-like representations have been essential for allowing agents to plan and execute trajectories through a real-world environment. One of the first robots to maintain a simple model of its environment was 'Shakey the robot' developed at Stanford between 1966 and 1972 (Nilsson, 1984). Its ability to detect and track simple primitives like lines in the environment was the basis for the development of the A* algorithm by Hart et al. (1968), one of the most significant advancements in robotic motion planning. The later 'Stanford Cart' platform was in 1980 augmented with the ability to track and avoid point-based objects (Moravec, 1980), although the robot would have to stop for 10-minute intervals to process a single map update (Moravec, 1979). In keeping with these early works in robotics, much research within autonomous driving has focused on mapping the free space in front of the vehicle using a variety of range sensors, including laser scanners (Kirchner and Ameling, 2000), RADAR (Lundquist et al., 2009), LiDAR (Silver et al., 2010; Chen, Zhang, et al., 2019), stereo cameras (Hadsell et al., 2009; Sahdev, 2017) and monocular videos (Yao et al., 2015). Many of these works, for example, those of Badino et al. (2007) and Lundquist et al. (2009), adopted the Bayesian occupancy grid framework, which is the basis for our mapping approach described in Chapter 4. For driving in unstructured environments, simple geometric maps of free and occupied space are sufficient. However, for navigating in urban scenarios, a greater semantic understanding of the scene is necessary, for example to distinguish between road and pavements. This has led to a plethora of machine learning-based approaches which focus on road segmentation

and lane boundary detection (Hillel et al., 2014; Fritsch et al., 2013; Lee, Kim, et al., 2017; Pan, Shi, et al., 2017; Teichmann et al., 2018). Often this analysis is performed in the image space, leveraging modern advancements in semantic segmentation (Long et al., 2015; Badrinarayanan et al., 2017), and then transformed into a birds-eye view map using inverse perspective mapping as a post-processing step (Patra et al., 2018). This approach suffers from the fact that small segmentation errors in the pixel space can lead to dramatically incorrect predictions in the birds-eye view space. One of the main aspirations of the work in Chapter 4 therefore, was to be able to reason semantically about the drivable road surface directly in the birds-eye view.

In a conventional autonomous driving pipeline, these maps of free or drivable space derived from one of the methods described above would then be fed into a motion planning algorithm such as probabilistic roadmaps (Kavraki et al., 1996) or rapidly-exploring random trees (LaValle, 1998; LaValle and Kuffner Jr, 2001). Recently however, Bansal et al. (2018) generated considerable excitement in the field of motion planning with their algorithm, ChauffeurNet; an imitation learning-based method able to learn a driving policy directly from a rasterised semantic map-like input in the birds-eye view space. Their semantic map representation included features such as the road lane layout, prior driving history and positions of other dynamic agents. They showed that this was sufficient to learn to handle complex driving scenarios such as nudging around parked cars and navigating difficult intersections. In a similar vein, Hecker et al. (2018) combined camera images with a birds-eye view map to directly predict steering commands. However, neither of these works addressed the problem of how to generate such representations on the fly, strongly motivating the work described in Chapter 4.

Closely related to the problem of motion planning is that of motion prediction or motion forecasting, i.e. estimating the future trajectories of other non-ego agents in the scene. Traditional approaches relied purely on extrapolation of observed trajectories (Wan and Van Der Merwe, 2000). However, recent works have shown that incorporating additional context, such as road layout and interaction with other agents, can considerably reduce long-term uncertainty (Lee, Choi, et al., 2017). Luo et al. (2018) predict future trajectories jointly with detecting and tracking by applying a deep CNN to a LiDAR occupancy grid. Djuric et al. (2018), Cui et al. (2019) and Casas et al. (2018) meanwhile adopted rasterised semantic maps similar to those described above to predict future agent motion. Bansal et al. (2018) also extended their ChauffeurNet framework to the problem of multi-agent prediction. Casas et al. in particular noted that the inclusion of a semantic map representation significantly improved prediction quality above simply using raw input data.

2.2.4 Mapping and deep learning

As with almost every other topic in robotics and vision, in recent years the field of mapping has been revolutionised by advances in machine learning and computer vision (Krizhevsky et al., 2012). Among the technologies to have experienced the biggest impact from the introduction of deep learning methods have been geographic information systems (GIS) and remote sensing (RS), which generate birds-eye view maps from offline data (Ma et al., 2019). A common approach is to apply standard semantic segmentation networks such as U-Net (Ronneberger et al., 2015) or SegNet (Badrinarayanan et al., 2017) to satellite or aerial images to build a semantic map. For example, works including those of Chen, Lin, et al. (2014) and Scott et al. (2017) have used this approach to classify the type of vegetation, soil type or artificial surface at each pixel. Maggiori et al. (2016) and Alshehhi et al. (2017) focused on segmenting artificial structures such as buildings and roads from satellite images, while Kampffmeyer et al. (2016) extended these maps to small objects such as individual cars in aerial images. Máttyus, Luo, et al. (2017) segmented roads from images and then post-processed them to obtain topological road networks that could be used for high-level path planning, while Liang et al. (2019) predicted vectorised road boundaries directly using recurrent neural networks. Máttyus, Wang, et al. (2016) proposed a method to align semantic road labels predicted in both aerial and road-level views. The maps produced by these methods are highly relevant to autonomous driving and the work described in Chapter 4. However, the fact that they were generated in an offline setting makes them insufficient for full self-driving by themselves, as they could not capture real-time information such as the positions of individual cars, nor could they track medium-term changes in the environment such as roadworks or obstacles. Operating entirely in the birds-eye view (starting from satellite or aerial imagery) made them simpler to implement than the networks considered in this thesis, which must transform between a ground-level view and birds-eye view perspective.

Most relevant to the work in Chapter 4, a small number of authors have in recent years attempted to tackle the same problem of predicting semantic birds-eye view maps directly from street-level images. Early work in this field by Sengupta et al. (2012) used the TexonBoost algorithm (Shotton et al., 2006) to perform semantic segmentation in the image space and transformed the resulting labels to the birds-eye view using inverse perspective mapping. A conditional random field model (Ladick et al., 2009) was used to construct a global semantic map representation in the birds-eye view space. Mozos et al. (2007) and Sünderhauf et al. (2016) built semantic occupancy grid maps of indoor scenes by combining image-level classification with traditional occupancy grid mapping, although such maps can only capture large-scale descriptions such as room type rather than fine details such

as individual objects. Deng et al. (2019) used inverse perspective mapping to transform image-based segmentations to the birds-eye view, but this approach broke down for points far away from the cameras. The lack of supervised training data has severely hampered other works which try to produce semantic maps of outdoor traffic scenes. Schulter et al. (2018) used an in-painting approach to learn pixel-wise depth and semantic labels for the road structure behind occluding objects such as cars and pedestrians. A network trained using a complex combination of simulated data, generative adversarial networks (Goodfellow et al., 2014) and weakly-aligned annotations from OpenStreetMaps (OpenStreetMap contributors, 2017) was then used to refine an initial birds-eye view map produced by back-projecting the predicted pixel depths and labels into 3D. Wang, Liu, et al. (2019) adopted a similar approach to that of Schulter et al., but their final output representation was a parametric road model consisting of human-interpretable attributes such as 'number of lanes' or 'existence of sidewalks'. Pan, Sun, et al. (2020) and Reiher et al. (2020) trained birds-eye view map prediction networks entirely on simulated data, using image-based semantic segmentation as an intermediate representation to allow their work to generalise to the real world. Reiher et al. used an inverse perspective mapping approach to transform features from the image to birds-eye view domain, while Pan, Sun, et al. introduced a 'view transformer model' to invert the perspective projection. Prior to work discussed in Chapter 4, the only previous approach to train their method directly on real monocular images was that of Lu, Molengraft, et al. (2019), who proposed an encoder-decoder-style network inspired by variational autoencoders. However, the ground truth semantic map annotations used to train this approach were only approximate, derived from noisy stereo depth estimates and image-based semantic labels. A transformative moment in the state of semantic mapping from images was the release of several large scale 3D object datasets with accompanying semantic map annotations, most notably the NuScenes dataset (Caesar, Bankiti, et al., 2019) and the Argoverse dataset (Chang et al., 2019). Alongside enabling the research described in this thesis, these datasets have been used in other more recent works to provide ground truth semantic map annotations. These include the FISHINGNet model of Hendy et al. (2020), which predicted the future positions of vehicles in a semantic occupancy grid-like representation; and Lift, Splat and Shoot by Philion and Fidler (2020) which predicted birds-eye view semantic maps and vehicle trajectories, adopting many of the ideas described in this thesis.

Chapter 3

Monocular 3D Object Detection

3.1 Introduction

What sets autonomous vehicles apart from many other forms of robotics is the hugely dynamic environment in which they operate. Cars, buses, pedestrians, pets, bicycles, motorcycles: exhibiting complex interactions and behaviours; all represent obstacles which must be avoided at all costs. To succeed, an autonomous vehicle must be capable of detecting these objects, identifying them, estimating their position in 3D space, and tracking their movements over time.

Addressing this problem has been the primary objective of a major sub-field of computer vision called object detection. The aim of a traditional 2D object detection algorithm is to return a list of all objects present in an image, to classify them into one of a number of semantic categories, and to accurately localise them within a tighly-fitting, axis-aligned 2D spatial window called a bounding box. The problem of 3D object detection meanwhile represents an additional level of challenge. In this case the aim is not only to determine the location of each object within the image, but to estimate the parameters of a bounding cuboid in 3D space. This corresponds to solving the full 6 degree of freedom pose estimation problem for each object i.e. finding its 3D translation (X, Y, Z) and rotation (α, β, θ); as well as determining the spatial dimensions (w, h, l) of its bounding box.

This chapter will introduce a deep-learning-based solution to the problem of 3D object detection from monocular images alone. The core component of this method is a convolutional neural network called the OFTNet, which is unique in that it applies convolutional processing to feature maps in both the perspective image space and the orthographic birds-eye view space. The rationale behind this design choice was that reasoning in the birds-eye view space allows the neural network to learn about the fundamental metric nature of 3D space and the spatial relationships between objects. Obtaining a birds-eye view feature representation

directly from an image is non-trivial. The OFTNet architecture was therefore built around a novel neural network module called the orthographic feature transform (OFT), which maps image-based features into the birds-eye view.

The remainder of this chapter is structured as follows: Section 3.3 introduces the novel orthographic feature transform component, which is then integrated into the wider OFTNet architecture which is discussed in Section 3.3. In Section 3.5 the OFTNet network is applied to a real-world 3D object detection dataset: the KITTI object detection benchmark. The performance of the OFTNet is evaluated both against contemporary state-of-the-art methods as well as more recent approaches, and ablation studies are used to gain insight into the various aspects of the approach. Finally, Section 3.6 discusses further analysis of the various failure modes of OFTNet and other related works. Concluding remarks are provided in Section 3.7.

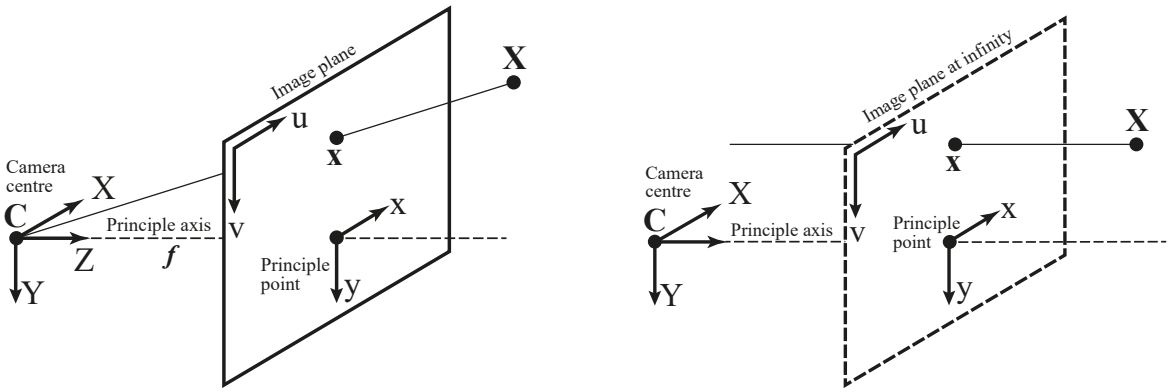
3.2 Preliminaries

At the heart of the work discussed in both in this chapter as well as the following chapter is the idea of using a combination of machine learning and known camera geometry to transform features from the image space into the birds-eye view. This idea is predicated on the mapping from a 3D point to a 2D image which is known as a *camera model*. We provide a brief overview of the fundamentals of the types of camera models encountered in this thesis below.

3.2.1 Perspective camera model

The most common form of camera model is the pinhole camera, which is used to describe perspective projection of a 3D point onto the image plane (Hartley and Zisserman, 2003). Consider a plane (known as the image plane) which is placed a distance f along the positive Z-axis away from the centre of projection $\mathbf{C}$, which we initially assume is the same as the origin of the coordinate system. The perspective projection of a 3D point $\mathbf{X} = (X, Y, Z)^T$ is given by the intersection of the ray passing through $\mathbf{C}$ and $\mathbf{X}$ with the image plane, as shown in Figure 3.1a. We can easily determine the projected coordinates $\mathbf{x} = (x, y)^T$ of the point X by considering similar triangles:

$$x = fX/Z \qquad\qquad y = fY/Z \tag{3.1}$$



(a) Perspective projection (b) Orthographic projection

Figure 3.1. The two types of camera models discussed in this thesis.

We can represent this transformation efficiently using the multiplication of the point $\mathbf{X}$ expressed in homogeneous coordinates $\tilde{\mathbf{X}} = (X, Y, Z, 1)$ with a 3×4 projection matrix P:

$$\tilde{\mathbf{x}} = w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P\tilde{\mathbf{X}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

where $\mathbf{x} = \tilde{\mathbf{x}}/w$. The coordinates $\mathbf{x}$ represent the projection of the point $\mathbf{X}$ onto the image plane in meters. To determine the location of a point on a digital image in pixels, we need to consider the (possibly non-square) size (k_u, k_v) and shear s of the pixels as well as the position of the principle point (the projection of the camera centre) in the image coordinate system (c_u, c_v). The pixel coordinates $\mathbf{u} = (u, v)^T$ of the projection of point $\mathbf{X}$ are therefore given by

$$\tilde{\mathbf{u}} = w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & s & c_u \\ 0 & k_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.3}$$

where the matrix K is known as the camera *intrinsic* matrix or *intrinsic calibration* matrix, and is given by

$$K = \begin{bmatrix} \alpha_u & s' & c_u \\ 0 & \alpha_v & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

where $\alpha_{\{u,v\}} = fk_{\{u,v\}}$ are the scaled horizontal and vertical focal lengths and $s' = fs$ the shear, which are measured in pixels. Note that throughout this thesis, we assume that the shear s' is negligible and that the horizontal and vertical scaled focal lengths were equal $\alpha_u = \alpha_v = \alpha$. This assumption reflected the observed intrinsic calibration parameters provided by the KITTI (Geiger, Lenz, and Urtasun, 2012), NuScenes (Caesar, Bankiti, et al., 2019) and Argoverse (Chang et al., 2019) datasets used in this thesis.

3.2.2 Orthographic camera model

An important element of our work is the orthographic projection of 3D points onto a birds-eye view plane. Like the perspective projection discussed above, this transformation can be represented as a multiplication with a 3×4 matrix P, which has the form

$$\tilde{\mathbf{x}} = w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P\tilde{\mathbf{X}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

Frequently, we will need to refer to birds-eye view coordinates using a discrete grid-based coordinate system. Analogously to mapping to pixel coordinates as discussed above, we can map world points to grid coordinates by pre-multiplying P using a 3×3 affine transformation matrix K_{ortho}, given by

$$K_{ortho} = \begin{bmatrix} \rho_u & s & x_0 \\ 0 & \rho_v & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\rho_{\{u,v\}}$ represents the resolution of the grid (measured in m^{-1}), s represents the shear, and $(x_0, v_0)^T$ is the projection of the origin in the orthographic coordinate system.

3.2.3 Coordinate system transformations

Thus far we have assumed that the camera centre $\mathbf{C}$ for the perspective and orthographic camera models coincides with the origin of the coordinate system, and that the cameras are orientated along the positive Z axis. In general, however, this need not be the case, and it is necessary to transform the 3D point $\mathbf{X}$ into the camera's local coordinate system. A rigid-body transformation from one coordinate system A to a second coordinate system B is

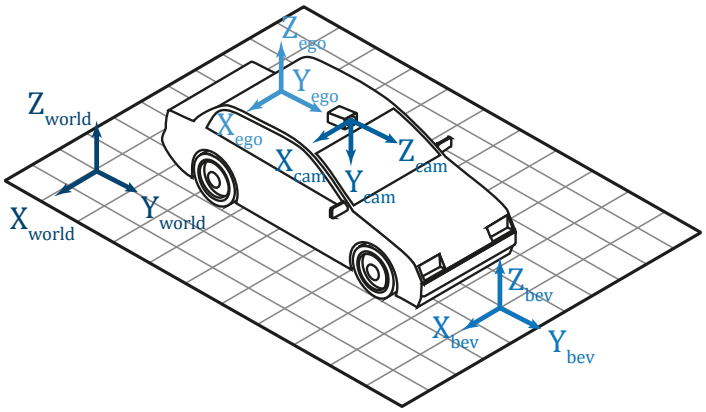


Figure 3.2. Overview of the four main coordinate systems considered in this thesis.

achieved through multiplication with a 4×4 transform matrix

$$T_{A\to B} = \left[\begin{array}{c|c} R & \mathbf{t} \\ \hline \mathbf{0}^T & 1 \end{array}\right]$$

which is composed of a 3×3 orthonormal rotation matrix R and a translation vector $\mathbf{t}$. In the context of camera models, the transform matrix $T_{A\to B}$ which maps from the world coordinate system to the camera coordinate system is commonly referred to as the *extrinsic* camera matrix.

In this thesis we consider four main coordinate systems, which are illustrated in Figure 3.2. These are:

World A fixed coordinate system in space which all other coordinate systems are defined relative to. The Z axis points vertically upwards and the X and Y axes correspond to longitude and latitude respectively.

Ego-vehicle A coordinate system which moves with the vehicle of interest. For the NuScenes (Caesar, Bankiti, et al., 2019) and Argoverse (Chang et al., 2019) datasets, this is positioned at the location of the onboard inertial measurment unit (IMU). The Z axis points upwards and the Y axis points in the direction of travel.

Camera The local coordinate system of a camera sensor. The coordinate system origin is chosen to be the optical centre, and the Z direction is aligned with the camera's principle axis. A given vehicle may employ multiple cameras, in which case a unique coordinate system is defined for each.

Birds-eye view The birds-eye view coordinate system is a rotation of the camera coordinate system, such that the Z-axis is aligned with the cameras negative Y direction, and is placed a fixed distance below the camera. Note that the birds-eye view coordinate system does not necessarily correspond to the ground plane, although is a reasonable approximation to it for moderate pitch and roll angles of the ego-vehicle.

3.3 Orthographic Feature Transform

The central philosophy of this thesis is the idea that allowing a neural network to reason about objects in a metric 3D space is vital to scene understanding tasks such as 3D object detection. Unfortunately, since the input to our object detection system was a monocular camera image taken from a street-level view perspective, reasoning in the birds-eye view directly was impossible. Therefore, in order to overcome this hurdle, we introduced a novel neural network component called the *orthographic feature transform* (OFT). The purpose of the OFT was to take a representation of the world in image space and transform it to a new representation in the orthographic birds-eye view space, making use of the underlying camera geometry. This component formed the core novelty of the OFTNet object detection architecture which is described in the subsequent section.

The orthographic feature transform consisted of three main steps:

1. Construct a voxel grid in the 3D world space.
2. Assign a feature vector to each voxel by accumulating features over its projected area in the image space.
3. Combine features along the vertical dimension to obtain a birds-eye view feature map.

An overview of the OFT algorithm is shown in Figure 3.3.

Voxel grid construction

The OFT's voxel grid is a regular 3D lattice $\Lambda \subset \mathbb{R}^3$ defined in the camera coordinate system, so that the x-axis points to the left of the camera, y-axis points vertically down and the z-axis is parallel to the optical axis. Each voxel $v_{ijk} \in \Lambda$ is a 3D cube of size ρ which is centred on a location $\mathbf{X}_{ijk} = (X_i, Y_j, Z_k)^T \in \Lambda$. The grid spans a cuboidal region of space defined by the Cartesian product $\Lambda = \{X_{min}, X_{min}+\rho, \ldots, X_{max}\} \times \{Y_{min}, Y_{min}+\rho, \ldots, Y_{max}\} \times \{Z_{min}, Z_{min}+\rho, \ldots, Z_{max}\}$, where $\{X,Y,Z\}_{min}$ and $\{X,Y,Z\}_{max}$ are the minimum and maximum bounds of the voxel grid region respectively.

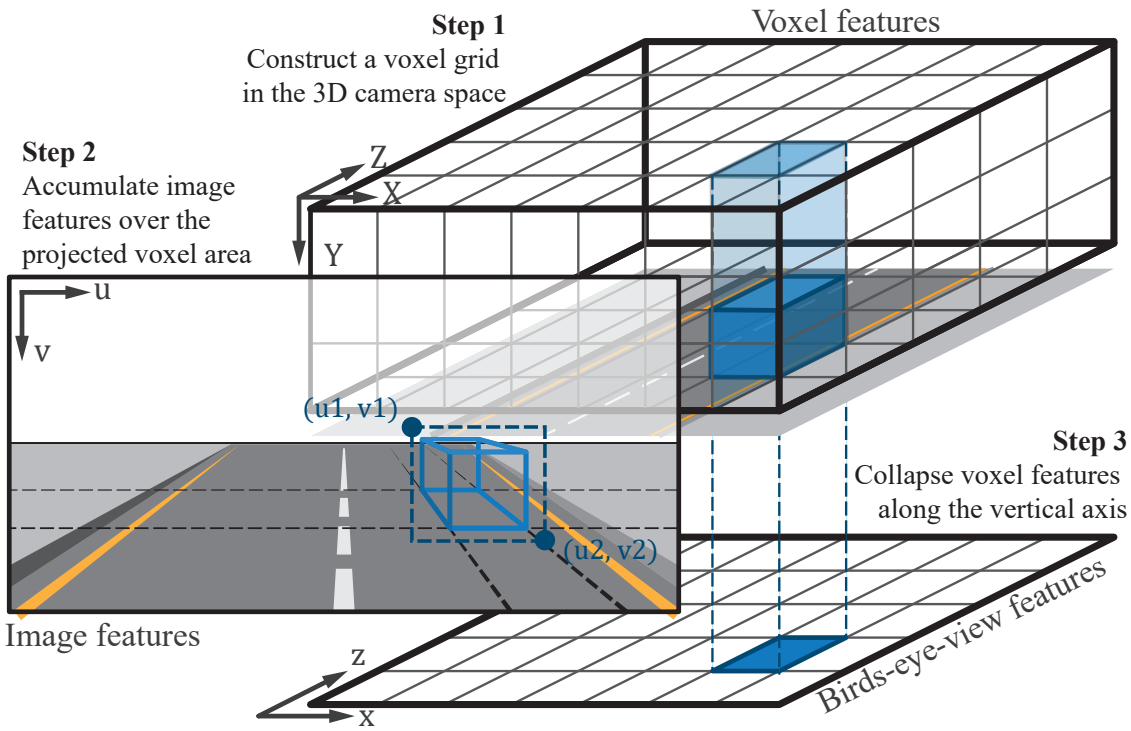


Figure 3.3. Overview of the Orthographic Feature Transform (OFT). The transform consists of three steps: (1) A voxel grid is constructed which spans the 3D space of interest. (2) The voxel feature $\mathbf{f}_{ijk}^{vox}$ is computed by projecting the voxel into the image, and accumulating image features over the region defined by bounding box (u_1, v_1, u_2, v_2). (3) birds-eye view feature $\mathbf{f}_{pq}^{bev}$ is computed by accumulating voxel features along the vertical axis.

Voxel feature computation

The primary aim of the OFT is to associate with each voxel v_{ijk} a feature vector $\mathbf{f}_{ijk}^{vox}$. The voxel feature $\mathbf{f}_{ijk}^{vox}$ is obtained by accumulating image features over a relevant region of the image which corresponds to the given voxel. For each voxel, we define a region Ω which corresponds to the projection of the voxel onto the image plane. In general, this region will be a polygon defined by the convex hull of the projection of the 8 corners of the voxel into the image:

$$\Omega_{ijk} = Hull\left(\begin{bmatrix}\alpha & 0 & c_u\\ 0 & \alpha & c_v\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}X_i \pm \frac{\rho}{2}\\ Y_j \pm \frac{\rho}{2}\\ Z_k \pm \frac{\rho}{2}\\ 1\end{bmatrix}\right) \tag{3.5}$$

The parameters α and (c_u, c_v) correspond to the intrinsic calibration parameters introduced in Section 3.2.1, which are assumed to be known for this problem. The voxel feature $\mathbf{f}_{ijk}^{vox}$ is

then given by the mean image feature over the polygonal region Ω_{ijk}:

$$\mathbf{f}_{ijk}^{vox} = \frac{1}{|\Omega_{ijk}|} \sum_{(u,v)\in\Omega} \mathbf{f}^{img}(u,v) \tag{3.6}$$

In practice, computing averages over arbitrary polygons is extremely computationally expensive, so each polygon was approximated by an axis-aligned rectangular bounding box. The top-left and bottom right corners of this box (u_1, v_1) and (u_2, v_2) are given by

$$u_1 = \alpha \frac{X_i - 0.5\rho}{Z_k + 0.5\frac{X_i}{|X_i|}\rho} + c_u, \qquad v_1 = \alpha \frac{Y_j - 0.5\rho}{Z_k + 0.5\frac{Y_j}{|Y_j|}\rho} + c_v, \tag{3.7}$$

$$u_2 = \alpha \frac{X_i + 0.5\rho}{Z_k - 0.5\frac{X_i}{|X_i|}\rho} + c_u, \qquad v_2 = \alpha \frac{Y_j + 0.5\rho}{Z_k - 0.5\frac{Y_j}{|Y_j|}\rho} + c_v \tag{3.8}$$

The voxel feature $\mathbf{f}_i^{vox}$ could then be computed using a simple average pooling operator over the image features:

$$\mathbf{f}_{ijk}^{vox} = \frac{1}{(u_2 - u_1)(v_2 - v_1)} \sum_{u=u_1}^{u_2} \sum_{v=v_1}^{v_2} \mathbf{f}^{img}(u,v) \tag{3.9}$$

Birds-eye view feature computation

The voxel grid feature map F^{vox} in principle encodes a deep feature vector for every 3D location in the scene. However, operating directly in this space is computationally and memory intensive. We therefore collapse the 3D representation down to a more compact 2D birds-eye view feature map. For a given birds-eye-location $\mathbf{x}_{ik} = (X_i, Z_k)$, the corresponding birds-eye view feature vector $\mathbf{f}_{ik}^{bev}$ is given by taking the sum of voxel features along the vertical (Y) axis. A simple summation however discards information about the height of each voxel, which is relevant for example to determine the correct height of each object's bounding box. In order to preserve this height information, we therefore instead concatenate the 3D feature maps along the vertical dimension, and then apply a 2D convolution to reduce the features to a more manageable dimensionality. The output from this final stage is a 2D birds-eye view feature map which encodes a compact 3D representation of the scene.

3.3.1 Fast average pooling using integral images

One of the challenges of the proposed approach was that despite the voxels having a fixed size in 3D, their projections in 2D could vary greatly as a result of the perspective projection. For example, under typical settings, a voxel placed 50m from the camera occupied a

region of the image approximately 6×6 pixels in size, while a voxel just 1m away would cover approximately 300×300 pixels. Directly computing a mean using the expression in Equation 3.9 for every voxel would be infeasible for even moderately-sized voxel grids.

This challenge was overcome through the use of an efficient average pooling operation using integral images. Integral images were popularised by Viola and Jones (2001) in their seminal paper on face detection, and provide a mechanism to compute sums over large numbers of rectangular areas with minimal computation. The value of an integral image $\mathbf{F}$ at a given location (u, v) is defined as the sum of features along the preceding rows and columns of the input feature map $\mathbf{f}$:

$$\mathbf{F}(u,v) = \sum_{u'=1}^{u} \sum_{v'=1}^{v} \mathbf{f}(u,v). \tag{3.10}$$

The integral image can be computed efficiently using the recurrence relation:

$$\mathbf{F}(u,v) = \mathbf{f}(u,v) + \mathbf{F}(u-1,v) + \mathbf{F}(u,v-1) - \mathbf{F}(u-1,v-1) \tag{3.11}$$

Once the integral image has been precomputed, the sum S over any rectangular region defined by coordinates (u_1, v_1) and (u_2, v_2) can be computed efficiently using

$$S = \mathbf{F}(u_1,v_1) + \mathbf{F}(u_2,v_2) - \mathbf{F}(u_1,v_2) - \mathbf{F}(u_2,v_1) \tag{3.12}$$

This means that each voxel feature in Equation 3.9 can be implemented using just four lookups in the image-based integral image:

$$\mathbf{f}_{ijk}^{vox} = \frac{\mathbf{F}(u_1,v_1) + \mathbf{F}(u_2,v_2) - \mathbf{F}(u_1,v_2) - \mathbf{F}(u_2,v_1)}{(u_2-u_1)(v_2-v_1)} \tag{3.13}$$

Crucially, the computational complexity of Equation 3.13 is $\mathcal{O}(1)$ with respect to the size of the 2D region, which means features corresponding to voxels which are very close to the camera can be computed without the need to perform a summation over large regions of the image.

3.4 OFTNet Architecture

The Orthographic Feature Transform described in the previous section formed the central element of a novel 3D object detection architecture developed during this PhD called the OFTNet. The OFTNet was a single-stage object detection architecture inspired by the

RetinaNet architecture of Lin, Goyal, et al. (2017). Its unique novelty lay in the fact that unlike RetinaNet, which processes feature maps exclusively from the image perspective, it operated both in the image space and the birds-eye view space. This enabled it to reason both about low-level appearance features such as edges and simple shapes in the image space, as well as higher-level geometrical relationships and the three-dimensional scene layout in the birds-eye view space.

The OFTNet consisted of five main components:

- A frontend **feature extractor** subnetwork, which processes the input in image space and generates a set of multiscale image-based feature maps.
- A stack of **orthographic feature transforms**, which map image-based features into the birds-eye view.
- A residual **topdown network**, used to process features in the birds-eye view and reason about the arrangement of objects in 3D.
- A set of **output regression heads**, which predict an encoded representation of the object present at each location.
- An **object decoder**, which decodes the predicted bounding boxes and applies non-maximum suppression to remove redundant predictions.

An overview of the architecture is shown in Figure 3.4. A full specification for the network, including details of individual layers and components is shown in Figure 3.7, and the source code for the OFTNet can be found at https://github.com/tom-roddick/oft. The remainder of this section will discuss each component of the network in detail.

3.4.1 Feature Extractor

The main philosophy behind the OFTNet and this thesis in general is that for 3D tasks, as much reasoning as possible should take place in the birds-eye view space. That said however, the image-based representation still contains valuable low-level information, such as edges and basic 2D structure, which would be lost if the image were mapped into a birds-eye view representation directly. For this reason, before transforming to the birds-eye view the OFTNet first applies a small convolutional feature extractor network which generates an initial image-based feature representation of the scene. In OFTNet, this feature extractor took the form of a modified ResNet-18 network from the highly successful ResNet family of architectures (He, Zhang, et al., 2016). ResNet-18 is the shallowest of the ResNet architectures proposed by He,

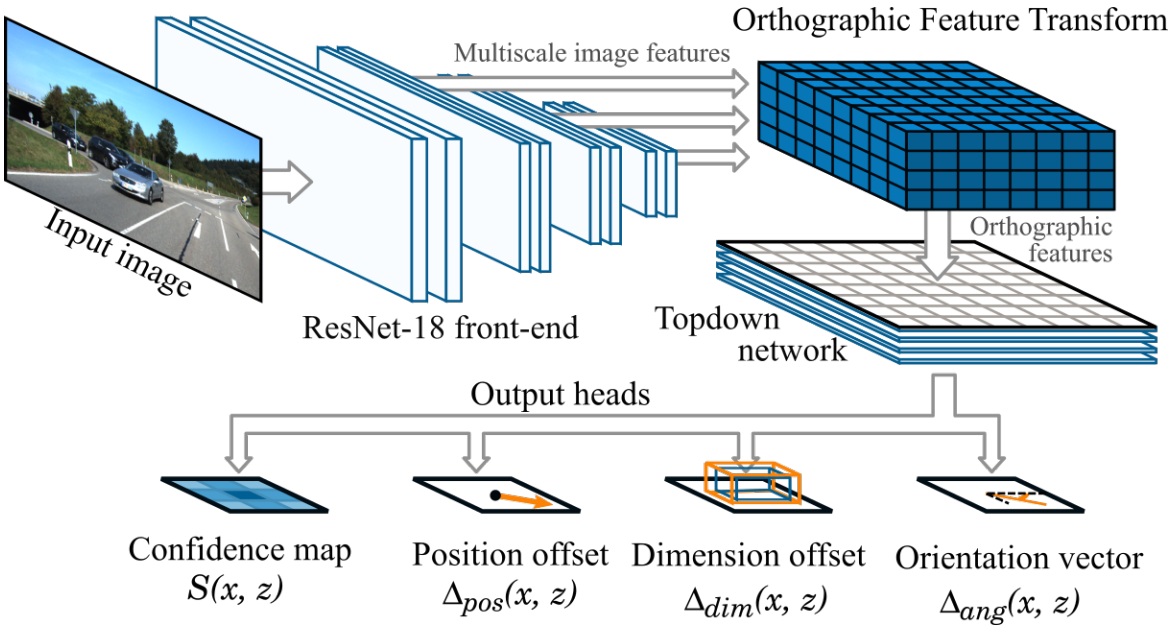


Figure 3.4. The OFTNet architecture

Zhang, et al. (2016), in contrast to the more commonly used ResNet-50 and ResNet-101. It was chosen to reduce overfitting and to place greater emphasis on later parts of the network, which operate on features in the birds-eye view space. This choice of architecture will be justified further through experiments which are described in Section 3.5.7. The use of a standard image-based feature extractor also has the added advantage that the weights of the network can be initialised by pretraining on the ImageNet dataset, which significantly reduces training time.

The feature extractor network was modified by removing the final average pooling and classification layers, and instead outputting three intermediate feature maps after the *conv3*, *conv4* and *conv5* layers of the network. These outputs are downsampled by a downsampling factor $d \in \{8, 16, 32\}$ respectively with respect to the original input image dimensions.

3.4.2 Orthographic Feature Transform

Given the image-based feature maps generated by the front-end network, the next stage was to convert these to a birds-eye view representation. This was achieved through the novel orthographic feature transform introduced in Section 3.3. The OFT took an image-based feature map $\mathbf{f}^{img}$, and mapped it to the birds-eye view using the camera intrinsic matrix K. The OFTNet network used OFT layers with a spatial resolution ρ of 0.5m in all dimensions. The grid covered a region 40m to the left and right of the camera (along the x-axis), 80m in front (z-axis), and from 1m above to 3m below the optical axis in the y-axis.

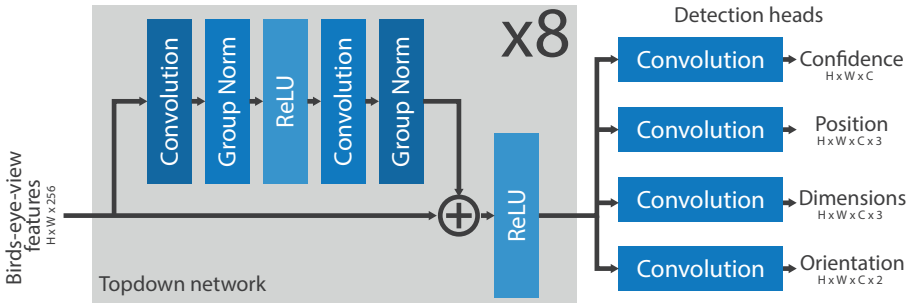


Figure 3.5. Illustration of the topdown network architecture and detection heads. The topdown network consisted of a sequence of 8 residual units derived from the basic blocks of He, Zhang, et al. (2016).

OFTNet included three OFT modules, applied to each of the feature maps $\mathbf{f}_i^{img}$ corresponding to the *conv3*, *conv4* and *conv5* stages of the network. Since these feature maps were downsampled with respect to the input image, the intrinsics matrix K was adjusted by dividing the first two rows of the matrix by the downsampling factor d_i. The final birds-eye view feature map was then obtained by summing the contribution from each OFT module. This multi-scale approach was similar to the skip connections used by e.g. SegNet (Badrinarayanan et al., 2017) and U-Net (Ronneberger et al., 2015), or the feature pyramid network of Lin, Dollár, et al. (2017). It enabled the OFTNet to take advantage of both high frequency information from higher-resolution feature maps (important for recognising distant objects), as well as the higher-level semantic context provided by deeper network features.

3.4.3 Topdown Network

While processing in the image space was performed by the feature extractor, processing in the birds-eye view space was performed by a second dedicated subnetwork called the topdown network. The topdown network was a simple residual network, consisting of eight residual blocks. The residual structure of the topdown network is shown in Figure 3.5 . It consisted of a pair of convolution layers which were bypassed by a residual (skip) connection. In the original ResNet paper, He, Zhang, et al. (2016) motivated the use of skip connections by arguing that if additional convolution layers do not bring any benefit, the network can simply learn to treat the residual block as identity. This takes on special significance in our model since in the early stages of training it allowed the network to primarily rely on the predictions from the pretrained frontend network, and gradually incorporate more 3D spatial context as training progressed.

A common problem in 2D object detection is that the appearance of objects can vary significantly depending on their position in 3D space, which complicates the learning process. Since the topdown network was fully convolutional, the filters applied to the input features were spatially-invariant. The aspiration of this approach therefore was that the network would learn a more generalisable representation of objects such as cars and pedestrians, allowing it to generalise to objects in unseen spatial configurations.

3.4.4 Classification Head

The primary aim of the OFTNet architecture was to identify regions of the 3D space which may contain objects belonging to one of the categories of interest. In a typical image-based object detector such as Faster R-CNN (Ren et al., 2015) or SSD (Liu et al., 2016), the space of all possible bounding box locations is partitioned into a large number of reference bounding boxes known as 'anchors', and the aim of the classification head is to classify each anchor as positive or negative.

Given the 3D nature of our particular task, choice of architecture and the particular problem domain of autonomous driving, a number of observations may be made which vastly simplified the design of the classification portion of the network:

1. No two objects may occupy the same region of 3D space, which means that an object may be uniquely identified by its centroid (unlike in 2D detection where both bounding box centroid and size are necessary).

2. All objects are constrained to lie on the ground surface, which means that only one vertical anchor is needed per birds-eye view location.

3. Because the topdown network operates in the birds-eye view, the variance in bounding box scale is much smaller and depends only on object dimensions, not on distance of the object from the camera. This means that fewer anchor boxes are needed to span the scale space.

4. Bounding box scale is highly correlated with object class. By comparison, intra-class variations in dimensions are relatively small.

On the basis of these observations, the OFTNet adopted a much simpler approach to anchor box classification. Each location on the birds-eye view was assigned a single anchor box per object category. The dimensions of each anchor box were given by the mean dimensions $\bar{d}$ over that object category. In other words the prediction of both the class and anchor box scale were coupled, which avoided the need for a complex two-stage architecture.

Since each birds-eye view location was associated with just a single anchor box per class, the anchor classification problem may be viewed as the estimation of a 2D spatial function S called a confidence map. We would expect the confidence map to have maximum value close to the centre of objects, and minimum value far away from objects. This problem bore similarity to the related landmark detection task (Cao et al., 2017), as well as the heatmap-based object detection approach of Huang, Yang, et al. (2015). We therefore adopted the following approach:

Instead of performing a binary classification of positive and negative anchors, we instead represented an object as a peak in the 2D confidence map. Given a set of birds-eye view feature maps from the topdown network, the aim of the classification head was to predict a set of C confidence maps $\{\mathbf{S}^c\}, c \in \{1,\dots,C\}$, where C was the number of object categories. Each ground truth object $o_n, n \in \{1,\dots,N\}$ in the scene with class label c_n was represented by a Gaussian blob, centred on the bounding box centroid $\bar{\mathbf{X}}_n = (\bar{X}_n, \bar{Y}_n, \bar{Z}_n)^T$ and with fixed variance σ^2. In other words, the ground truth confidence associated with object o_n at birds-eye view location $\mathbf{x}_{pq} = (X_p, Z_q)$ was given by

$$S_{pq} = \exp\left(-\frac{(\bar{X}_n - X_p)^2 + (\bar{Z}_n - Z_q)^2}{2\sigma^2}\right). \tag{3.14}$$

The value of the confidence map S^c associated with class c was then given by the maximum confidence across all objects belonging to that class, i.e.

$$S^c_{pq} = \max_n \mathbb{1}(c_n = c) \exp\left(-\frac{(\bar{X}_n - X_p)^2 + (\bar{Z}_n - Z_q)^2}{2\sigma^2}\right), \tag{3.15}$$

where $\mathbb{1}(\cdot)$ is the boolean indicator function. The use of the maximum function here was important to ensure that nearby peaks remain separated, unlike the mean or sum aggregation functions which have a tendency to merge adjacent peaks into a single, poorly-defined peak (Cao et al., 2017).

The network was then trained to minimise the absolute (ℓ_1) difference between the predicted confidence S and ground truth confidence $\hat{S}$, i.e.

$$\mathscr{L}(S, \hat{S}) = \sum_{p,q} \sum_{c=1}^{C} \left| S^c_{pq} - \hat{S}^c_{pq} \right| \tag{3.16}$$

An example of the predicted confidence maps obtained using this approach is shown in Figure 3.6.

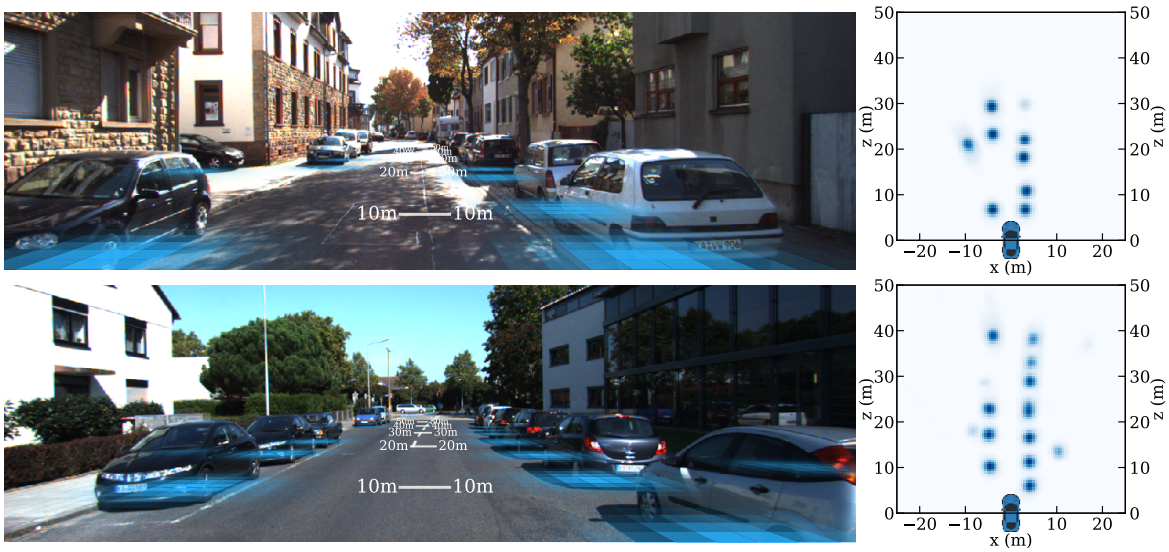


Figure 3.6. Examples of confidence maps for the car class predicted by our approach. The image on the right shows the predicted peaks in the birds-eye view, while the image on the left shows their corresponding projection onto the ground plane. Note that the ground plane is used only for visualisation and is not an input to our approach.

Class balancing

A common problem with object detection is the extreme class imbalance between positive and negative anchor boxes. Although the OFTNet adopted a confidence regression rather than a classification approach, the same problem still applied, since the vast majority of locations had close to zero confidence. In this setting, the network was able to achieve a low overall loss simply by setting the confidence at all locations to zero. We overcame this problem using a simple weighting strategy, where each birds-eye view location (p,q) was assigned a weighting factor α_{pq} based on its associated confidence. The modified loss function is given by

$$\mathscr{L}(S,\hat{S}) = \sum_{p,q}\sum_{c=1}^{C} \beta_{pq}^{c} \left| S_{pq}^{c} - \hat{S}_{pq}^{c} \right| \tag{3.17}$$

where

$$\beta_{pq}^{c} = \begin{cases} 1 & if S_{pq}^{c} > 0.05 \\ 0.01 & otherwise \end{cases} \tag{3.18}$$

3.4.5 Bounding Box Estimation

The classification procedure described above served to identify objects and their categories and to provide an approximate localisation of their position in 3D space. Most 3D object

detection benchmarks however demand a more detailed description of the detected objects, including their approximate size and orientation. This was achieved using an additional set of regression heads collectively known as the bounding box regression heads.

In 3D object detection, a 3D bounding box o_n is typically represented by seven degrees of freedom: three position parameters $\bar{\mathbf{X}}_n = (\bar{X}_n, \bar{Y}_n, \bar{Z}_n)^T$; three dimension parameters $\mathbf{d}_n = (w_n, h_n, l_n)^T$ representing the width, height and length of the bounding box; and one orientation parameter θ_n representing the yaw angle (rotation about the vertical axis). In autonomous driving applications it is usually assumed that rotation around the pitch and roll axes are negligible, which simplifies the problem compared to the more general 6 degree of freedom pose estimation task.

Directly predicting these seven quantities is a challenging learning task. Instead, most object detection approaches use some form of encoded representation to predict object bounding boxes. This approach was adopted by the OFTNet, which included three further regression heads in addition to the classification head described above. These regression heads were used to predict separate encodings for the position, dimension and orientation parameters respectively at each location on the birds-eye view. The encoded representations were as follows:

Position encoding

The coarse position estimate provided by the classification head was refined by encoding the relative offset between the centre of the object bounding box $\bar{\mathbf{X}}_n$ and the centre of the birds-eye view grid cell $\mathbf{x}$. To encode the vertical (Y) coordinate, we used the offset relative to a fixed height Y_0 below the camera origin. The encoded position offset at birds-eye location $\mathbf{x}$, referred to as Δ_{pq}^{pos}, was therefore given by

$$\Delta_{pq}^{pos} = \begin{bmatrix} \frac{\bar{X}_n - X_p}{\sigma} & \frac{\bar{Y}_n - Y_0}{\sigma} & \frac{\bar{Z}_n - Z_q}{\sigma} \end{bmatrix}^T \tag{3.19}$$

The offset was scaled by the width of the confidence map peaks σ described in Section 3.4.4, which approximately normalised the offsets to a standard unit normal distribution.

Dimension encoding

The dimensions were encoded using the logarithmic offset between the object width, height and length of the object, and the mean dimensions across all objects belonging to the same

class. For an object belonging to class c_n, the dimension encoding Δ^{dim} was therefore

$$\Delta_{pq}^{dim} = \begin{bmatrix} \log \frac{w_n}{\bar{w}^{c_n}} & \log \frac{h_n}{\bar{h}^{c_n}} & \log \frac{l_n}{\bar{l}^{c_n}} \end{bmatrix}^T \tag{3.20}$$

where $\bar{w}^c$, $\bar{h}^c$, $\bar{l}^c$ were the mean width, height and length respectively of all ground truth bounding boxes belonging to class c. Predicting the log dimensions rather than estimating the size parameters directly has the advantage that after decoding the parameters are guaranteed to be strictly positive.

Orientation encoding

Predicting the orientation parameter θ presented a particular challenge, since it was unclear how to represent a periodic angle using a linear regression layer. This also raised difficulties in choosing a loss function due to the discontinuity at $0 = 2\pi$. To overcome this problem, we followed a number of previous works (Wirges et al., 2018; Ku, Mozifian, et al., 2018; Yang, Luo, et al., 2018) in encoding the orientation in terms of its sine and cosine, i.e.

$$\Delta_{pq}^{ang} = \begin{bmatrix} \sin\theta_n & \cos\theta_n \end{bmatrix}^T. \tag{3.21}$$

Note that most image-based detection networks (Mousavian et al., 2017) predict an intermediate quantity called the *observation angle*, which is the yaw angle θ adjusted for the perceived rotation due to the relative position of the object with respect to the camera. A further advantage of the birds-eye view representation is that it decouples this dependency between object position and rotation, allowing the OFTNet to predict the absolute rotation θ directly.

Loss functions

As with the classification head, each encoding was predicted using a single convolution layer which operated on the birds-eye view features generated by the topdown network. In each case the network was trained to minimise the ℓ_1 distance between the ground truth encoding Δ and the corresponding network prediction $\hat{\Delta}$. The loss was only computed for positive birds-eye view locations i.e. those which intersected at least one object bounding box. The total loss for each regression head was given by the sum of the ℓ_1 losses over all locations and object classes.

$$\mathscr{L}_k(\Delta^k, \hat{\Delta}^k) = \sum_{p,q} \sum_{c=1}^{C} \mathbb{1}\left(\hat{S}_{pq}^c > 0.05\right) \left|\Delta_{pq}^k - \hat{\Delta}_{pq}^k\right|, \quad k \in \{pos, dim, ang\} \tag{3.22}$$

The total loss for the network, including the confidence map loss $\mathscr{L}_{conf}$ and all three bounding box encodings, was obtained by a weighted sum

$$\mathscr{L}_{total} = \lambda_{conf}\mathscr{L}_{conf} + \lambda_{pos}\mathscr{L}_{pos} + \lambda_{dim}\mathscr{L}_{dim} + \lambda_{ang}\mathscr{L}_{ang} \tag{3.23}$$

where the values of the weighting factors λ were obtained through hyperparameter search.

3.4.6 Non-Maximum Suppression

The output from the classification and bounding box heads was a set of 2D tensors representing the confidences and associated bounding box parameters for each location in the birds-eye view. At test time, the final task of the network is to turn these dense predictions into a discrete set of object bounding boxes. This was achieved using a process known as non-maximum suppression (Dalal and Triggs, 2005).

As discussed in Section 3.4.4, the classification head of the OFTNet network was trained to indicate the existence of an object by generating a Gaussian peak in the predicted confidence function S. We could therefore recover and approximately localise predicted objects by searching for local maxima in the confidence function. The non-maximum suppression algorithm consisted of three main steps.

Firstly, a Gaussian smoothing kernel G with variance σ^2_{NMS} was applied to the predicted confidence maps S. This helped alleviate the effects of high frequency noise in the predicted confidence function. The smoothed confidence function $\tilde{S}$ was given by

$$\tilde{S}^c_{pq} = S^c_{pq} * G(\sigma_{NMS}). \tag{3.24}$$

The second step was to identify local maxima by considering the confidence at the eight neighbouring locations surrounding each point (p,q) in the birds-eye view. A location was deemed to be a local maximum if

$$\tilde{S}^c_{pq} \geq \tilde{S}^c_{p+m,q+n}, \quad \forall m,n \in \{-1,0,1\}. \tag{3.25}$$

For each detected maximum, the corresponding confidence score S^*_{pq} (prior to the smoothing score) and bounding box parameters were sampled to obtain a list of detected objects. To avoid including low confidence bounding box predictions in the final output, the final step was to apply a confidence threshold t which eliminated all predictions where $S^*_{pq} < t$.

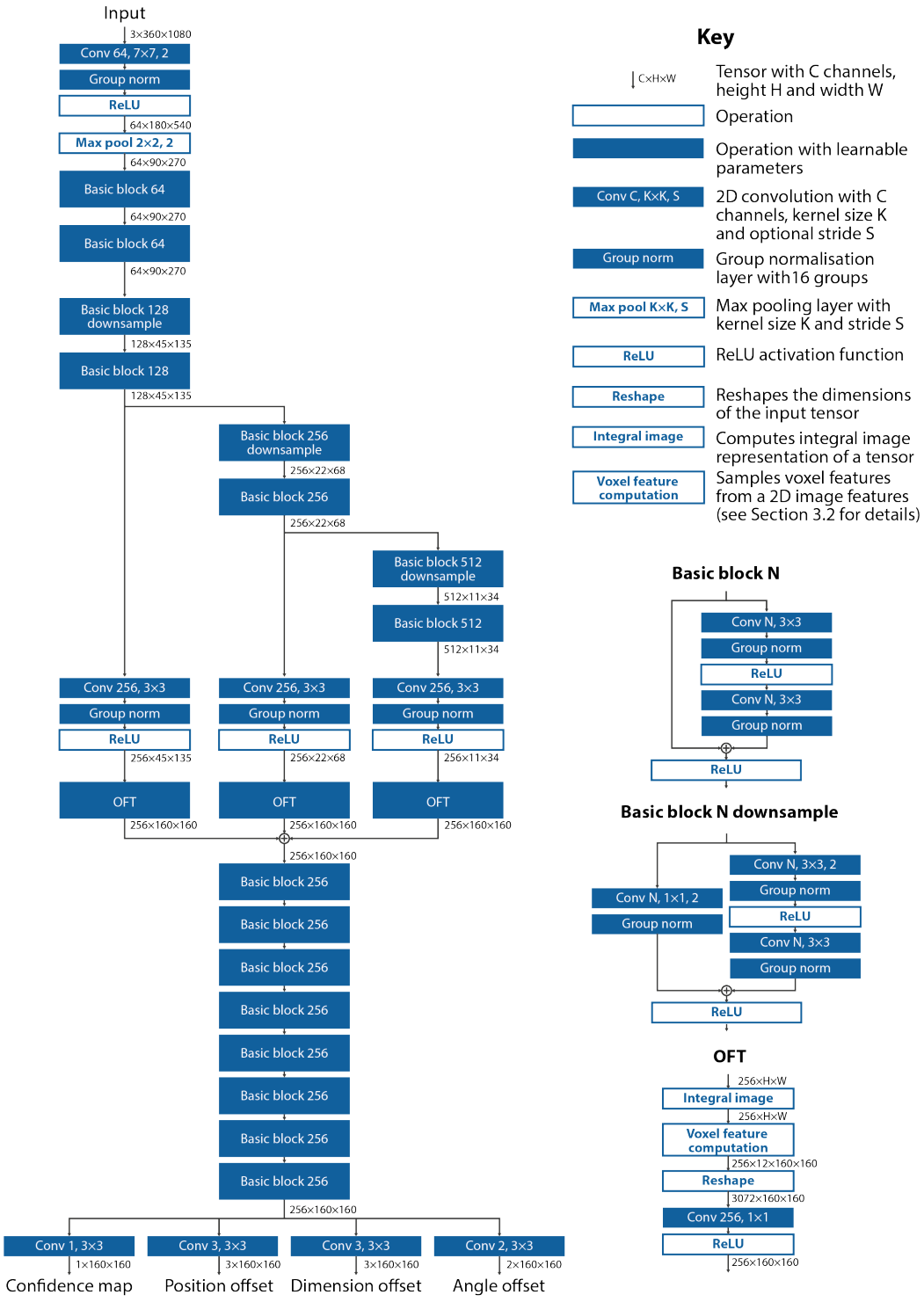


Figure 3.7. A complete specification of the OFTNet architecture. Filled blocks represent neural network layers with trainable parameters. Descriptions of submodules within the network, consisting of the residual basic block (He, Zhang, et al., 2016), residual basic block with downsampling, and orthographic feature transform, are shown on the right.

(a) City (b) Residential (c) Road

Figure 3.8. Examples of images from the KITTI object detection benchmark.

3.5 Experiments

3.5.1 KITTI Object Detection Benchmark

At the time that the OFTNet algorithm was developed, the only large-scale computer vision dataset to provide ground truth 3D object bounding boxes was the object detection dataset associated with the KITTI vision benchmark suite (Geiger, Lenz, and Urtasun, 2012). KITTI is a large-scale driving dataset which has been hugely significant in the development of many autonomous driving technologies and algorithms. It provides training and evaluation data, metrics and online leaderboards for 14 fundamental computer vision tasks including optical flow prediction, depth estimation, visual odometry and road segmentation, as well as approximately 1.5 hours of raw sensor data (Geiger, Lenz, Stiller, et al., 2013). The data was collected from a mobile data collection platform driving through urban areas of Karlsruhe, Germany, and incorporates a number of different environments including city, residential and road scenes. Examples of typical scenes are shown in Figure 3.8. The platform incorporates a sensor suite including two pairs of stereo cameras (one grayscale and one colour), a Velodyne LiDAR sensor, and a high-fidelity inertial and GPS navigation system. A rigorous calibration and synchronisation procedure was used to provide accurate camera intrinsic parameters and transformation matrices between the different sensor modalities.

Most of the subsequent evaluation and training described in this section focused on the data provided in the KITTI object detection benchmark. The object detection benchmark dataset is a subset of the data provided by Geiger, Lenz, Stiller, et al. (2013) which provides ground truth annotations for eight object categories including cars, trucks, trams and pedestrians. These annotations include basic information such as position, size and orientation, as well as more detailed attributes such as level of occlusion by other objects or truncation by the boundaries of the image. A wide range of sensor data is available, but for this work only monocular colour images were used as input.

The object detection dataset is divided into 7481 frames of training data and 7518 frames of test data. To prohibit new algorithms from overfitting to the test set, the ground truth annotations for these frames is withheld. As a result most works in the object detection literature have adopted the strategy first proposed by Chen, Kundu, Zhu, et al. (2015) and further subdivide the training set into 3712 frames of training data and 3769 frames of validation data. In this thesis results are reported on both the official test set and the Chen, Kundu, Zhu, et al. validation set. Auxiliary experiments, such as the ablation studies in Section 3.5.7, were conducted on the validation set. The official KITTI benchmark evaluates detection performance on three object categories: cars, pedestrians and cyclists. However, at the time of developing the OFTNet, no monocular detection algorithm had demonstrated reliable performance on the pedestrian and cyclist classes. Therefore, following the example of other contemporary works (e.g. Xu and Chen (2018)) the majority of the evaluation in this section focuses predominantly on the car object category.

3.5.2 Training procedure

The OFTNet network was implemented as described in Section 3.4, and was trained on the KITTI object detection training set using the following procedure. The weights of the network W were optimised using the stochastic gradient descent (SGD) algorithm (Kiefer, Wolfowitz, et al., 1952), according to the update rule

$$W_{n+1} = W_n - \eta \sum_{i \in B_n} \nabla \mathscr{L}_i(W_n) \tag{3.26}$$

where η is the learning rate, $\mathscr{L}_i$ is the loss function evaluated with respect to training example i, and B_n is a randomly sampled mini-batch of examples at step n. The mini-batch size $|B_n|$ was set at 8 for the experiments described in this chapter. Following work by Masters and Luschi (2018), the sum rather than the mean of the mini-batch gradients $\nabla \mathscr{L}_i$ was used to compute the gradient update. This helped to decouple the choice of learning rate η from other hyperparameters such as the batch size and birds-eye view dimensions. To compensate, η was set at a very low value of 10^{-7}, which was approximately equivalent to a learning rate of 0.2 under the more common gradient averaging scheme. In addition to the standard SGD update rule, the optimisation made use of the popular momentum technique of Rumelhart et al. (1986) using an exponential decay factor of 0.9; and an ℓ_1 weight regularisation term with a magnitude of 10^{-4}. The model was trained for 600 epochs over the training set, which took approximately 120 hours on 4 Titan X GPUs.

3.5.3 Metrics

Precision and recall

Object detection can be viewed as an information retrieval problem. Given a query (in the form of an image), the objective is to return a list of all relevant bounding boxes in the scene (i.e. those that correspond to ground truth objects) while ignoring all irrelevant ones (i.e. those that correspond to background). Thus, two main metrics are important: the percentage of relevant objects which are correctly detected, and the percentage of detected objects which are relevant. These two quantities are referred to as recall and precision respectively.

Recall is defined as the number of correct detections, divided by the total number of ground truth objects present. In other words,

$$\text{recall} = \frac{\text{number of true positive detections}}{\text{number of true positive} + \text{number of false negative}}. \tag{3.27}$$

A detection is described as a *true positive* if it overlaps a ground truth bounding box by a specified amount. In 3D object, this overlap criteria is given by the *intersection over union* (IoU) score between the detected and ground truth bounding box, given by:

$$IoU(\hat{B}, B) = \frac{|\hat{B} \cap B|}{|\hat{B} \cup B|} \tag{3.28}$$

where $|\hat{B} \cap B|$ and $|\hat{B} \cup B|$ represent the volumes of intersection and union between the predicted and ground truth bounding boxes $\hat{B}$ and B respectively (Everingham et al., 2011). A false negative detection meanwhile is a ground truth object which does not have any matching detections.

Similarly, **precision** is defined as the number of correct detections, divided by the total number of detections:

$$\text{precision} = \frac{\text{number of true positive detections}}{\text{number of true positive} + \text{number of false positive}}. \tag{3.29}$$

A false positive here means a detection which doesn't correspond to any ground truth objects.

Note that it is always necessary to quote both precision and recall together when analysing an object detection system. This is because it is, for example, always possible to achieve an arbitrarily high recall by enumerating all possible bounding boxes, at the expense of precision.

Precision-recall curves

In reality, it is often necessary to compromise between achieving a high precision or a high recall. The relative importance of these two metrics will often depend on the application. For example in autonomous driving, a high recall may be favoured to ensure potential hazards are not missed. Meanwhile, a biometric identity recognition system may prioritise a high precision to prevent unauthorised users gaining access. It is therefore useful for an object detection system to be able to adjust the trade-off between precision and recall depending on the application. This is usually achieved by associating a confidence score s_i with each detected bounding box. The actual values of the confidence score are arbitrary, but should be chosen such that the highest confidence bounding boxes are those most likely to contain an object. All predicted boxes below a certain confidence threshold t are eliminated, allowing a particular trade-off between precision and recall, known as the *operating point*, to be selected by altering the value of t.

The overall performance of an object detection system can therefore be characterised by plotting the set of all possible operating points on a pair of axes showing recall against precision, as functions of the threshold t. Several examples of this type of chart, which are called precision-recall curves, are shown in Figure 3.10. The optimum precision-recall curve is one which occupies the top-right corner of the axes, indicating that even at very high recall rates, very few false-positive predictions are returned.

Average precision

Whilst precision-recall curves are helpful in visualising the range of possible operating points for a system, in order to compare between methods it is useful to have a single scalar metric which summarises the overall performance of the system. A common solution is the average precision (AP) score which corresponds to the area under the precision-recall curve, i.e.

$$AP = \int_0^1 p(r) d\mathrm{r} \tag{3.30}$$

where r is the recall and $p(r)$ is the precision evaluated at a particular recall value.

KITTI AP metric

The KITTI object detection benchmark uses a slight variation on the standard average precision score as its primary evaluation metric (Geiger, Lenz, and Urtasun, 2012). It adopts the interpolated AP metric of Everingham et al. (2011), which samples the recall at 11

discrete thresholds, i.e.

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p(r). \tag{3.31}$$

This is the metric used to rank methods in the official KITTI benchmark, and therefore provides the basis of the quantitative evaluation in Section 3.5.5 below. The KITTI benchmark requires an 3D overlap of at least 70% between the ground truth and predicted bounding boxes for a prediction to be treated as a true positive.

Drawbacks of the KITTI AP metric

One limitation of the KITTI benchmark metric is that for monocular methods including ours, where it is difficult to accurately estimate the depth of objects, the 70% overlap criteria is extremely stringent. This commonly results in scenarios where the majority of objects may be correctly detected and localised in 2D space, but the localisation or scale accuracy in 3D space is insufficient, resulting in very low average precision scores (<10%) for monocular methods. For example, given an car object of average dimensions, a localisation error of just a few centimetres is sufficient for the prediction to be treated as a false positive. Due to this factor, a large number of otherwise correct detections are rejected, which leads to poor performance on the official metric.

Despite these drawbacks, we include quantitative evaluation based on this metric in order to allow comparison against state-of-the-art methods at the time, but we note that the low average precision scores do not necessarily reflect the quality of driving performance in a real autonomous driving application. To better understand the true performance of the method, we consider other metrics, such as the average translation, scale and rotation error (Caesar, Bankiti, et al., 2019) in Sections 3.5.9 and 3.6. Whilst not considered in this thesis, additional metrics, such as those which consider the reprojection of the 3D box into the 2D image space or the impact of missed detections on the path predicted by a complete autonomous system, may give further insights into the true performance of the method.

3.5.4 Qualitative comparison

Figure 3.9 provides a qualitative comparison between the OFT method and the Mono3D approach of Chen, Kundu, Zhang, et al. (2016), the leading competitor at the time. For each image the N most confident bounding boxes are shown, where N is the number of ground truth bounding boxes present. Using a fixed value for N allows a fair comparison between the two methods.

Both methods qualitatively achieve similar recall, with the majority of objects visibly detected. The advantage of our method appeared to primarily lie in improved localisation performance: the bounding boxes produced by OFTNet were routinely better aligned with the shape of the ground truth objects. OFTNet also typically performed better in the case of objects which are very distant (e.g. row 3) or very close (rows 7, 8) to the camera. We argue the transformation of features into the birds-eye view space helps the network to better handle these extrema in scale.

The last two rows illustrate failure cases of the OFTNet. One scenario that both methods consistently failed on is rows of cars parked at right-angles to the direction of travel. An example of this effect is shown in row 8 of Figure 3.9. It is clear that both networks were strongly biased towards predictions which were oriented directly towards or away from the camera. We suspect this was a product of a severe inbalance between the training set and the validation set, and consider this problem further in Section 3.6.3.

3.5.5 Comparison to state of the art

The OFTNet method was evaluated by considering two tasks from the KITTI object detection benchmark. The first was the 3D bounding box detection task. For this task, in order for a predicted bounding box to be treated as a true positive detection, it must have at least a 70% IoU score with a ground truth bounding box. At the time OFTNet was originally developed, the official KITTI benchmark contained only one other published image-only method: (Novak, 2017), so we evaluated against both the official benchmark test set (with held-out annotations) as well as the validation set proposed by Chen, Kundu, Zhu, et al. (2015) (for which ground truth annotations were publicly available). Each evaluation set further subdivided into three difficulty levels: 'easy', 'moderate', and 'hard', based on the level of occlusion present and the apparent size of objects in the image. We focused on the 'car' object category since no previous monocular method had reported results on either of the two remaining benchmark classes, 'cyclist' or 'pedestrian'. The results are based on the interpolated average precision score described in Section 3.5.3, and are tabulated in the first four rows of Table 3.1. To provide additional context to these results, we also include a selection of more recent works which were published after OFTNet was first developed, as well as a small number of contemporary methods which made use of LiDAR and other sensors.

The second benchmark we considered was the birds-eye view car detection task. This uses a slightly more lenient overlap criteria: for a predicted bounding box to be considered correct its projection into the birds-eye view should overlap a ground truth box by at least

(a) Mono3D (b) OFTNet (Ours)

Figure 3.9. Qualitative comparison between Mono3D (Chen, Kundu, Zhang, et al., 2016) and OFTNet on the KITTI validation set. We show the top N most confident predictions, where N is the number of ground truth boxes. The bottom two rows show failure cases for both approaches.

70%, based on the IoU score. The results on this benchmark, on both the KITTI test and validation sets, are shown in Table 3.2.

The first observation to be made is that across both benchmarks, the performance of all monocular-only methods was very low compared to that of methods which made use of LiDAR data. At the time that OFTNet was first introduced, the highest performing method on the KITTI test benchmark achieved less than 5% of the AP_{3D} score of corresponding LiDAR-based methods. As can be seen from the qualitative results in Section 3.5.4, these low scores do not necessarily reflect the fact that objects are rarely detected, but rather that the extreme difficulty of accurately localising objects in 3D using monocular methods means that many predictions fail the extremely challenging 70% overlap requirement of the KITTI benchmark.

In spite of this challenge however, we were able to achieve modest improvements over the baseline methods of Novak (2017) and Chen, Kundu, Zhang, et al. (2016) across both benchmarks and all three difficulty levels. The improvement was particularly pronounced on the birds-eye view benchmark, where we were able to more than double the performance of the prior state-of-the-art across all but one metric. The fact that OFTNet performed well on the birds-eye view benchmark may perhaps be expected, given that the ability to reason in the topdown space was a key part of the design philosophy of the method.

Another noteworthy observation is that on the birds-eye view benchmark, OFTNet compared favourably with the stereo-image-based 3DOP method of Chen, Kundu, Zhu, et al. (2015), even outperforming it on the 'hard' difficulty category. This result is significant because Chen, Kundu, Zhu, et al.'s use of stereo images provides them with explicit measurements of the depth of points in the scene, which simplifies the problem of localising objects. Our method on the other hand must rely on learned, implicit estimates of depth alone.

Precision-recall curves

While the average precision values reported in Tables 3.1 and 3.2 give some indication of the relative ordering of the different methods, they are rather more difficult to interpret intuitively. In order to understand these quantities more fully, it was necessary to consider the full set of possible trade-offs between precision and recall. To visualise these trade-offs, Figure 3.10 shows precision-recall curves for a small subset of methods where full results were available.

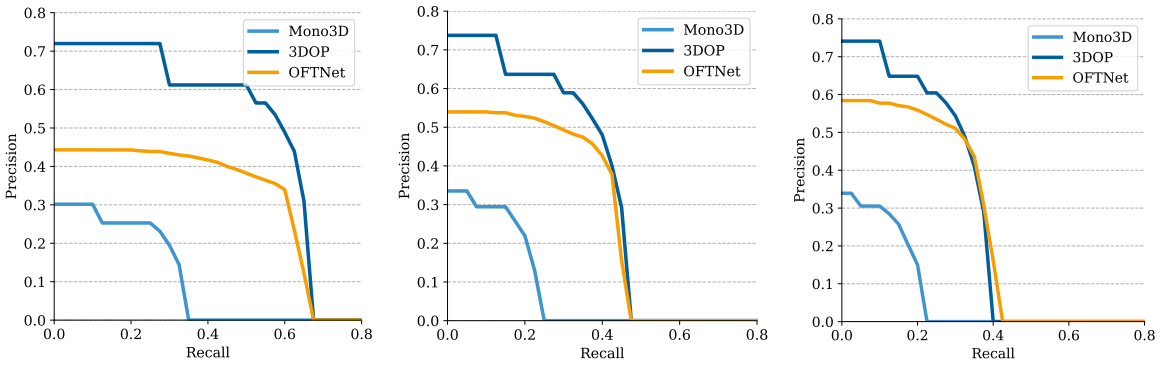
The curves in Figure 3.10 show that, compared to the monocular baseline method Mono3D (Chen, Kundu, Zhang, et al., 2016), the OFTNet was always able to achieve a better compromise between precision and recall. In other words, for any chosen operating point, the OFTNet always produced fewer false positive detections for the same number of false negatives.

Table 3.1. Average precision (%) for car 3D bounding box detection (AP_{3D}) on the official KITTI test set as well as the validation set defined by Chen, Kundu, Zhu, et al. (2015). The first section of the table provides results for image-based methods which predated OFT-Net. For context the third section shows a selection of more recent methods which illustrate the rapid advancement in monocular 3D object detection. The fourth section shows LiDAR and fusion-based methods.

Method	Modality	Validation			Test		
		Easy	Moderate	Hard	Easy	Moderate	Hard
3DOP (Chen, Kundu, Zhu, et al., 2015)	Stereo	6.55	5.07	4.10	-	-	-
Mono3D (Chen, Kundu, Zhang, et al., 2016)	Mono	2.53	2.31	2.31	-	-	-
3D-SSMFCNN (Novak, 2017)	Mono	-	-	-	2.28	2.39	1.52
OFTNet (Ours)	Mono	4.07	3.27	3.29	2.50	3.28	2.27
A3DODWTDA (Gustafsson and Linder-Norén, 2018)	Mono	10.13	8.32	8.20	6.76	6.45	4.87
Multi-level Fusion (Xu and Chen, 2018)	Mono	10.53	5.69	5.39	7.08	5.18	4.68
MonoPSR (Ku, Pon, et al., 2019)	Mono	12.75	11.48	8.59	12.57	10.85	9.06
MV3D (Chen, Ma, et al., 2017)	LiDAR+RGB	83.87	72.35	64.56	71.09	62.35	55.12
VoxelNet (Zhou and Tuzel, 2018)	LiDAR	-	-	-	77.47	65.11	57.73
AVOD (Ku, Mozifian, et al., 2018)	LiDAR+RGB	84.41	74.44	68.65	81.94	71.88	66.38

Table 3.2. Average precision (%) for birds-eye view object detection (AP_{BEV}) on the KITTI validation and test sets. The first section of the table shows image-based methods which predated OFT-Net. For context the third section shows a selection of more recent methods which illustrate the rapid advancement in monocular 3D object detection. The fourth section shows LiDAR and fusion-based methods.

Method	Modality	Validation			Test		
		Easy	Moderate	Hard	Easy	Moderate	Hard
3DOP (Chen, Kundu, Zhu, et al., 2015)	Stereo	12.63	9.49	7.59	-	-	-
Mono3D (Chen, Kundu, Zhang, et al., 2016)	Mono	5.22	5.19	4.13	-	-	-
3D-SSMFCNN (Novak, 2017)	Mono	-	-	-	3.66	3.19	3.45
OFTNet (Ours)	Mono	11.06	8.79	8.91	9.50	7.99	7.51
A3DODWTDA (Gustafsson and Linder-Norén, 2018)	Mono	15.64	12.90	12.30	10.21	10.61	8.64
Multi-level Fusion (Xu and Chen, 2018)	Mono	22.03	13.63	11.60	13.73	9.62	8.22
MonoPSR (Ku, Pon, et al., 2019)	Mono	20.63	18.67	14.45	20.25	17.66	15.78
MV3D (Chen, Ma, et al., 2017)	LiDAR+RGB	86.55	78.10	76.67	86.02	76.90	68.49
VoxelNet (Zhou and Tuzel, 2018)	LiDAR	-	-	-	89.35	79.26	77.39
AVOD (Ku, Mozifian, et al., 2018)	LiDAR+RGB	-	-	-	88.53	83.79	77.90



(a) Easy (b) Moderate (c) Hard

Figure 3.10. Precision-recall curves on the KITTI validation set for the Mono3D (Chen, Kundu, Zhang, et al., 2016), 3DOP (Chen, Kundu, Zhu, et al., 2015) and OFTNet algorithms. To better differentiate between the three methods, we use a more lenient IoU threshold of 50% to determine true positive predictions.

The most interesting feature of these curves was the point at which the precision drops to zero. The recall at this value represents the maximum fraction of ground truth objects that could be detected, if false positives were not a concern. It can be seen from Figure 3.10 that, across all difficulty levels, the OFTNet and stereo-based 3DOF method (Chen, Kundu, Zhu, et al., 2015) obtained almost identical recall, despite the stereo method having a considerable advantage on account of having access to explicit depth information. At lower recall values, the 3DOP did achieve considerably better precision than the OFTNet, but this improvement was less pronounced for the moderate and hard difficulty categories. A higher difficulty category corresponds to more objects which are heavily occluded or distant from the camera. The results in Figure 3.10 implied that these edge-cases were an area of strength for the OFTNet.

3.5.6 Evaluation on other object categories

As discussed in Section 3.5.1, the majority of the evaluation in this chapter focuses exclusively on the 'Car' object category, since no previous works had reported results on the remaining 'Pedestrian' or 'Cyclist' categories. For image-based methods, achieving high average precision on these classes is extremely challenging, as the objects' small sizes mean that they must be localised with extreme accuracy to meet the required bounding box overlap criteria. Nonetheless, we evaluated the OFTNet's performance on these classes and compared against image-based methods where detections for these classes were publicly available. The results are shown in Table 3.3.

Table 3.3. Average precision (%) for 3D bounding box detection (AP_{3D}) on the KITTI validation set for the Cyclist and Pedestrian object categories. No image-based method was able to achieve $> 1\%$ average precision on these categories.

Method	Modality	Cyclist			Pedestrian		
		Easy	Moderate	Hard	Easy	Moderate	Hard
3DOP (Chen, Kundu, Zhu, et al., 2015)	Stereo	0.73	0.52	0.51	0.60	0.57	0.46
Mono3D (Chen, Kundu, Zhang, et al., 2016)	Mono	0.06	0.05	0.05	0.10	0.11	0.11
OFTNet (Ours)	Mono	0.07	0.05	0.05	0.65	0.69	0.53

The results in Table 3.3 verified the anecdotal observation that image-based methods at the time were unable to reliably localise cyclist and pedestrian instances. For the cyclist class, both Mono3D (Chen, Kundu, Zhang, et al., 2016) and OFTNet, which relied on monocular inputs, achieved close to 0% average precision. The stereo 3DOP (Chen, Kundu, Zhu, et al., 2015) method meanwhile performed only marginally better. The Mono3D also failed on the pedestrian category, but interestingly, OFTNet actually fared better, slightly outperforming 3DOP. However, across both classes and all difficulty categories no method was able to achieve $> 1\%$ average precision. In this low-accuracy domain, random effects such as the particular composition of the dataset have a significant impact and it was therefore difficult to draw robust conclusions about the relative performance of the different methods.

3.5.7 Ablation study

The principle philosophy behind the design of the orthographic feature transform and the derived OFTNet detection architecture was that as much reasoning as possible should take place in the birds-eye view. As described in Section 3.4, most of the processing in OFTNet took place in two components: the feature extractor network, which operated on the image-based features; and the topdown network, which processed birds-eye view features. In order to assess the importance of operating in the birds-eye view, we conducted an ablation study where relative processing power is traded-off between these two components by removing layers from the topdown network and adding layers to the feature extractor network.

Specifically, two variations on the feature extractor network were considered: ResNet-18, which consists of 8 residual blocks, and ResNet-34, consisting of 16 residual blocks. Unfortunately deeper feature extractor networks, such as ResNet-50, exceeded the available GPU memory resources so the study was restricted to these two front-end networks.

We also considered three variants on the topdown network, consisting of 16, 8 and zero convolution layers. In the zero-unit version, the output features from the orthographic feature transform were passed directly to the bounding box and classification heads. This allowed

Table 3.4. Effect of modifying the number of layers in the OFTNet feature extractor and topdown network on average precision over the KITTI validation dataset.

Feature Extractor	Topdown Network	#Parameters (million)	AP_{3D} (%)	AP_{BEV} (%)
ResNet-18	0 layers	11.94	0.64	1.35
ResNet-18	8 layers	16.66	5.58	10.43
ResNet-18	16 layers	21.38	8.92	16.73
ResNet-34	0 layers	22.05	2.20	4.11
ResNet-34	8 layers	26.77	4.27	8.73
ResNet-34	16 layers	31.49	5.72	10.63

us to simulate a purely image-based network, where the OFT module effectively acted as a region of interest pooling layer. The results of this ablation study are tabulated in Table 3.4.

The findings from this study are clear: increasing the number of layers in the topdown network significantly improved the detection performance, irrespective of the feature extractor architecture chosen. This strongly suggested that the topdown network and the 3D reasoning that it performed were crucial to the overall success of the network. One might argue that the reason that the average precision increase is not specifically due to the topdown representation, but rather that the overall depth of the network, and therefore its representational power, increases as more layers are added. However, contrary to expectations we found that using a deeper feature extractor (ResNet-34 in place of ResNet-18) actually reduced performance, as a result of increased overfitting. This was true even when the overall number of parameters across different networks were similar. We conclude, therefore, that layers in the topdown network were much more generalisable and that reasoning in the birds-eye view space did make an important contribution to the performance of the OFTNet method.

3.5.8 Impact of voxel grid resolution

Aside from the choice of front-end architecture and topdown network as discussed in the previous section, perhaps the biggest single factor in the performance of the OFTNet architecture was the resolution of the voxels used in the orthographic feature transform. At low resolutions, the voxel feature representation would be too coarse to represent small objects such as pedestrians and to capture the fine details of the scene. At high resolutions, the memory usage and computation time for the OFT layer becomes infeasibly large. We therefore conducted further investigation to assess the impact of this trade-off.

The results from this investigation, shown in Table 3.5, confirmed the expected trend that as the size of the voxels increases (i.e. the resolution of the grid decreases), the performance of the OFTNet degrades significantly. At the lowest resolution considered of $\rho = 2.0m$, the average precision of the method is negligible. The best results were obtained using the default

Table 3.5. Effect of voxel size on speed and detection performance. AP_{BEV} and AP_{3D} represent the birds-eye view and 3D bounding box average precision for the car class on the KITTI validation set. Runtime measures the time taken for the OFTNet to process a single image with dimensions 1242×375px.

Metric	Voxel grid resolution (m)			
	0.5	0.75	1.0	2.0
AP_{BEV}	7.83	6.16	4.46	1.91
AP_{3D}	5.16	3.93	2.64	0.90
Runtime (ms)	393	188	114	75

resolution of 0.5m: unfortunately we were unable to evaluate at any higher resolutions due to the memory constraints of the GPU resources available at the time. However there is no reason not to suspect that using an even higher resolution would yield further improvements. Such improvements would come at considerable computational cost however: the runtime measurements in Table 3.5 implied that the resolution of the voxel grid was the dominant factor in determining the runtime of the method, and that the impact of other components such as the frontend network was relatively small by comparison. At the default resolution of 0.5m, the network took approximately 0.4 seconds to process a single image: considerably slower than realtime (approx. 0.04 seconds); a major limitation of this method.

3.5.9 Results on the NuScenes dataset

At the time of publishing OFTNet, KITTI was the only large-scale public dataset for evaluating 3D object detection. Shortly afterwards an number of alternative autonomous driving datasets were released, including the NuScenes dataset by Caesar, Bankiti, et al. (2019) which increased the number of annotated 3D bounding boxes by a factor of almost seven-fold. NuScenes is considerably more challenging than KITTI, featuring 11 object categories and over 1.4 million images captured across four locations and diverse driving conditions. As part of their analysis, Caesar, Bankiti, et al. investigated the performance of OFTNet, alongside other monocular and LiDAR-based methods, on the NuScenes test set. Results from their study are reproduced in Table 3.6.

Unfortunately, from the results in Table 3.6, it can be seen that OFTNet was not able to compete with more recent monocular detection methods such as SSD+3D (Caesar, Bankiti, et al., 2019) or MonoDIS (Simonelli et al., 2019). Unlike our approach, which aimed to use the structure of the network (i.e. the transformation to the birds-eye view) to decouple the relationship between 3D bounding box coordinates and their 2D projections, MonoDIS instead used a novel disentangling loss which eliminated dependencies between bounding box parameters. They were able to show that this decoupling was able to provide a

Table 3.6. Object detection results on the NuScenes test set. For Nuscenes detection score (NDS) and mean average precision (mAP), higher is better. For mean average translation error (mATE), scale error (mASE) and orientation error (mAOE), lower is better. Results are reproduced with permission from Caesar, Bankiti, et al. (2019).

Method	Modality	NDS (%)	mAP (%)	mATE (m)	mASE 1-IoU	mAOE (rad)
OFTNet	Mono	21.1	12.6	0.82	0.36	0.85
SSD+3D (Caesar, Bankiti, et al., 2019)	Mono	26.8	16.4	0.90	0.33	0.62
MonoDIS (Simonelli et al., 2019)	Mono	38.4	30.4	0.74	0.26	0.55
Point Pillars (Lang et al., 2019)	LiDAR	45.3	30.5	0.52	0.29	0.50
Megvii (Zhu, Jiang, et al., 2019)	LiDAR	63.3	52.8	0.30	0.25	0.38

more direct loss function trajectory for optimisation and thereby improved object detection performance compared to our and others' works. SSD+3D, a baseline provided by the authors of NuScenes (Caesar, Bankiti, et al., 2019), adopted the same loss but incorporated it into a single-stage architecture more comparable to ours. The one area where the OFTNet performed well was in the average translation error (mATE), which measures the average distance from the centre of the predicted bounding box to the corresponding ground truth. On this metric, the OFTNet performed slightly better than the SSD+3D approach and only marginally worse than MonoDIS. A strength of the OFTNet therefore was its ability to accurately localise objects. Across other metrics however, such as average scale error and average orientation error, the OFTNet performed worse than subsequent methods on the NuScenes dataset.

Nonetheless, an important observation from the analysis by Caesar, Bankiti, et al. (2019) was that the performance of OFTNet on the KITTI dataset was heavily data-limited. They carried out an additional experiment where they trained OFTNet (along with other methods) on different fractions of the NuScenes training set. The results of this experiment are reproduced in Figure 3.11. Their finding was that when trained on just 15% of the training data (roughly the size of the KITTI training set), OFTNet achieved a mean average precision score of less than 50% of that when the full training set was used. This shows, firstly, that one of the drawbacks of the OFTNet approach was its large data requirement. However, it also illustrates the limitations of the KITTI dataset and suggests that more research is needed to understand the behaviour of OFTNet in the scenario where data is more abundant.

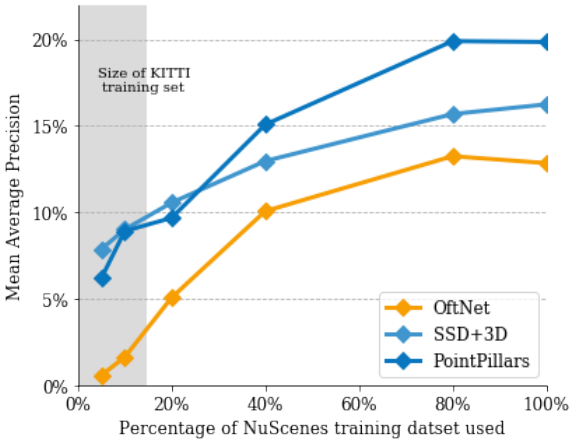


Figure 3.11. Effect of training set size against average precision on the NuScenes dataset. Grey shaded area represents the number of annotated images available in the KITTI training set. Figure generated using data from Caesar, Bankiti, et al. (2019) with permission from the authors.

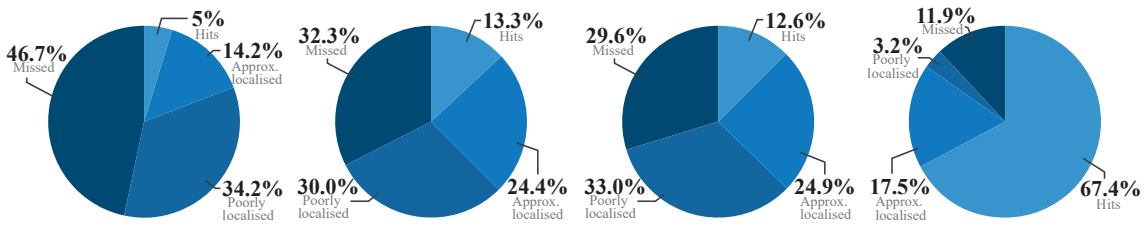
3.6 Analysis

3.6.1 Error analysis

As discussed in Section 3.5.3, quantifying the performance of an object detection system is in general a highly challenging task. This is in part due to the complex interplay between the different requirements of the task: correctly identifying regions of the image which contain an object of interest; recognising the object's semantic category; and accurately localising the object's bounding box. To attempt to disentangle some of these factors, Hoiem et al. (2012) propose a framework for diagnosing sources of error within a 2D object detection algorithm. The analysis in this section builds on this framework to more deeply understand the performance of OFTNet in reference to other contemporary methods.

In general, errors in an object detection system may be characterised according to three main categories: detection errors, classification errors, and localisation errors. Detection errors occur when a detector fails to recognise that an object is present, or generates a false positive detection when an object is not present. Classification errors occur when an object is successfully detected, but is assigned to the wrong semantic category. Finally, a localisation error arises when an object is correctly identified, but its bounding box is incorrectly placed around an object. The first question we therefore seek to understand is the relative impact of these types of errors on the final performance of the OFTNet network.

Following the work of Hoiem et al. (2012), the analysis began by ranking all detections in descending order of confidence, and then selecting the top N predictions, where N is the



(a) Mono3D (b) OFTNet (c) 3DOP (d) MV3D

Figure 3.12. Breakdown of detection and localisation errors in object detection algorithms. Clockwise from the top, the shaded regions represent the proportion of detections which were correctly detected ('hits'), detected but not accurately localised, detected but poorly localised, and not detected.

total number of ground truth objects present. This implies that all ground truth objects are successfully detected, there will be no false positive predictions, and also provides a fair way of comparing between different methods. Each detected bounding box was categorised according to its intersection over union score with the closest ground truth object. Following the KITTI evaluation criteria, a bounding box was considered a true-positive detection (referred to as a 'hit') if it has an IoU score of $\geq 70\%$. Boxes which have no or very little ($< 10\%$) overlap with a ground truth bounding box were considered a 'miss', and are treated as detection errors. The remaining detections were considered to be localisation errors, which were further subdivided into 'approximately localised' bounding boxes, with an overlap of $50\% \leq IoU < 70\%$; and 'poorly localised' boxes, with an intersection over union of $10\% \leq IoU < 50\%$.

A breakdown of the different types of error is shown in Figure 3.12. In addition to evaluating the OFTNet network, this analysis was repeated for three contemporary approaches for which detections were publicly available: Mono3D (Chen, Kundu, Zhang, et al., 2016), the leading monocular approach at the time; 3DOP (Chen, Kundu, Zhu, et al., 2015), a stereo method; and MV3D (Chen, Ma, et al., 2017), a LiDAR-image fusion method which has full access to 3D information.

The first interesting observation from Figure 3.12 was that for the baseline method, Mono3D, the majority of errors were caused by failures in detecting objects, whereas for the OFTNet method, localisation errors made up the majority of false-negative detections. The original hypothesis behind the OFTNet was that reasoning in the birds-eye view should primarily improve localisation, due to the metric representation and better use of 3D context. This intuition seemed to be correct, since out of the objects which were detected, OFTNet did improve the proportion which were approximately or correctly localised. However these

results also suggested that the birds-eye view representation had an important impact on detection performance, implying it may be easier to recognise objects in this space.

Also noteworthy is that the addition of stereo information, as in the 3DOP (Chen, Kundu, Zhu, et al., 2015) framework, did not seem to help either detection or localisation, as the error profiles of OFTNet and 3DOP are almost identical. This was surprising, as one might expect that the addition of the metric depth information provided by stereo triangulation would improve localisation performance. Given, however, that MV3D, which uses accurate LiDAR measurements, does receive a significant reduction in both detection and localisation errors, this may simply suggest that stereo alone does not provide accurate or dense enough depth measurements for robust 3D detection.

In the next two sections, we seek to understand these results further and determine the cause of both failures in detection and in localisation.

3.6.2 Detection errors

The first objective was to understand why certain objects in the scene failed to be detected completely. Hoiem et al. (2012) attempted to quantify this false-negative behaviour by categorising detections according to various object attributes such as bounding box size, shape and visible parts. In analogy to their work, we considered four attributes and their effects on the detection accuracy: bounding box average depth, orientation, level of occlusion, and level of truncation.

Depth

The distance of an object away from the camera was perhaps the most important factor to consider, as this largely determined the apparent size of objects in the camera frame. To investigate the effects of depth on detection performance, each object was first sorted into one of D equally-spaced bins based on its average z-coordinate. Each bin was then scored based on the recall-@-N metric, which corresponds to the percentage of ground truth objects that were in the top-N most confident predictions. Here N was chosen to be the number of ground truth objects, so that at 100% recall, all ground truth objects were detected. To decouple the detection accuracy from localisation accuracy, a relatively permissive intersection over union threshold of $IoU \geq 0.1$ was used to determine true-positive matches. The results for each of the four detection algorithms discussed in Section 3.6.1 are shown in Figure 3.13.

At relatively short depths, around the range $5m < Z \leq 20m$, all four methods achieved roughly similar recall, with the 3D LiDAR depth information used by MV3D providing little advantage over the image-only methods. The main benefit of the MV3D framework was

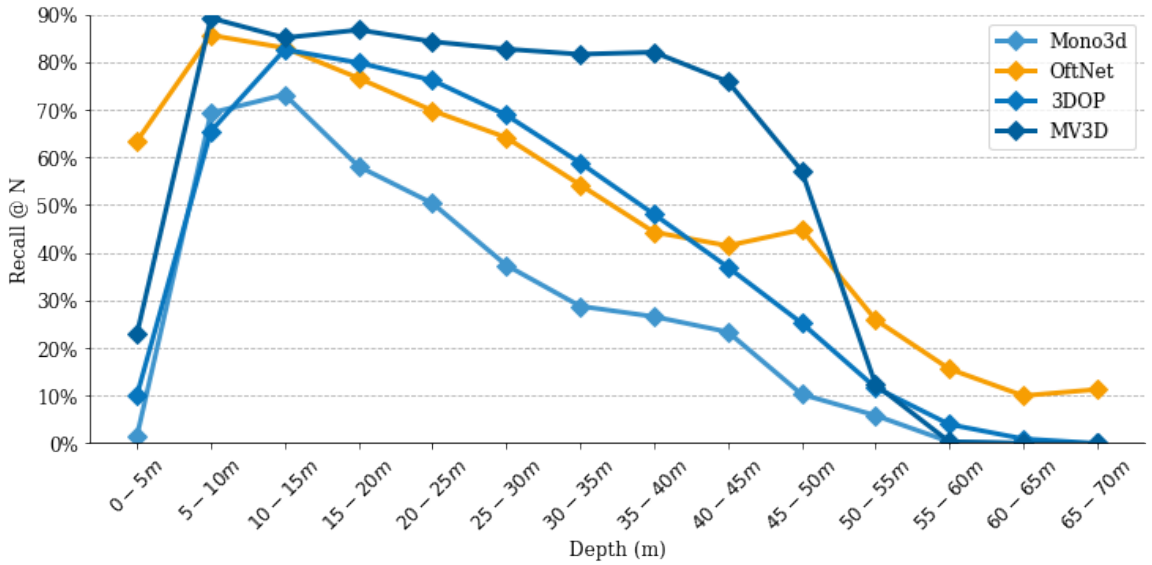


Figure 3.13. Top *N* recall as a function of distance of the object centre away from the camera.

most evident over medium range detections, between 20m and 40m, where the detection performance remains largely stable. Over this range the accuracy of the image-based systems, including OFTNet, fell gradually. This was anticipated behaviour, as modern image-based object detectors are widely known to struggle to detect small objects (Nguyen et al., 2020). Nonetheless OFTNet, which like MV3D used an birds-eye view feature representation, significantly outperformed Mono3D, and compared favourably with the stereo-based 3DOP approach.

The most interesting observation however was that at extreme distances, beyond 50m, OFTNet outperformed all other approaches, including the LiDAR-based MV3D method. This was likely to be because, although the Velodyne LiDAR sensor has a manufacturer-specified range of 120m (Glennie and Lichti, 2010), at large distances the sparsity of the point cloud increases dramatically, making it more difficult to identify objects. The input to the OFTNet was an image with a vertical resolution of 384 pixels: considerably higher than that of the Velodyne LiDAR sensor, giving the network more information to work with.

The other area in which OFTNet excelled was for objects at very close ranges, less than 5m away from the camera. Under these conditions, the object typically occupies a large proportion of the camera's field of view. Typical image-based object detectors often fail in this scenario because they have a limited receptive field size which cannot capture the full extents of the object. Part of the motivation behind the orthographic feature transform was that the receptive field size is adaptive, with a large receptive range for objects at close range

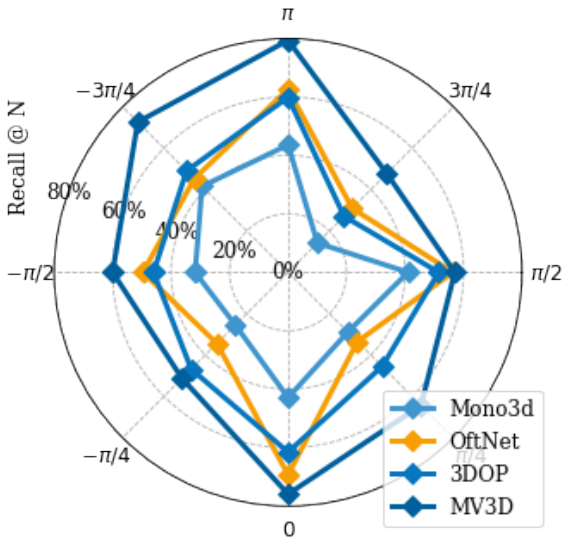


Figure 3.14. Top N recall as a function of orientation. Zero radians represents objects facing directly towards the camera, while $\pm\pi$ radians represents objects facing directly away.

and a smaller receptive field for objects at long range. This may go some way to explain the improved performance at small distances.

Orientation

In Section 3.5.4, qualitative observations showed that a common failure case for both Mono3D and OFTNet was for objects at right angles to the camera. This failure case was investigated further by sorting each ground truth object into one of 8 orientation bins, centred on orientations $\bar{\theta} \in \{-\frac{3\pi}{4}, -\frac{\pi}{2}, ..., \pi\}$ radians. We then computed the recall @ N values for each bin, which are shown in Figure 3.14.

The results in Figure 3.14 seemed to confirm the qualitative observation that both OFTNet and Mono3D performed best for objects which were either facing directly towards (0 radians) or away from ($\pm\pi$ radians) the camera. In fact, all methods experienced a significant drop in performance at oblique or perpendicular directions. This consistent loss of performance was symptomatic of the fact that the training dataset is heavily biased towards scenes of driving along straight roads, with the majority of visible vehicles oriented parallel or anti-parallel to the direction of travel.

Occlusion and truncation

The final element of detection performance that was considered was the visibility of the object within the image. For each object, the KITTI dataset provided two useful attributes: the level

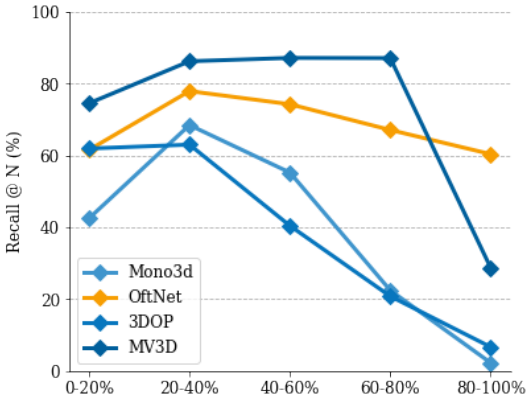


Figure 3.15. Top N recall as a function of truncation.

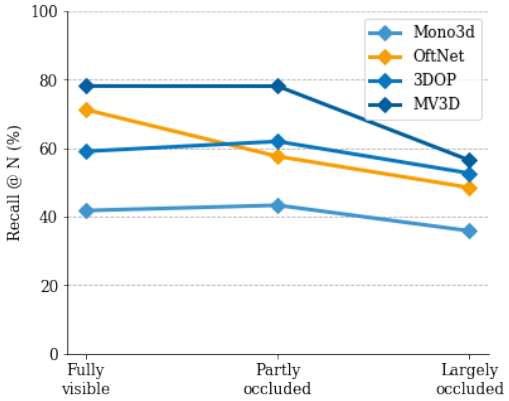


Figure 3.16. Top N recall as a function of occlusion.

of truncation i.e. what percentage of the object's bounding box is outside the boundaries of the image, and the level of occlusion i.e. to what extent the object is obscured by other objects in the foreground. The effects of these attributes on performance are visualised in Figures 3.15 and 3.16 respectively.

At first glance, the results in Figures 3.15 and 3.16 seemed somewhat contradictory, as the OFTNet ostensibly excelled in the scenario where objects were occluded by the edges of the image, but under-performed compared to other methods when objects were occluded by other objects in the scene. This implied that the reason why occluded objects were challenging was not simply the fact that the object was only partially visible, but rather the visual clutter implied by multiple objects in front of one another, making it harder to disambiguate individual instances. OFTNet may have been particularly susceptible to this since it contained more trainable parameters than competing methods so may be more prone to overfitting.

It should also be noted however that truncation is strongly correlated with depth, as objects closer to the camera are more likely to appear partially outside the field of view. In Figure 3.13 it was observed that OFTNet performed uniquely well at short distances, which may also partly explain the behaviour at high levels of truncation.

3.6.3 Localisation errors

The second factor which could have an impact on the overall accuracy of an object detection system was the effect of localisation errors, where an object is correctly detected, but the predicted bounding box is not accurately aligned to the underlying shape of the ground truth object. From the analysis in Section 3.6.1 it was found that localisation errors made up

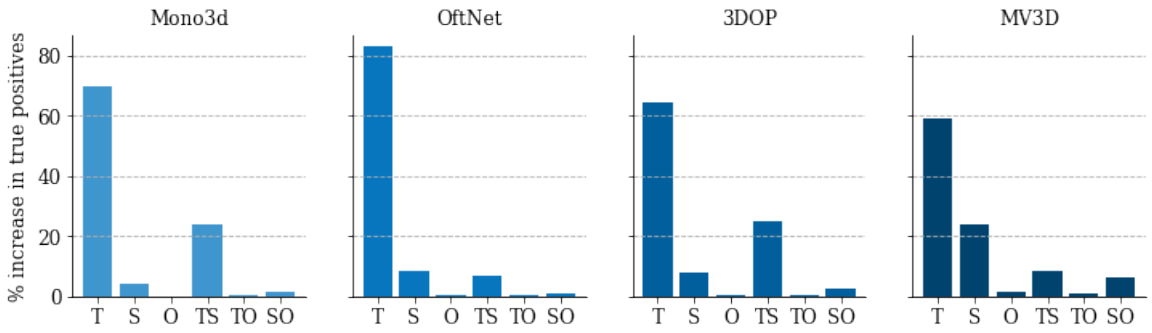


Figure 3.17. Percentage improvement in true-positive detections when translation, scale or orientation predictions are corrected to their ground truth value. T - translation, S - scale, O - orientation, TS - translation and scale, TO - translation and orientation, SO - scale and orientation.

more that 50% of the total number of failed detections in the OFTNet system. This section considers how these errors may have arisen.

In 3D object detection, localisation can be decomposed into three separate sub-tasks: translation estimation, scale estimation, and orientation estimation. The first question to consider is what is the relative importance of each of these three sub-tasks in determining overall localisation performance? This impact could be quantified by imagining that an omniscient oracle was able to predict the value of one of these attributes with perfect accuracy, and observing how the performance improved. Specifically, for each detection, the predicted translation, scale or orientation was replaced with the appropriate value from the corresponding ground truth bounding box. The proportion of poorly or approximately localised predictions which would now meet the criteria for a true positive detection ($IoU \geq 70\%$) were then computed using the same method as described in Section 3.12. This analysis also considered the importance of pairs of attributes: for example, if both translation and scale were estimated accurately, what proportion of predictions could be corrected, which were not corrected by fixing either translation or scale alone? The results of this analysis are shown in Figure 3.17.

It is clear from Figure 3.17 that across all methods the vast majority of localisation errors were caused by inaccurate translation predictions. In contrast, whilst orientation errors were visually noticable (see Figure 3.9), they seemed to have negligible impact on the overall detection performance. The importance of scale estimation varied between the different methods. For Mono3D and 3DOP, improving scale estimation had the potential to reduce the number of false-negatives by more than 20%, but only if the translation estimation was also corrected. For OFTNet on the other hand, completely solving the scale estimation problem would only confer minor improvements. This suggested either that the scale estimation in

Table 3.7. Average translation error (ATE), average scale error (ASE) and average orientation error (AOE) on the KITTI validation set. Lower numbers are better.

Method	Modality	ATE (m)	ASE	AOE (rad)
Mono3D	Mono	1.095	0.218	0.338
OFTNet	Mono	0.824	0.178	0.648
3DOP	Stereo	0.855	0.217	0.343
MV3D	LiDAR	0.220	0.150	1.374

OFTNet was particularly accurate, or that the effects of translation errors are dominated by other types of error. To seek to tease apart these two scenarios, each aspect of localisation was considered individually.

Position estimation

As identified in Figure 3.17, errors in accurately estimating the translational offset of objects had by far the largest impact on overall detection performance. In order to compare the accuracy of translation estimation between methods, Caesar, Bankiti, et al. (2019) introduced a metric called Average Translation Error (ATE), which is defined as the average Euclidean distance between each predicted bounding box centre and that of the corresponding ground truth bounding box, where one exists, i.e.

$$ATE = \frac{1}{N}\sum_{ij} a_{ij} \left| |x_i - \hat{x}_j| \right| \tag{3.32}$$

where x_i is the centre of predicted bounding box i, $\hat{x}_j$ is the centre of ground truth bounding box j, and a_{iJ} is a binary indicator which is equal to 1 if prediction i is assigned to ground truth j, 0 otherwise. The average translation error for OFTNet and other methods on the KITTI dataset is tabulated in Table 3.7. To help visualise the translation error in greater detail, Figure 3.18 shows histograms which approximate the distribution of translation errors.

The results in Table 3.7 and Figure 3.18 first of all make abundantly apparent the benefit of having access to accurate 3D measurements from LiDAR, as MV3D outperformed both OFTNet and 3DOP by almost a factor of four in terms of average translation error. In contrast, across all three image-based approaches only a very small proportion of objects could be localised to within 20cm. Interestingly 3DOP performs marginally worse than OFTNet based on translation error, suggesting that the noisy depth estimates provided by stereo triangulation may even have been detrimental to detection performance, compared to learning depth solely from monocular cues.

This conclusion is further supported by considering the anisotropy in localisation errors in different spatial dimensions. Figure 3.19 contains kernel density estimate plots showing the

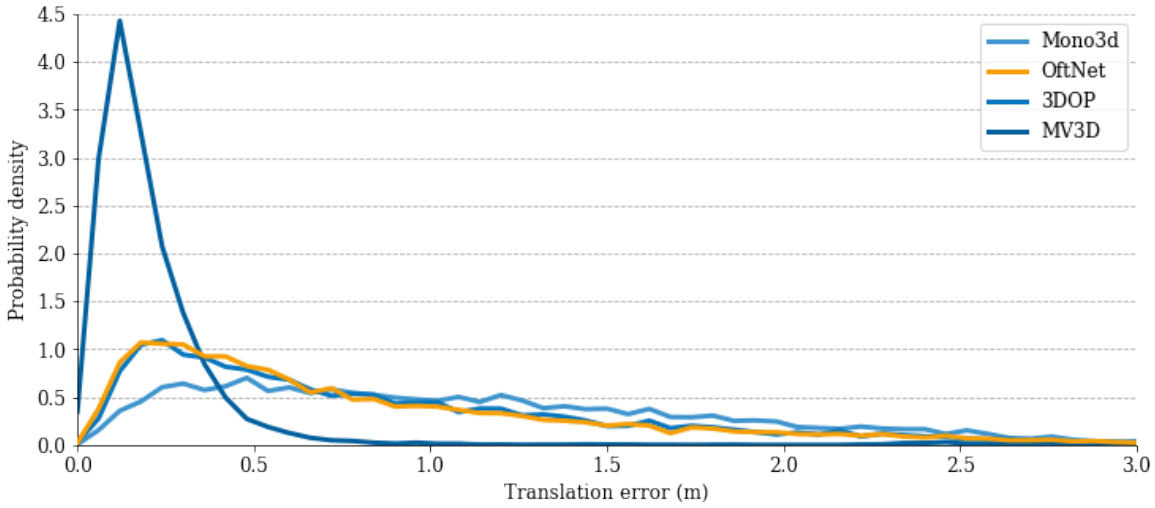


Figure 3.18. Histogram of translation error.

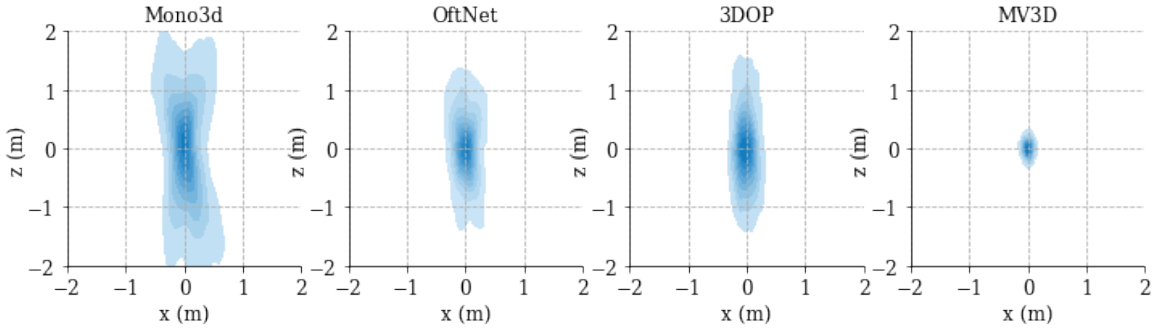


Figure 3.19. Kernel density estimate showing the distribution of position errors in the birds-eye view XZ-plane.

distribution of localisation errors in the depth (z) dimension and lateral (x) dimension. In the case of MV3D, where accurate LiDAR depth information was available, the distribution was relatively isotropic, with roughly similar error in both the depth and lateral dimensions. Given that stereo provides absolute estimates of depth based on disparity, one might have expected a similar distribution for 3DOP, perhaps somewhat scaled up to account for inaccurate depth at low disparity. In fact, errors in depth estimation were more significant in 3DOP than they were for OFTNet, suggesting that the dense monocular depth information encoded in the orthographic feature transform were at least as robust as the sparser, explicit depth estimates used in 3DOP.

Dimension estimation

In analogy to the average translation metric described above, Caesar, Bankiti, et al. (2019) introduced a second metric called the Average Scale Error (ASE), which captures the accuracy of estimating the dimensions of a predicted bounding box. The scale error is defined as one minus the intersection over union between the predicted box and ground truth box, assuming the translation and rotation of the predicted bounding box are aligned to the ground truth values. This can be expressed as

$$ASE = \frac{1}{N}\sum_{ij} a_{ij} \frac{min(w_i, \hat{w}_j)min(h_i, \hat{h}_j)min(l_i, \hat{l}_j)}{w_i h_i l_i + \hat{w}_j \hat{h}_j \hat{l}_j - min(w_i, \hat{w}_j)min(h_i, \hat{h}_j)min(l_i, \hat{l}_j)} \tag{3.33}$$

where w_i, h_i, l_i are the width, height and length of the predicted bounding box and $\hat{w}_j$, $\hat{h}_j$, $\hat{l}_j$ are the corresponding ground truth values. These values are tabulated in Table 3.7 and plot the distribution of errors in Figure 3.20.

It is under this metric that OFTNet clearly differentiated itself from previous methods, outperforming 3DOP by a considerable margin and even approaching the performance of the LiDAR-based MV3D approach. This may be interpreted as a clear indication that the transformation into the metric birds-eye view space is beneficial to accurate estimation of object dimensions. This supports the observation from Figure 3.17 that OFTNet suffered less from errors in scale estimation, but unfortunately given that errors due to translation have a much bigger impact, this has a small overall effect on localisation performance.

Orientation estimation

Despite observing in Figure 3.17 that orientation has a very small overall impact, we briefly consider a final metric: Average Orientation Error (AOE). This is defined as the smallest difference in yaw angle between the predicted and ground truth box.

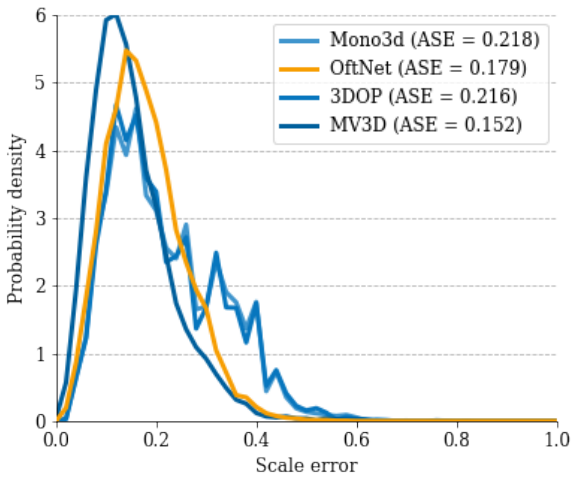


Figure 3.20. Histogram of scale error.

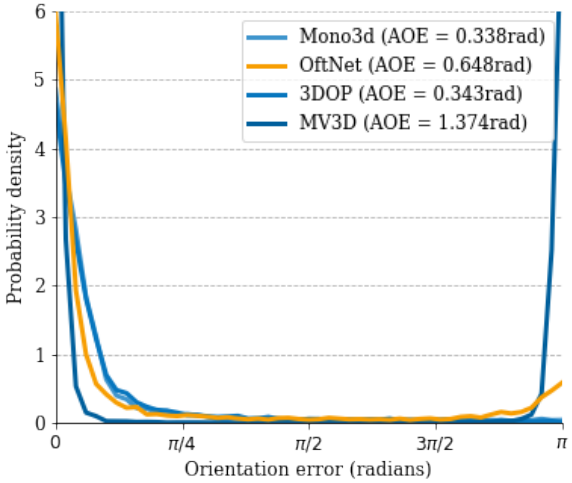


Figure 3.21. Histogram of orientation error.

Plotting the distribution of orientation error in Figure 3.21 sheds light on an interesting failure mechanism of MV3D. Although MV3D was more precise in its predictions, it often became confused between the front and back of the vehicle, resulting in not insignificant number of predictions which are incorrect by almost 180 degrees. This may have been differentiating between the front and back of a car is a highly appearance-based task, so the LiDAR point cloud alone may have been insufficient to resolve this ambiguity. Unlike Mono3D and 3DOP, the OFTNet method also suffered from this problem to some extent, suggesting that by operating in a birds-eye view space takes on some of the downsides of a LiDAR-based method, as well as the advantages. This had a large effect on the average orientation error for OFTNet and MV3D, but was not reflected in the overall average precision score as the KITTI IoU metric did not distinguish between boxes which are correctly aligned and boxes which are 180 degrees out of phase.

3.7 Conclusions

This chapter discussed the fundamental and challenging problem of monocular 3D object detection in the context of autonomous driving. The main contribution was a new object detection architecture, OFTNet, which was inspired by recent developments in single-stage 2D object detection and LiDAR-based 3D object detection. This architecture incorporated ideas from both these areas, reasoning about the world in both the perspective image space and the orthographic birds-eye view space. The core novelty of the architecture was a neural network component called the Orthographic Feature Transform, which provided a geometry-based mechanism for transforming features between the two spaces.

The efficacy of the OFTNet algorithm was extensively evaluated on the KITTI 3D object detection dataset. It was found that compared to the limited number of monocular methods available at the same time, the OFTNet performed very favourably, achieving state of the art results among comparable methods on the official KITTI benchmark. The performance of the OFTNet however fell significantly short of the success of methods which employed LiDAR or multiple sensors, and was quickly surpassed by other more recent monocular 3D object detection methods which made use of explicit estimates of depth. Experiments conducted by Caesar, Bankiti, et al. (2019) on the more recent NuScenes dataset suggested that the small size of the KITTI dataset was a major obstructing factor for the OFTNet.

To try and understand the behaviour of the OFTNet more deeply, detailed analysis of the various failure modes was conducted for the OFT and other similar methods. This exploration gave insights into the failure mechanisms of the OFTNet, as well as revealing several areas in which the philosophies of camera-only 3D object detection and reasoning in the birds-eye view were beneficial, such as the ability to identify objects at very long ranges and accurately estimate the scale of objects from images alone.

3.7.1 Limitations

At the time that much of the research presented in this chapter was conducted, monocular 3D object detection was a relatively under-explored topic. As reported in Section 3.5.5, there were only two methods that offered a suitable direct comparison to our work: the Mono3D algorithm of Chen, Kundu, Zhang, et al. (2016) as well as the network proposed by Novak (2017). Although we were able to exceed the performance of these methods, the best average precision score we were able to achieve was only 11.1% on the KITTI birds-eye view detection task. By comparison, even relatively low-performing LiDAR-based methods such as the MV3D algorithm (Chen, Ma, et al., 2017) routinely achieve >80% average precision on the same task (see Table 3.2 for more details). The analysis in Section 3.6.1 elucidated some of the reasons behind this, such as the poor detection performance of monocular compared to LiDAR-based methods at medium ranges (20 - 50m) and the inability to localise objects accurately. Results on the NuScenes dataset in Section 3.5.9 suggested that the OFTNet was relatively data inefficient, with large datasets essential for the method to generalise. The complexity of the OFT layer also incurred a heavy computation time penalty, restricting our algorithm from operating in a real-time setting. These findings suggested that our method was still far from replacing LiDAR-based counterparts in deployment-ready autonomous vehicles.

3.7.2 Current context

In the intervening years since this work was first presented, however, considerable progress has been made in closing the performance gap between image-based and LiDAR-based methods (Ku, Pon, et al., 2019; Xu and Chen, 2018; Wang, Chao, et al., 2019; You et al., 2019). The main innovation behind the success of many of these more recent methods has been to train networks to explicitly predict the depth of each pixel in the input image, rather than learn this information implicitly as was the aim of our network. For example, Xu and Chen (2018) used an unsupervised monocular depth estimation network called MonoDepth (Godard et al., 2017) to estimate per-pixel depth estimates. These estimates, together with image features, were used as input to the second stage of a Faster R-CNN-like (Ren et al., 2015) architecture. Ku, Pon, et al. (2019) predicted a 3D point cloud for each object instance, and used this information to refine the initial 3D bounding box. The most impressive improvements in monocular object detection performance have been achieved by Wang, Chao, et al. (2019), who simply applied a stand-alone monocular depth estimation network (DORN (Fu, Gong, et al., 2018)) to generate a dense 3D point cloud, and then fed this directly to an established LiDAR-based 3D object detector such AVOD (Ku, Mozifian, et al., 2018) or Frustum PointNet (Qi, Liu, et al., 2018). This simple approach, which they term "Psuedo-LiDAR" was remarkably effective, achieving 26.3% average precision on the KITTI validation set. While this performance was still significantly below that of most LiDAR based methods, it is encouraging to note that within just a couple of years the state-of-the-art performance of monocular detection algorithms has more than doubled, promoting optimism that camera-only systems may one day offer a reliable replacement for LiDAR.

Chapter 4

Semantic Map Prediction

4.1 Introduction

Over the years many researchers in computer vision (Caesar, Uijlings, et al. (2018), Sun, Kim, et al. (2013), and Heitz and Koller (2008)) have subscribed to the philosophy the world can generally be decomposed two broad concepts: “Things" (objects with a well defined shape and position), and “Stuff" (amorphous concept such as patches of grass, road or sky). The previous chapter was devoted to identifying the “Things” of a scene: detecting obstacles such as cars, bikes and pedestrians, and accurately estimating their pose and dimensions. However, these objects only represent part of the challenge for an autonomous car. Understanding the layout of the road, positions of road markings, and areas which may contain parked cars or pedestrians, are vital to achieving full autonomy. This chapter therefore turns to the problem of discovering the “stuff” components of the environment. Many works have focused on classifying these amorphous categories from the image perspective: this corresponds to the common scene-understanding task of semantic segmentation (Ulku and Akagunduz, 2019). The objective of this thesis however was to discover both the semantic content and 3D layout of the world, so we therefore represent the scene in the form of a two-dimensional map, viewed from the birds-eye view perspective, which captures the 3D arrangement of all relevant obstacles and surfaces.

The aim of the work in this chapter was, given a sequence of images, to predict a semantic 2D map of the environment. This map should contain all essential information needed for driving, so should encode the position of both static scene elements, such as roads and pavements, as well as dynamic objects such as cars, motorbikes and pedestrians. Combining these elements into a single compact representation opened the possibility that these maps could directly be used by a path planning algorithm to determine the optimal trajectory for the vehicle. The scope of these maps encompassed not just the immediate area surrounding

the autonomous platform at a given moment in time, but information accumulated over an entire journey.

These goals were accomplished through the use of a well-known framework from the robotics community known as Bayesian occupancy grid mapping (Moravec, 1989). This framework divides the world into a discrete grid of cells, and reasons probabilistically about whether each cell is occupied. For this thesis the concept of an occupancy grid was expanded to incorporate the notion of *semantic occupancy* i.e. to determine whether an instance of a given class was present at each location. To estimate the semantic occupancy state of the scene from a given camera, a novel neural network architecture, the Pyramid Occupancy Network, was introduced, which used only a single monocular image as input. The final element of this pipeline was then to combine multiple predictions from the Pyramid Occupancy Network into a global map representation using the Bayes filter algorithm; a central feature of the Bayesian occupancy grid framework.

The remainder of this chapter is divided into the following sections: Section 4.2 formulates the problem of semantic occupancy grid mapping. In Section 4.3. the Pyramid Occupancy Network, which formed the basis of a deep-learning-based approach to predicting semantic maps from images, is introduced. Section 4.5 presents evaluation of the Pyramid Occupancy Network on two large scale autonomous driving datasets: the NuScenes dataset of Caesar, Bankiti, et al. (2019) and the Argoverse dataset of Chang et al. (2019). This section focuses exclusively on the problem of predicting maps from a single image, and includes discussion of the design trade-offs and comparisons to existing works. Finally, Section 4.6 introduces the Bayesian occupancy grid framework which enables the construction of large-scale maps using sequences of images. Additional quantitative evaluation is presented on this task. Section 4.7 summarises the main findings from this chapter.

4.2 Semantic occupancy grid mapping

In this chapter, our principal goal was, given some observation of the world in the form of an image z, to construct a map-like representation m which completely described our surroundings. As discussed in Chapter 2, numerous possible representations for the map m exist, including discrete, geometric or topological maps. In this work we chose to adopt the discrete occupancy grid representation first proposed by Moravec and Elfes (1985). This representation has several key advantages, namely, it is simple to construct and is amenable to processing by a convolutional neural network. Crucially, occupancy grids are also highly suited to incremental updates over time via the Bayesian filtering algorithm: a property which we shall return to in Section 4.6.

An occupancy grid map represents the world by partitioning it into a set of discrete grid cells, i.e.

$$m = \{m_{pq}\}. \tag{4.1}$$

Each grid cell m_{pq} is a binary random variable which represents the underlying state of the world at a location (X_p, Y_q) in the birds-eye view map. A cell can either be *occupied*, indicating that an object is present at the given location, or *free*, denoting that the cell is empty and may be traversed by an autonomous vehicle or other mobile robot. The aim of the occupancy grid mapping algorithm is then to estimate the most likely map state m, given an observation z. In practice, estimating the joint posterior of the map $P(m|z)$ directly is intractable since an occupancy grid map can consist of thousands or even millions of grid cells. We therefore approximate the map as a zeroth-order Markov random field, in which each grid cell m_{pq} is assumed to be independent of its neighbours. Under this assumption, we can then approximate the joint posterior as the product of marginal posteriors:

$$P(m|z) = \prod_{p,q} P\left(m_{pq}|z\right) \tag{4.2}$$

The optimal estimate for our global map m^* can then be found by maximising the log posterior:

$$m^* = \arg\max_m \sum_{p,q} \log P\left(m_{pq}|z\right). \tag{4.3}$$

4.2.1 Semantic occupancy grids

In traditional occupancy grid mapping, the main purpose is simply to be able to distinguish between drivable and non-drivable areas. In this work, we aimed to capture a richer representation of the world: encoding different types of road features, obstacles and dynamic objects. We therefore proposed a simple extension to the standard occupancy grid representation to incorporate semantic information. Rather than encoding a single state, free or occupied, each grid cell pq is associated with a set of binary random variables $\{m^c_{pq}\}$, where the index c refers to one of several semantic categories representing road, pavement, car etc. Each of these state variables now encodes the presence or absence of the semantic concept at that location, for example $m^{car}_{pq} = 1$ if a car is present at grid location (p,q), $m^{car}_{pq} = 0$ otherwise. In contrast to other works (Lu, Molengraft, et al. (2019) and Pan, Sun, et al. (2020)) we do not assume that the semantic categories are mutually exclusive: it is perfectly conceivable that a road, pedestrian crossing, pedestrian and stroller can all co-exist at the same location. Making this assumption allows us to employ the binary Bayesian filter algorithm described in Section 4.6.1.

4.2.2 Deep inverse sensor model

In traditional occupancy grid parlance, the function used to estimate the marginal probability of occupancy $P(m_{pq}|z)$ is typically referred to as the *inverse sensor model*, in contrast to the *forwards sensor model* which measures the likelihood of an observation given the underlying state of the map. Where the observation z represents a single scalar measurement from a range sensor such as LiDAR, the inverse sensor model would often take the form of a simple hand-crafted function or a shallow neural network (Thrun, 2002). In our scenario however, the observation variable z represents a high-dimensional image will may contain millions of pixels. We therefore chose to represent the inverse sensor model as a deep convolutional neural network Φ. The network accepts as input an image z and set of camera parameters K, and predicts the probability of occupancy $\phi^c_{pq} = P(m^c_{pq} = 1|z)$ for each occupancy grid cell pq and semantic class c. The network is trained by minimising the binary cross entropy loss between the predicted $\hat{\phi}^c_{pq} \in (0,1)$ and ground truth $\phi^c_{pq} \in \{0,1\}$ occupancy probabilities:

$$\mathscr{L}\left(\phi, \hat{\phi}\right) = \sum_{p,q,c} \phi^c_{pq} \log \hat{\phi}^c_{pq} \tag{4.4}$$

The architecture and training scheme for this neural network is described in more detail in Section 4.6.

4.3 Pyramid Occupancy Network Architecture

In this section we begin by describing the convolutional neural network-based solution to the problem of semantic occupancy grid prediction, which was referred to as the Pyramid Occupancy Network (PyrOccNet). The PyrOccNet network fulfilled the role of a deep inverse sensor model Φ as described in Section 4.2.2, taking a monocular image I_k, set of camera parameters x and batch of grid cells $\{x_i\}$ as input, and estimating the probability of (semantic) occupancy for each grid cell. Since the occupancy grid was defined in the 2D birds-eye view, transferring image-based features into the birds-eye view proved a major design challenge. Once in this space, the occupancy grid map probabilities could be computed efficiently by operating convolutionally on birds-eye view features, allowing the network to share computation across different grid locations and reason about the structure of the 3D world. An overview of the Pyramid Occupancy Network is shown in Figure 4.1, while a complete network specification, including details of each individual layer type, is provided in Figure 4.4.Complete source code is also available at https://github.com/tom-roddick/mono-semantic-maps.

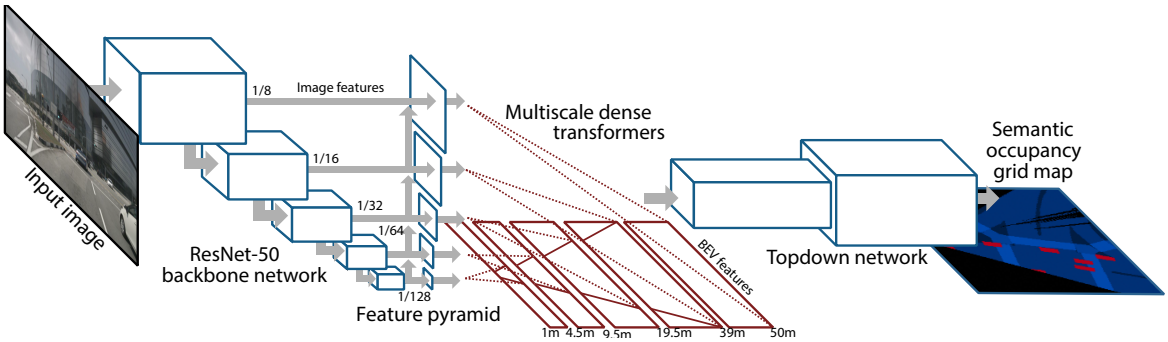


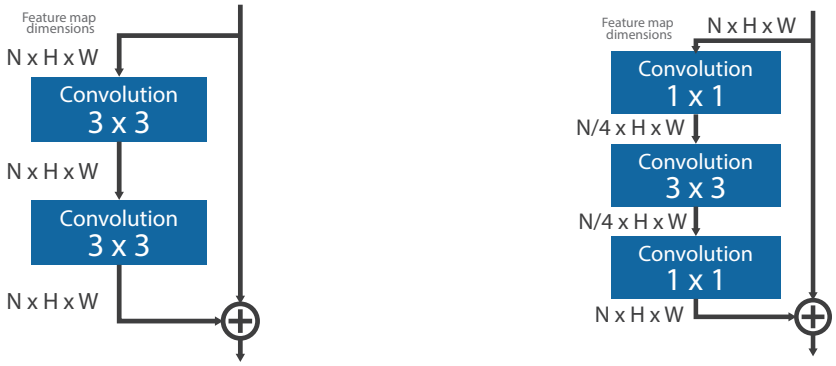
Figure 4.1. Overview of the Pyramid Occupancy Network architecture.

Architecturally, the PyrOccNet network shared many design features in common with the OFTNet architecture described in Chapter 3. Despite differences in the final output representation, both networks adopted the same broad approach to solving their respective problems: processing features in the image-space, transforming to a birds-eye view and then applying further processing in this space. However, the PyrOccNet network built on experiences from the OFT layer and OFTNet network, allowing substantial improvements to its design. Most significantly, the orthographic feature transform was extremely memory intensive, due to the need to explicitly build the full 3D voxel grid. To overcome this challenge an improved transformer module, called the *dense transformer layer*[1], was introduced, which bypassed the need for an explicit 3D representation. The resulting reduction in memory usage allowed for greater flexibility in the design of the front-end and topdown networks, which is discuss below.

The Pyramid Occupancy network consisted of four principle components:

- A frontend **feature extractor** network, based on the ResNet-50 architecture, which processed features in the image space.
- A **feature pyramid**, which upsampled low-resolution feature maps from the frontend network to provide context to shallower, higher-resolution features.
- A stack of **dense transformer layers**, which transformed the multiscale image-based features from the feature pyramid into the birds-eye view perspective.
- A **topdown network**, which processed the features in the birds-eye view and generated the final map occupancy estimates for each location in the occupancy grid.

[1]Note that the term 'transformer' was chosen in reference to the spatial transformer networks of Jaderberg et al. (2015), which first demonstrated the use of a differentiable resampling layer to transform spatial feature maps into a new representation. It does not relate to the more recent transformer networks proposed by Vaswani et al. (2017) commonly employed in sequence modelling tasks.



(a) Basic block

(b) Bottleneck block

Figure 4.2. Comparison between the two types of residual block used in the ResNet-18 and ResNet-50 feature extractors. $K \times K$ indicates the size of the convolutional kernel. N is the number of feature channels, and $H \times W$ indicates the spatial dimensions of the feature map.

This architecture is visualised in Figure 4.1, and each component is described in detail in the sections below.

4.3.1 Feature extractor

As was the case with the OFTNet, the primary purpose of the PyrOccNet's front-end feature extractor was to identify low-level features in the image and to implicitly infer the depth of each image location from the camera. In the OFTNet architecture, the choice of feature was largely restricted by two factors: GPU memory constraints; and the relatively small size of the KITTI dataset (Geiger, Lenz, and Urtasun (2012)), which made overfitting to the training set a considerable challenge. However, this work had access to considerably larger datasets (Caesar, Bankiti, et al. (2019) and Chang et al. (2019)), as well as a greater memory budget on account of the more efficient dense transformer layer. As a result, the PyrOccNet was able to employ a much deeper and more powerful feature extractor, based on a pretrained ResNet-50 network (He, Zhang, et al., 2016). Like the shallower ResNet-18 used in the previous chapter, ResNet-50 is composed of a hierarchy of convolutional network blocks with residual skip connections. ResNet-50 however makes use of an efficient 'bottleneck' residual structure, as shown in Figure 4.2b, which allows for a deeper and wider network with minimal computational and memory cost. The ResNet-50 network produces a stack of output feature maps which are a factor $d \in \{8, 16, 32\}$ smaller than the original input dimensions. In the PyrOccNet network, the architecture was extended to include two additional downsampling stages, resulting in a final set of feature maps with downsampling factors $d \in \{8, 16, 32, 64, 128\}$.

4.3.2 Dense transformer layer

The dense transformer layer sought to accomplish the same task as the orthographic feature transform from the previous chapter: to map a feature map which was computed in the image perspective, into an orthographic birds-eve-view projection. However, while the previous chapter demonstrated the effectiveness of the OFT layer and the birds-eye view philosophy in general, it also highlighted one its shortcomings: the extreme memory use required to explicitly build the 3D voxel grid.

The solution to this problem was inspired by recent works at the time such as those of Lu, Molengraft, et al. (2019) and Pan, Sun, et al. (2020), who also tackled the problem of transforming image-based features to the birds-eye view. The orthographic feature transform may be viewed as a 'geometry-based' approach, in that it predominantly used known camera geometry to determine the relationship between features in the image and locations in the birds-eye view. In contrast, Lu, Molengraft, et al. and Pan, Sun, et al., adopted a purely 'learning-based' approach. They employed auto-encoder-like architectures to directly learn the transformation from image to topdown perspective. This approach had the advantage that there was no need to explicitly construct the 3D full voxel grid. In fact, the memory consumption was considerably reduced since the image-based feature map was reduced down to a single feature vector which, in principle, summarised all the relevant information contained within the image. However, collapsing the image to a single feature vector had the downside that the spatial configuration of the image was lost, making it difficult to accurately reconstruct the 3D layout of the scene in the birds-eye view. Recovering these details required a central feature encoding with a large number of parameters, which could expose the network to overfitting.

The dense transformer layer therefore sought to combine the advantages of both approaches, providing a hybrid solution to the problem. The approach was based on the observation that, for a given point p,q in the birds-eye view with coordinates (X_p, Z_q), it is always possible to infer the corresponding horizontal pixel coordinate in the image u using camera geometry alone:

$$u = \frac{\alpha X_p}{Z_q} + c_u \tag{4.5}$$

where α is the scaled camera focal length and (c_u, c_v) the camera optical centre as described in Section 3.2. Conversely however, it is not generally possible to infer the horizontal pixel coordinate v, given by

$$v = \frac{fY}{Z_q} + c_v \tag{4.6}$$

since the height Y of the scene at location (p,q) is unknown. Moreover, the point (X,Z) might actually map to multiple v values, in the case of a vertical surface such as the wall of a building. If the object at (p,q) is occluded, there may be no values of v which map directly to this point. This asymmetry naturally suggested that a different approach should be taken between processing the horizontal rows and vertical columns of the image.

The proposed dense transformer layer therefore consisted of three steps:

Step 1 Each column of the image was encoded into a single feature vector. This resulted in a 1D feature map with the same width W as the original image. In theory, each feature vector incorporated information about both the world height Y and depth Z of every pixel along the image column.

Step 2 Decode each column feature into a ray in the birds-eye view. This resulted in a 2D feature map in the birds-eye view, where the vertical coordinate q represented the depth of each location Z_q, and the horizontal coordinate u represented the ratio $\frac{X}{Z}$.

Step 3 Resample this 2D feature map into a Cartesian coordinate system, such that each location (p,q) represented a point (X_p,Z_q) in the birds-eye view. The location in the feature map to sample from was given by coordinates $(\frac{\alpha X_p}{Z_q}+c_u,Z_q)$.

In step 1, the encoding operation was implemented by concatenating the feature vectors along the vertical image dimension, and then applying a 1D convolution kernel to the resulting 1D feature map. Similarly, step 2 reversed this operation: applying a second 1D convolution layer to the column embeddings before distributing the resulting features along the birds-eye view depth axis. An illustration of the dense transformer layer is shown in Figure 4.3.

4.3.3 Transformer pyramid

The dense transformer layer described above was built on the observation that each vertical column of the image-based feature map corresponds to a ray in the birds-eye view extending outwards from the camera centre. A problem with this assumption, however, was that while points along the rays which were near to the camera were spaced closely together, distant points were spaced much further apart. In order to densely populate the birds-eye view feature maps with features, it was therefore necessary to exploit higher-resolution feature maps to increase the ray density at large distances. Unfortunately, as well as magnifying the computational cost of the method, using higher-resolution feature maps at short distances could lead to aliasing effects and narrow objects such as lamp-posts could potentially be missed.

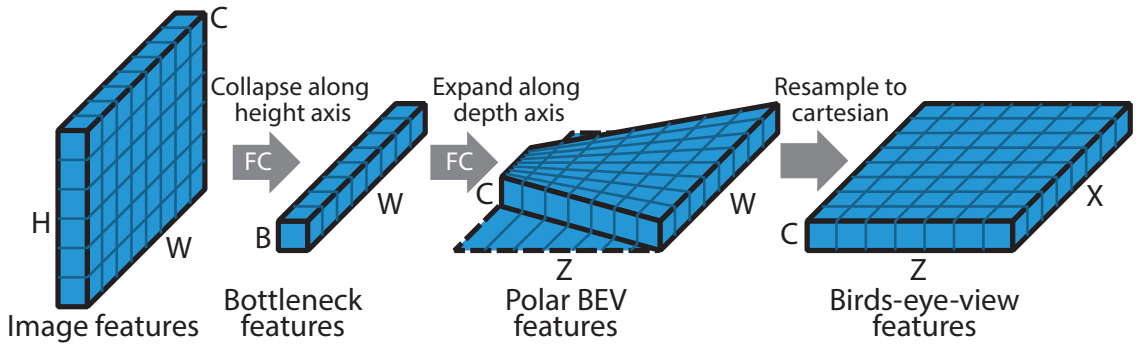


Figure 4.3. The Dense Transformer Layer. (1) Each column of the image was encoded to a single feature vector. (2) The column features were expanded along camera rays in the birds-eye view. (3) The non-Cartesian feature map was resampled into a Cartesian coordinate system, to give a single feature per birds-eye view location.

The solution to this dilemma was inspired by the image pyramid representation which is ubiquitous to techniques such as mip-mapping in computer graphics (Williams, 1983) or SIFT keypoint detection in computer vision (Lowe, 2004). The output from the front-end feature extractor was a stack of multi-scale feature maps (denoted *conv-k*, $k \in \{3,\ldots,7\}$), each of which was downsampled by a factor of 2^k relative to the original input image. To avoid aliasing effects, the birds-eye view space was divided into five zones based on the distance of points from the camera. A point (p,q) on the birds-eye view which corresponds to position (X_p,Z_q) was sampled from feature map k if it satisfied

$$k_{pq} = \left\lfloor \log_2 \left(\frac{\alpha \rho}{Z_q} \right) \right\rfloor, \tag{4.7}$$

where ρ was the resolution of the birds-eye view feature map. This ensured that when projected onto the image plane, the distance between two adjacent points (p,q) and $(p+1,q)$ was never greater than twice the resolution of the corresponding feature map k. From an information theoretic standpoint, this can be viewed as enforcing that the sampling frequency of the resampling process never exceeded the Nyquist frequency.

Feature pyramid

Unfortunately, utilising feature maps from different stages of the front-end feature hierarchy introduced a further concern. Feature maps from the early stages of the network (e.g. *conv-3* or *conv-4*) had undergone considerably fewer layers of processing than those from deeper in the network (e.g. *conv-7*). This meant that higher-resolution feature maps contained significantly

less semantic content than low-resolution features. This imbalance was addressed by adopting the approach of Lin, Dollár, et al. in their Feature Pyramid Network architecture (Lin, Dollár, et al., 2017), which used a feature pyramid to overcome this challenge. In their proposed feature pyramid, low resolution feature maps were upsampled to a larger spatial dimension before being combined with higher-resolution feature maps at the next level of the pyramid. This ensured that the high-resolution features incorporated both high-frequency information from the current pyramid level, as well as deeper semantic reasoning from lower levels. The impact of this addition to the Pyramid Occupancy Network is explored in Section 4.5.2.

4.3.4 Topdown network

The topdown network of the PyrOccNet network served a similar purpose to that within the OFTNet architecture: processing features in the birds-eye view and reasoning about the 3D structure of the scene. In design, it was also similar, consisting of a stack of residual blocks, followed by a single convolutional layer to predict the final log-odds occupancy probability at each birds-eye view location. The main departure from the architecture of Chapter 3 however was that because of the improved efficiency of the dense transformer layer, it was possible to process the birds-eye view at a higher resolution. The topdown network of PyrOccNet was therefore divided into two stages: the first which operated on feature maps at a resolution of 50cm, and the second at a resolution of 25cm. The transition between the two stages was accomplished by means of a transposed convolution layer with a stride of two, which is equivalent to applying a learned upsampling filter to the low-resolution birds-eye view feature map.

4.4 Data curation and preprocessing

4.4.1 Datasets

One of the chief limitations of studying the problem of semantic map prediction was data availability. Unlike other important scene understanding tasks such as semantic segmentation (Cordts et al., 2016; Zhou, Zhao, et al., 2017; Neuhold et al., 2017), object detection (Lin, Maire, et al., 2014; Geiger, Lenz, and Urtasun, 2012) or optical flow estimation (Butler et al., 2012; Baker et al., 2011), no publicly-available dataset focused on addressing this problem directly. Previous works have overcome the lack of data through the use of synthetically-generated images (Schulter et al., 2018; Pan, Sun, et al., 2020), coarsely aligned online maps (Schulter et al., 2018), or by using weak labels obtained automatically e.g. from stereo depth estimates (Lu, Molengraft, et al., 2019). However, the release of several new object detection

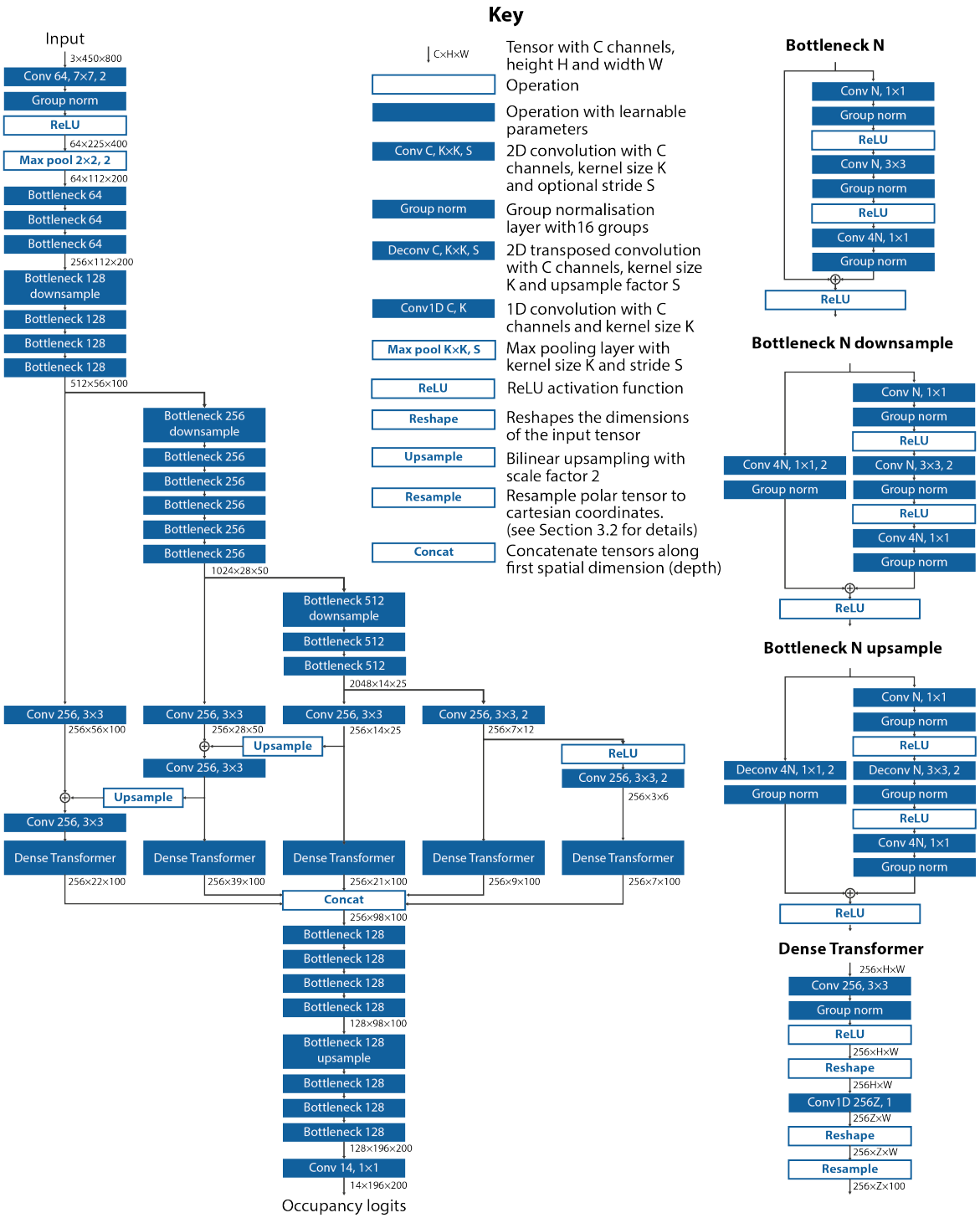


Figure 4.4. A complete specification of the Pyramid Occupancy Network architecture. Filled blocks represent neural network layers with trainable parameters.

datasets in 2019 offered a new and unique opportunity to learn a model for semantic map prediction in a supervised manner directly from annotated data. In order to be useful for this investigation however, such a dataset had to provide the following key elements:

1. **Detailed birds-eye view semantic map labels** for static elements of the scene such as road, pavement, traffic signs etc.
2. **3D bounding box annotations** for dynamic objects such as cars, pedestrians etc.
3. **Accurate localisation information** to align data from the semantic maps into the image view.
4. **Large-scale**. From our investigations in Chapter 3, it was found that despite being the largest dataset of its kind at the time, the KITTI dataset did not contain sufficient variety to train a generalisable model. A key focus of this work was to train a map detector which could generalise to unseen environments.

We identified two datasets which were able to meet this criteria: the NuScenes dataset of Caesar, Bankiti, et al. (2019) and the Argoverse dataset of Chang et al. (2019).

NuScenes

NuScenes (Caesar, Bankiti, et al., 2019) is a large-scale autonomous driving dataset, which is primarily focused on 3D object detection and sensor fusion. It consists of 1000 driving sequences captured across four locations in two cities: Boston, USA and Singapore. Each sequence is 20 seconds in length and includes data from a comprehensive sensor suite including a LiDAR sensor, five radar sensors and six 1.4 megapixel surround-view cameras. The data is annotated at 0.5s intervals with 3D bounding boxes for 23 object categories. Most crucially for the application discussed in this chapter however, it also includes highly detailed semantic maps of the routes traversed by the capture-vehicle. These consist of eleven semantic categories including road, sidewalk, pedestrian crossing and traffic light. These maps were human-annotated and were provided as vector graphics which means they could be rendered at arbitrary resolutions. They were primarily intended to be used as priors to inform object detection, but in this work were repurposed to provide ground truth labels for the map detection task. Equally importantly, the maps were paired with precise trajectory estimates, which used a Monte-Carlo localisation scheme (Chong et al., 2013) to localise the ego-vehicle within the map to an accuracy of up to 10cm.

The NuScenes dataset is split into 650 training sequences and 150 validation sequences. It also includes a private test set containing 200 sequences, however, since ground truth

annotations for this split were not available, we conducted all experiments on a permutation of the train-val set as described in Section 4.4.2.

Out of the 23 semantic object categories, we chose to focus on the 10 categories which were part of the official NuScenes detection benchmark, ignoring certain low-frequency classes such as wheelchairs and emergency vehicles. Some object classes were grouped into supersets, for example pedestrian.adult and pedestrian.child were combined into a single pedestrian class. Among the static map classes some did not have a well-defined spatial area such as traffic lights, while others are defined more by their usage than their visual appearance e.g. stopping zones. We therefore selected a subset of four classes which are important to navigation: road, sidewalk, pedestrian crossing and parking space.

Argoverse

The second dataset considered for this task was the Argoverse dataset (Chang et al., 2019), which places particular emphasis on the use of high-definition map data to aid other computer vision tasks. It is smaller than NuScenes, consisting of 65 training sequences and 24 validation sequences, which were captured in two US cities: Miami and Pittsburgh. Each sequence is between 15 and 20s in length. Argoverse also includes a full sensor suite including a LiDAR sensor and seven surround-view cameras, and provides 3D bounding box annotations for 15 object categories. The map information it provides is less detailed than that of NuScenes, consisting of just a single semantic category representing the drivable area. However the dataset also includes detailed geometrical information such as a rasterised elevation map of ground heights and vectorised lane centrelines.

4.4.2 Training and validation split selection

Since both NuScenes and Argoverse focus first and foremost on problems such as object detection and motion prediction, their data selection process emphasises interesting driving scenarios such as dense traffic or construction zones, over geographical diversity. As a result, many of the validation sequences in both datasets traversed the same areas of the maps as those used in training (see Figure 4.5b for an example). This is of little concern in assessing object detection performance, but has substantial impact on map prediction performance as it permits a learning algorithm to overfit to a particular geographical location. To remedy this situation, a refined set of dataset splits were proposed which did not share regional overlap whilst also maintaining balanced statistics over locations, environmental conditions such as day/night/rain, and object categories present. The NuScenes dataset consists of four distinct locations captured in two cities: Singapore and Boston, US. An ideal solution would have

(a) Original train-val split

(b) Custom train-val split

Figure 4.5. Ego-vehicle trajectories over the Singapore OneNorth region, one of four locations in the NuScenes dataset (Caesar, Bankiti, et al., 2019). Blue lines represent the sequences used for training, orange for validation. Shaded areas represent the approximate regions which are visible to the cameras over the sequences. In the original dataset split, there is considerable overlap between the regions used for training and validation. In our custom split, this overlap is minimised whilst still ensuring diversity of sequences.

been to use one of the locations for validation and the remaining for training. However, these locations differed so substantially in appearance and object distribution that generalising to completely unseen regions was impossible. We therefore proposed a semi-automatic procedure for dividing the existing regions into disjoint dataset splits.

We first began by generating a graph where the nodes represent a sequence of ego-vehicle poses $T_i = \bar{\mathbf{X}}_i^t$. The edge costs $d_{i,j}$ represent the minimum pairwise distance between ego-vehicle positions along a pair of trajectories

$$d_{i,j} = \min_{t,t'} \|\bar{\mathbf{X}}_i^t - \bar{\mathbf{X}}_j^{t'}\|_2 \tag{4.8}$$

This distance function was treated as infinite for trajectories captured in different cities or locations. We then decomposed the graph into connected subgraphs by cutting the graph at a distance threshold of $d_{i,j} \geq D$, and applying a depth-first search to find connected components (Hopcroft and Tarjan, 1973). We then assign each subgraph a label $l \in \{train, validation\}$, ensuring that the same stretch of road is never traversed in both the train and validation sets. Given an initial assignment, we manually permute the labels to find an optimal split which satisfies the desired ratio of train sequences to validation sequences, as well as approximately matching the distributions of sequences over locations, day/night/rainy sequences, and number of objects of each category present.

4.4.3 Label generation

Before training a deep neural network to perform the task of semantic map prediction, we first had to convert the annotations provided by our two datasets, Argoverse and NuScenes, into a suitable format. The ground truth map labels took the form of a rasterised 2D image, with each pixel (p,q) representing a grid cell at location (X_p, Y_q) in the global world space and the colour channels representing the different semantic categories $c \in C$. The mapping between points in the 3D world space $\mathbf{X}$ and points in the label image (p,q) were given by the orthographic projection matrix

$$P_{ortho} = \begin{bmatrix} \rho & 0 & x_0 \\ 0 & \rho & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{c|c} R & \mathbf{t} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \tag{4.9}$$

as described in Section 3.2. Each entry in the ground truth label image may consist of one of three values: 1 if an the semantic category is present, 0 if it is absent, and 0.5 if the status of the cell is unknown. It was desirable to mark cells as unknown if for example they are outside the field of view of the camera, in which case it would be unfair to expect our system to predict a sensible value.

The ground truth semantic occupancy labels were constructed through the union of two sources of information. For the static/background classes such as road, sidewalk etc., we made use of the birds-eye view map annotations provided by the two datasets. For dynamic objects such as cars and pedestrians, we repurposed the 3D object bounding boxes to indicate birds-eye view regions in which an object is present.

Static labels

In the NuScenes dataset, semantic map labels were provided in the form of vector graphics, with each segment of road, sidewalk etc. represented as a 2D polygon. Each map covers a large geographical area and so to make searching for relevant polygons efficient, we represented the collection of polygons as an R-Tree (Guttman, 1984), a hierarchical spatial data structure optimised for fast intersection queries. Given the 3D region of interest represented by the occupancy grid, we queried the R-Tree to find all polygons which intersected this region. We then transformed each polygon into the label image coordinate system using the projection matrix P_{ortho} (see Section 3.2.2 for definition) and rasterised it to the label image using the OpenCV software library (Bradski, 2000). In Argoverse, the drivable area labels were already provided in the form of a rasterised binary mask, so we simply resampled this mask using the pseudo-inverse of the projection matrix P^*_{ortho}.

Dynamic labels

Across both NuScenes and Argoverse, dynamic semantic categories such as cars, pedestrians, bicycles, were represented in the form of 3D bounding boxes, in the format discussed in Chapter 3. In the absence of more detailed information, we assumed that the occupancy of these dynamic objects is fully defined by the footprint of their 3D bounding boxes. We therefore represented each object as a quadrilateral in the 2D label image whose vertices $(p,q)_n^{1:4}$ are given by

$$\begin{bmatrix} p_n^1 & p_n^2 & p_n^3 & p_n^4 \\ q_n^1 & q_n^2 & q_n^3 & q_n^4 \end{bmatrix} = \pi \left(P_{ortho} \left(\frac{1}{2} R_n \begin{bmatrix} -w_n & w_n & -w_n & w_n \\ -l_n & -l_n & l_n & l_n \\ -h_n & -h_n & -h_n & -h_n \end{bmatrix} + \begin{bmatrix} X_n \\ Y_n \\ Z_n \end{bmatrix} \right) \right) \tag{4.10}$$

where π is the projection operator and $\bar{\mathbf{X}}_n = (X_n, Y_n, Z_n)^T$, $\mathbf{d}_n = (w_n, l_n, h_n)^T$ are the translation and dimensions the bounding box n, and R_n is a 3×3 rotation matrix representing the rotation of the object around the yaw axis. We then rasterised this quadrilateral to the label image using OpenCV as before.

Uncertainty labels

The static map annotations described above represented what could be accomplished with complete knowledge of the state of the world. In reality of course, only a small portion of the map is visible to the autonomous vehicle at any given time, either due to the field of view of the sensor, or because of occluding objects such as buildings and vehicles. It would be unreasonable to expect a learning algorithm to correctly predict the occupancy state for parts of the world which was hidden from it. We therefore injected a notion of uncertainty in the label images by setting the ground truth occupancy to 0.5 if a given cell was deemed to be out of sight of the input. This could occur for two reasons. Firstly, we marked as uncertain all grid cells which fall outside the cameras viewing frustrum: this could easily be determined by considering the camera focal length f and input image dimensions.

Secondly, a grid cell was also be marked as uncertain if it was occluded by another object. This information is not known *a priori*, so we proposed a simple heuristic to determine whether a grid cell is blocked. Taking advantage of the additional sensor data provided by NuScenes and Argoverse, we marked a grid cell as occluded if it had no LiDAR rays passing through it or terminating within it. This information was only available to the network at training time: it was the responsibility of the algorithm to assign uncertainty to these regions at test time.

4.5 Single Image Experiments

The first scenario discussed in this chapter is the task of predicting a birds-eye view map directly from a single monocular image. This is arguable the most challenging setting, since the scene is viewed only from a single perspective, and so there is no opportunity for the network to refine its predictions based on observations at subsequent timesteps. However this setting also provides the fairest comparison between methods, since existing works in the literature (Lu, Molengraft, et al., 2019; Pan, Sun, et al., 2020) do not explicitly tackle the multi-view fusion problem.

4.5.1 Metrics

The primary metric used throughout this evaluation was the Intersection over Union (IoU) score between the predicted and ground truth occupancy grids. The predicted occupancy probabilities were first thresholded according to a Bayesian decision boundary i.e. $\hat{m}_{pq} = 1$ if $p(m_{pq}|z_k) > 0.5$. The IoU score for a semantic class was then given by

$$IoU(c) = \frac{\sum_{p,q} \mathbb{1}\left(\hat{m}^c_{pq} = 1\right) \wedge \mathbb{1}\left(m^c_{pq} = 1\right)}{\sum_{p,q} \mathbb{1}\left(\hat{m}^c_{pq} = 1\right) \vee \mathbb{1}\left(m^c_{pq} = 1\right)} = \frac{TP}{TP + FP + FN} \tag{4.11}$$

where TP, FP and FN are the number of true positive, false positive and false negative predictions respectively.

4.5.2 Ablation study

The first set of experiments conducted sought to assess the significance of each of the four principal components of the Pyramid Occupancy Network: the backbone network, dense transformer layer, transformer pyramid and topdown network. To do so, the network was reduced to its most basic form and then each component was incrementally added back in turn. Four variants of the network were considered, as defined below:

Baseline In the simplest variant of the pyramid occupancy network, the network was reduced to just the ResNet-50 backbone network. In place of the dense transformer layer, the transformation from the image view to the birds-eye view was achieved using a simple inverse perspective mapping (further details of this mapping are given in Section 4.5.3). A pixelwise linear layer was then applied at each location on the birds-eye view to produce the final output logits. The network therefore had no opportunity to reason about the 3D spatial structure of the world: all reasoning took place in the image space.

Table 4.1. Ablation study showing the impact of each of the four principal components of the pyramid occupancy network. **Baseline** - backbone only. **D** - Dense transformer layer. **P** - Transformer pyramid. **T** - topdown network. Values are intersection over union scores on the validation set of the Argoverse dataset.

Method	Drivable	Vehicle	Pedest.	Large veh.	Bicycle	Bus	Trailer	Motorcy.	Mean
Baseline	58.5	23.4	3.9	5.2	0.5	11.0	0.4	1.9	13.1
Baseline + D	63.8	27.9	4.8	8.8	1.0	11.0	0.0	3.4	15.1
Baseline + D + P	**65.9**	30.7	7.3	10.2	1.7	9.3	**1.7**	2.2	16.1
Baseline + D + P + T	65.4	**31.4**	**7.4**	**11.1**	**3.6**	11.0	0.7	**5.7**	**17.0**

Baseline + Dense transformer In the second variant, the inverse perspective mapping in the baseline model was replaced by a single dense transformer layer. There was still no convolutional processing in the birds-eye view space, but the network could begin to reason about the heights and depths of objects in a more informed way.

Baseline + Dense transformer + Pyramid Next, the dense transformer pyramid was incorporated into the network. The ResNet-50 backbone was augmented with a feature pyramid as described by Lin, Dollár, et al. (2017), and multiple dense transformer layers were applied in the manner described in Section 4.3.

Baseline + Dense transformer + Pyramid + Topdown network Lastly, the final component of the pyramid occupancy network; the topdown network, was incorporated into the architecture. This allowed the network to reason convolutionally in the birds-eye view space and take greater account of the 3D structure of the scene. This variant was equivalent to the full pyramid occupancy architecture described in Section 4.3.

The four variants were trained and evaluated on the Argoverse train and validation splits described in Section 4.4.2. The results from this ablation study are tabulated in Table 4.1, which shows intersection over union scores for each of the Argoverse dynamic map categories.

4.5.3 Baseline methods

Having justified the design choices for the pyramid occupancy network in Section 4.5.2, the subsequent sections in this chapter will focus on comparing the pyramid occupancy network approach with existing state-of-the-art approaches in the literature. At the time of publishing Roddick and Cipolla (2020), only two prior works: the Variational Encoder Decoder (VED) approach of Lu, Molengraft, et al. (2019) and the view Parsing Network (VPN) of Pan, Sun, et al. (2020), provided suitable candidates for evaluation. A third work by Schulter et al. (2018) also tackled the birds-eye view map prediction task, however their approach relied

extensively on a simulator which was not publicly available, making it impossible to provide a fair comparison. The Pyramid Occupancy Network was therefore evaluated against the Variational Encoder Decoder network and the View Parsing Network. To provide further comparison, two additional baseline methods were implemented which reflected broad trends in the literature but for which no specific approaches have been published which tackle the specific map-prediction problem. These four approaches are outlined briefly below.

Variational Encoder Decoder

The Variational Encoder Decoder (VED) network of Lu, Molengraft, et al. (2019) was based around a modified version of a variational autoencoder (Kingma and Welling, 2013), in which a deep encoder network was used to compresses an image into a low-dimensional bottleneck feature representation. This is subsequently decoded back to an image-like representation by a second decoder network. Unlike a traditional autoencoder which seeks to reconstruct the input, the VED took advantage of the non-spatial bottleneck features to transform the input representation into a birds-eye view. In their original paper, Lu, Molengraft, et al. trained VED on weak ground truth annotations from the Cityscapes dataset (Cordts et al., 2016), which were obtained by back-projecting 2D image-based labels into 3D using approximate stereo disparity estimates. The authors claimed that the use of variational sampling at the information bottleneck makes the method robust to the imprecise nature of the ground truth. The models used in this thesis were implemented using the code provided by Lu, Molengraft, et al. In order to train on the real-world data described in Section 4.4, minor architectural changes were made to the VED network, which are detailed in the Appendix. It was trained using an equally-weighted combination of the cross entropy loss and the Kullback-Leibler divergence, which enforces that the latent feature embedding at the bottleneck approximately follows a normal distribution.

View Parsing Network

Similarly to VED, the View Parsing Network (VPN) of Pan, Sun, et al. (2020) also adopted an encoder-decoder-like architecture comprising a deep convolutional encoder network and a deconvolutional decoder network. The transformation from the image perspective to the birds-eye view perspective was achieved using a subnetwork called the 'View Relation Module': a two-layer perceptron which learned the change of perspective using fully-connected layers. The network also included a 'View Fusion Module' to combine the features from different views. However since this chapter focuses on the single-view case this module did not have any effect in practice. In the original work of Pan, Sun, et al., their network was

trained on simulated data and synthetic-to-real domain adaptation was used to evaluate the model on realistic images. For this research, ground truth annotations for real images were available, so it was possible to train the method directly on real data using binary cross entropy loss. As with the VED network, the View Parsing Network was implemented using code provided by the original authors, with minor modifications made to ensure the input and output dimensions of the VPN matched the dimensions of the new dataset.

Inverse Perspective Mapping

A widely used approach in problems which involve mapping from an image to a birds-eye view is known as inverse perspective mapping (IPM) (Hartley and Zisserman, 2003). Inverse perspective mapping involves computing a 3×3 homography matrix which represents the geometric relationship between points in the image and locations on the 2D ground plane. For this baseline method we first predicted semantic labels for each pixel in the image space by applying a state-of-the art semantic segmentation network: DeepLabv3 (Chen, Papandreou, et al., 2017), to each image in Argoverse and NuScenes. Unfortunately, neither of these datasets provide ground truth pixel label annotations in the image space, so the DeepLab network was pre-trained on the Cityscapes dataset (Cordts et al., 2016) — a large semantic segmentation segmentation dataset for urban autonomous driving. The homography matrix was then computed from the normal vector and offset of the ground plane. In order to demonstrate an upper-bound of the performance of this approach the ground plane paramters were estimated by fitting a 2D plane to 3D LiDAR points using random sample consensus (Fischler and Bolles, 1981): information which would not be available to a monocular approach *a priori*.

Depth Unprojection

The IPM baseline described above represented only a relatively crude solution to the problem since it assumed that all objects in the scene lie on a single two-dimensions plane: clearly not the case in a realistic traffic scenario. To offer a more sophisticated benchmark, a second baseline method was used which is referred to as depth unprojection. In addition to estimating the semantic label of each pixel using DeepLabV3 as before, the distance of each pixel from the camera was computed. Unprojecting each pixel to a 3D point using the camera calibration parameters resulted in a semantic 3D point cloud in the ego-vehicle coordinate system. Each point was then re-projected to the ground plane to obtain the semantic label for each location in the birds-eye view. Multiple points being projected to the same birds-eye view grid cell resulted in multiple semantic labels being present at that location, while cells with no

Table 4.2. Per-class intersection-over-union scores on the Argoverse validation set. An asterisk (*) indicates classes which are present in the CityScapes dataset. CS Mean is the average over these classes only. Best scores are shown in **bold**.

Method	Drivable*	Vehicle*	Pedest.*	Large veh.	Bicycle*	Bus*	Trailer	Motorcy.*	Mean	CS Mean
IPM	43.7	7.5	1.5	-	0.4	7.4	-	0.8	-	10.2
Depth Unproj.	33.0	12.7	3.3	-	1.1	**20.6**	-	1.6	-	12.1
VED	62.9	14.0	1.0	3.9	0.0	12.3	**1.3**	0.0	11.9	15.0
VPN	64.9	23.9	6.2	9.7	0.9	3.0	0.4	1.9	13.9	16.8
PyrOccNet	**65.4**	**31.4**	**7.4**	**11.1**	**3.6**	11.0	0.7	**5.7**	**17.0**	**20.8**

projected points were treated as unknown. In a practical system, the depth estimates at each pixel could be obtained using a monocular depth estimation network such as DORN (Fu, Gong, et al., 2018), but to test the upper-bounds of this style of approach we took advantage of highly accurate depth measurements from on-board LiDAR. Such LiDAR observations are however naturally sparse, so a common depth in-painting approach was applied to fill the gaps between observations (Levin et al., 2004).

4.5.4 Evaluation on the Argoverse Dataset

Having explored the impact of each of the main four components of the network in the ablation study described in Section 4.5.2, we next conducted a large-scale comparison between the final model (referred to subsequently as PyrOccNet) and the four baseline methods described in Section 4.5.3. The first evaluation was conducted on the Argoverse dataset using the same training-validation splits used in the ablation study. The results of this experiment are tabulated in Table 4.2. The table shows per-class intersection over union scores as well as the macro-average of the IoU scores, firstly across all classes and secondly across only the classes which are also present in the Cityscapes dataset. This allows for a fair comparison against the IPM and depth unprojection baseline which were trained on Cityscapes data.

From the summary statistics in Table 4.2 it is clear that the PyrOccNet approach outperformed all of the existing approaches, including the state-of-the-art VED and VPN methods, by a considerable margin. The Argoverse dataset includes a single map-based class representing the *drivable* area of a scene, and on this class the performance of the three end-to-end methods (VED, VPN and PyrOccNet) was roughly equivalent. The area where PyrOccNet truly excelled however was on the classes representing smaller objects, such as *vehicle*, *bicycle* and *pedestrian*. Indeed on the *pedestrian* and *bicycle* classes, the performance of Lu, Molengraft, et al.'s VED method breaks down completely. Surprisingly however, VED outperformed both PyrOccNet and the VPN approach on the *bus* and *trailer* classes. This

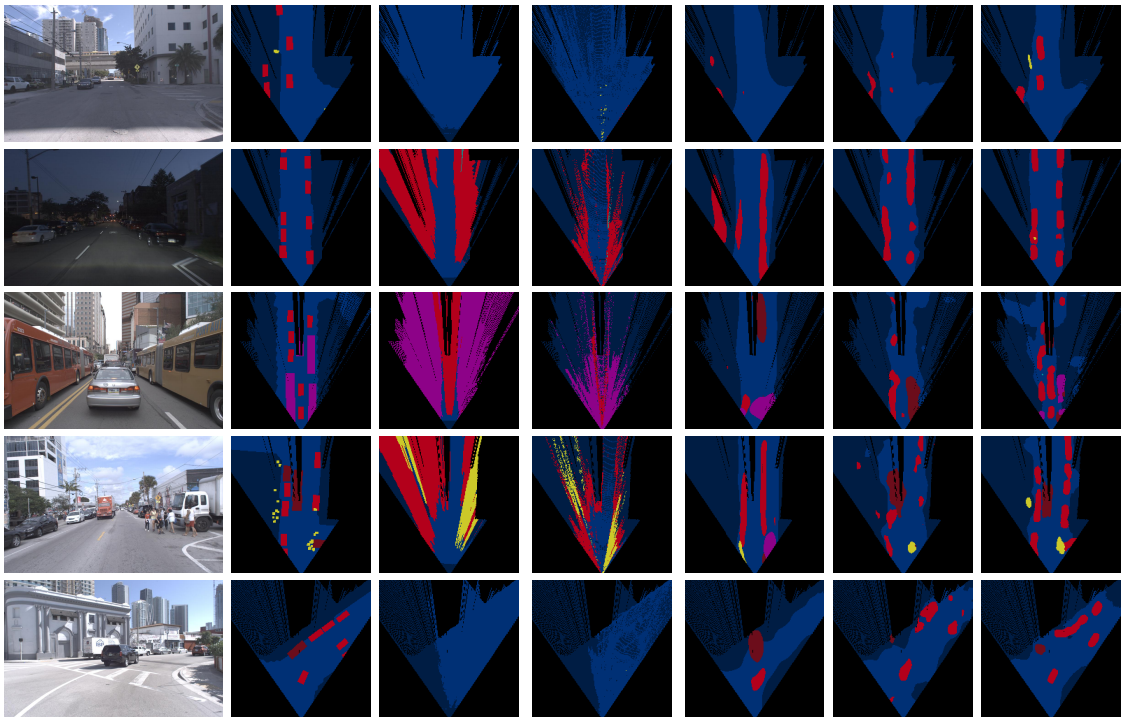


Figure 4.6. Qualitative comparison of results on the Argoverse validation set.

could be attributed to the relatively small size of the Argoverse dataset and the fact that VED contains significantly fewer parameters than VPN and PyrOccNet, resulting in a lesser degree of overfitting on the relatively rare *bus* and *trailer* categories. Similarly, perhaps the biggest anomaly in the results on the Argoverse dataset was the fact that the depth unprojection baseline, which performed poorly across every other category, achieved the best results on the *bus* category. However, while the Argoverse training split contained only 28 unique bus instances, the Cityscapes dataset (which was used to train the DeepLabv3 network at the heart of the depth unprojection method) is considerably larger-scale and contains significantly more examples of buses. This discrepancy is likely to explain the unexpectedly high score for the depth unprojection method.

To expand on the quantitative evaluation presented above, Figure 4.6 visualises the five approaches. From these results, the weaknesses of the baseline IPM and depth unprojection methods quickly become apparent. The IPM approach assumed that all objects lay at ground level, and as a result the *drivable* surface class was reasonably well captured. However pixel labels corresponding to objects which lay above the ground, such as the cars in row 2, were artificially stretched along camera projection rays, causing the method to significantly overestimate the prevalence of such classes. The depth unprojection approach on the other hand suffered significantly from the sparsity of the point cloud: despite densifying the

LiDAR points as described in Section 4.5.3, many points on the ground plane did not have a corresponding pixel in the image.

In comparison to the IPM and depth unprojection methods, the three end-to-end approaches (VED, VPN and PyrOccNet) were able to achieve a much more accurate reconstruction of the scene, both in terms of the static road geometry (see Figure 4.6, row 1, 2), and in terms of the placement of objects within the scene (row 2, 5). The main difference between the three methods was in the level of detail that they were able to capture. VED in particular was unable to distinguish the gaps between individual cars as seen in row 2 of Figure 4.6. It was also unable to detect smaller objects such as the clusters of pedestrians in row 4. These failures may be attributed to the dense fully-connected bottleneck in the centre of the VED encoder-decoder architecture, which removes the explicit spatial relationships between features in the input and output, resulting in finer details being lost. The PyrOccNet approach on the other hand was much more successful at distinguishing individual cars and accurately localising clusters of pedestrians (rows 2, 4). All three end-to-end methods suffered a failure case in row 3 where the buses were incorrectly classified as either trucks or cars, validating the quantitative observation discussed above.

4.5.5 Evaluation on the NuScenes Dataset

As discussed above in Section 4.5.4, many of the challenges encountered by the proposed PyrOccNet method and other learning-based baselines arose due to the relatively small size of the Argoverse dataset, consisting of just 65 training sequences. Argoverse is also relatively limited in terms of the available map categories, with just a single static category representing the drivable area. We therefore conducted a much larger-scale evaluation on the NuScenes dataset, which consists of 650 training sequences and 148 validation sequences. It also includes a larger variety of static map categories including pedestrian crossings, pavements and parking spaces. The results of this evaluation are presented in Table 4.3.

The overall trend in Table 4.3 largely mirrored the results on the Argoverse dataset, where the three end-to-end methods achieved significantly better scores than the image-based baselines, with the PyrOccNet network emerging as the most successful. The anomaly discussed in Section 4.5.4 with the *bus* class was corrected, with VPN and PyrOccNet outperforming the depth unprojection method by a considerable margin. The most notable observation was that again the performance of the VED method collapsed completely for many object categories: in particular for small object classes like *pedestrian* and *motorcycle*, but also for larger but relatively rare classes such as *truck* and *construction vehicle*. This may again be attributed to the fully-connected bottleneck layer, which reduced the localised spatial information available to the network.

Table 4.3. Per-class intersection-over-union scores on the NuScenes validation set. An asterisk (*) indicates classes which are present in the CityScapes dataset. CS Mean is the average over these classes only. Best scores are shown in **bold**.

Method	Drivable*	Ped. crossing	Walkway*	Carpark	Car*	Truck	Bus*	Trailer	Constr. veh.	Pedestrian*	Motorcycle*	Bicycle*	Traf. Cone	Barrier	Mean	CS Mean
IPM	40.1	-	14.0	-	4.9	-	3.0	-	-	0.6	0.8	0.2	-	-	-	9.1
Depth Unproj.	27.1	-	14.1	-	11.3	-	6.7	-	-	2.2	2.8	1.3	-	-	-	9.4
VED	54.7	12.0	20.7	13.5	8.8	0.2	0.0	7.4	0.0	0.0	0.0	0.0	0.0	4.0	8.7	12.0
VPN	58.0	27.3	29.4	12.9	**25.5**	**17.3**	20.0	**16.6**	4.9	7.1	5.6	4.4	4.6	**10.8**	17.5	21.4
PyrOccNet	**60.4**	**28.0**	**31.0**	**18.4**	24.7	16.8	**20.8**	**16.6**	**12.3**	**8.2**	**7.0**	**9.4**	**5.7**	8.1	**19.1**	**23.1**

Differently to the Argoverse results, the performance gap between the VPN and PyrOccNet networks was much closer, with VPN actually outperforming PyrOccNet slightly on some important object categories such as *car* and *barrier*. Like VED, VPN also featured a fully connected bottleneck at the centre of an encoder-decoder architecture. The bottleneck of VPN was considerably larger than that of VED however, which may have made it able to retain more information about the spatial layout of the scene. This denser bottleneck came with the expense of a greater number of trainable parameters, which explains why VPN performed better on the much larger NuScenes dataset compared to its performance on Argoverse.

In contrast to both VED and VPN, the PyrOccNet network aimed to preserve as much spatial information as possible through the use of a set of dense transformer layers (Section 4.3.2). The success of this approach was demonstrated by the performance on small object classes such as *pedestrian*, *motorcycle* and *bicycle* which considerably exceeded that of other methods. PyrOccNet however also achieved the highest accuracy over the four static map categories, which included *drivable area* and *pedestrian crossing*. This hinted at the ability of the network to perform well across multiple scales: handling both small objects such as pedestrians and large amorphous regions such as the drivable area of large intersections. This was an explicit aim of the transformer pyramid described in Section 4.3.2.

A qualitative overview of the results on the NuScenes dataset is show in Figure 4.7. As was the case with the Argoverse dataset, the IPM method was relatively accurate in reconstructing the flat geometry of the scene, such as the location of the road surface and pavements, but any objects which extended above the ground plane appear stretched along the lines of projection, such as the pedestrian in row 6. The depth unprojection suffered from a similar problem, where inaccuracies in the image-based segmentation led to a 'bleeding' effect around the boundaries of an object in the image, where road or other background classes were incorrectly assigned to a foreground class. This effect highlighted one of the

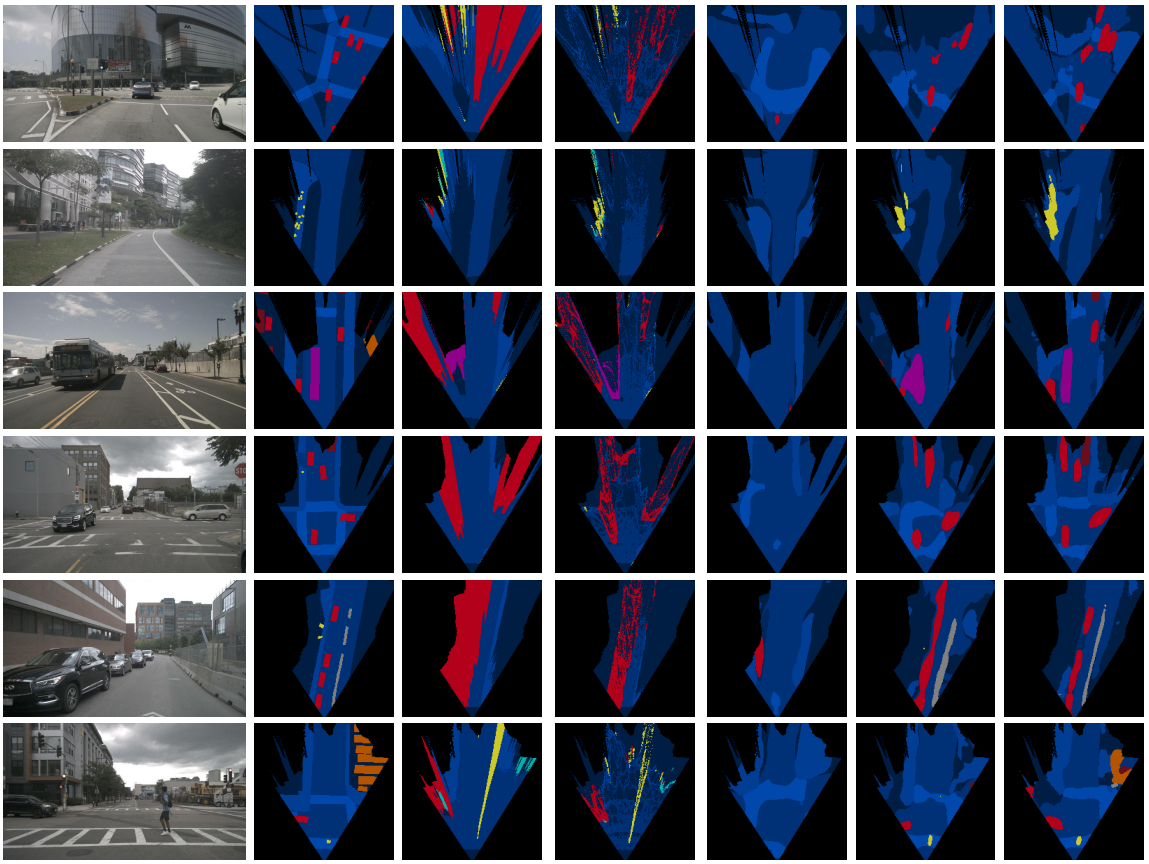


Figure 4.7. Qualitative comparison of results on the NuScenes validation set.

main motivations for operating in the birds-eye view: points which are close together in the image are not necessarily close together in 3D space.

The VED method of Lu, Molengraft, et al. typically outperformed the naïve baselines on the static map classes *drivable area* and *walkway*, and this is apparent from the visualisations in Figure 4.7, where VED produced a much smoother and more aesthetically pleasing result, although it was interesting to note that VED significantly underestimated the distance to the junction in row 2. The failure of the method on the smaller dynamic classes is however clear, as the network failed to reliably detect either the pedestrians in rows 2 and 6 or the cars in rows 1, 3 and 4.

In keeping with the quantitative results, the visualisations for the VPN and PyrOccNet methods were qualitatively relatively similar. The PyrOccNet network appears to have produced slightly sharper boundaries around individual objects, such as the bus in row 3 and the row of parked cars in row 4. On the other hand, the network overpredicted the extents of pedestrians in row 2 compared to VPN. Since the boundaries of these objects represent only

a small fraction of the total number of object pixels, these qualitative variations may not have been well reflected in the intersection over union scores used to assess the methods above. The PyrOccNet's superior quantitative performance on the static classes is however apparent from the visualisations, for example in row 4 of Figure 4.7 where the network accurately captured the complex road geometry including pavements and pedestrian crossings.

4.5.6 Cross-dataset Generalisation

An important aspect of the performance of a deep learning method is its ability to generalise to novel and unseen scenarios. This capability in the PyrOccNet network was assessed by conducting a cross-dataset analysis, where a model trained on the NuScenes dataset was evaluated on the Argoverse dataset, and vice versa. Table 4.4 shows intersection over union scores for the VED, VPN and PyrOccNet methods across the four possible permutations of training and evaluation datasets. Since Argoverse contains fewer semantic classes, predictions by models trained on the NuScenes dataset were mapped to the Argoverse categories to allow a comparison between different datasets.

As one might expect, there was typically a considerable drop in performance across all methods when the training and evaluation datasets differed. For example, the mean intersection over union score for the PyrOccNet method fell from 17% when trained and evaluated on the Argoverse dataset, to 14.1% when trained on NuScenes and evaluated on Argoverse. The NuScenes and Argoverse datasets were captured in drastically different geographical locations: the former in Boston, USA and Singapore, and the latter in Pittsburg and Miami, USA. These locations featured drastically different vegetation, climate and architecture, and since each model was trained on only two cities, it is unsurprising that the network was unable to generalise robustly across this vast appearance gap. Nonetheless, despite this drop in performance when transferring between datasets, it was encouraging to see that the effectiveness of the methods did not collapse completely. In the first half of the table for example, the IoU scores for the PyrOccNet method trained on NuScenes were comparable to those of the VPN and VED networks trained on Argoverse, suggesting it still provided a useful description of the objects and layout of road scenes.

Examining these results in further detail, it can be seen that much of the drop in performance on the Argoverse validation set in particular was due to the larger, dominant classes such as *drivable area* and *vehicle*. In some cases, some of these losses were partially offset by the performance on rarer classes such as *bicycle* and *trailer*. Unexpectedly, the VPN model trained on NuScenes actually outperformed the same model when trained on Argoverse, in spite of the domain gap. This could largely be explained by significantly improved perfor-

Table 4.4. Cross-dataset generalisation performance on the NuScenes and Argoverse validation sets. The second column indicates which dataset was used to train each method, third column indicates the dataset used for evaluation. Results from Tables 4.2 and 4.3 are reproduced in the first and third section of the table for convenience.

Method	Dataset		Intersection over Union (%)								Mean
	Train	Eval.	Drivable	Vehicle	Pedest.	Large veh.	Bicycle	Bus	Trailer	Motorcy.	IoU
VED	Argo.	Argo.	62.9	14.0	1.0	3.9	0.0	12.3	1.3	0.0	11.9
VPN			64.9	23.9	6.2	9.7	0.9	3.0	0.4	1.9	13.9
PyrOccNet			65.4	31.4	7.4	11.1	3.6	11.0	0.7	5.7	17.0
VED	NuSc.	Argo.	52.6	5.2	0.0	0.4	0.0	0.0	0.0	0.0	7.26
VPN			56.0	16.0	4.3	6.3	2.5	25.2	2.1	2.0	14.3
PyrOccNet			51.6	16.0	2.0	3.9	1.4	31.9	4.7	1.4	14.1
VED	NuSc.	NuSc.	54.7	8.8	0.0	0.1	0.0	0.0	7.4	0.0	8.9
VPN			58.0	25.5	7.1	13.1	4.4	20.0	16.6	5.6	18.8
PyrOccNet			60.4	24.7	8.2	12.7	9.4	20.8	16.6	7.0	20.0
VED	Argo.	NuSc.	46.9	10.1	0.8	2.9	0.1	0.3	0.1	0.1	7.7
VPN			44.8	14.8	4.1	3.7	2.0	0.6	0.0	1.9	9.0
PyrOccNet			45.3	15.4	2.3	6.4	1.1	4.9	0.0	0.0	9.4

mance on the *bus* class, supporting the conclusion from Section 4.5.4 that the Argoverse training set contains insufficient examples of this class.

4.6 Multiple Frame Experiments

The previous section demonstrated the effectiveness of the proposed PyrOccNet neural network on the problem of predicting a map-like representation of a scene from a single image. This representation was useful in that it captured a snapshot of the immediate surroundings of the autonomous vehicle in time, including the locations of dynamic agents in the scene such as other cars or pedestrians; providing vital information for path planning and obstacle avoidance. In many applications however it is also useful to obtain a larger scale representation of the world, for example in the creation of high-definition maps for use by other autonomous vehicles or for registration against an existing map. Fortunately, by formulating the map prediction problem in terms of the semantic occupancy grid framework as described in Section 4.2, we were able to take advantage of the binary Bayes filter algorithm (Thrun, 2002), which provides a simple and elegant mechanism to combine the single frame predictions from Section 4.5 to produce comprehensive large-scale maps. In this section we will introduce the binary Bayes filter algorithm and discuss how the PyrOccNet network from this chapter was used to generate large-scale maps, providing qualitative and quantitative evaluation.

4.6.1 Binary Bayes Filtering

One of the major advantages of the probabilistic occupancy grid framework described in Section 4.2 is that it provides a powerful and simple mechanism for combining map predictions over multiple observations. This can be achieved by means of the *binary Bayes filter* (Thrun, 2002); a recursive algorithm which updates the current estimate of the map state m using the most recent sensor observation z_k at time k and the state estimate based on the set of previous observations $z_{1:k-1}$.

To derive the binary Bayes filter update equation, recall from Section 4.2 that the aim of the occupancy grid approach is to estimate the marginal posterior probability of occupancy $P(m^c_{pq}|z)$ given a single observation z. If we are instead presented with a sequence of observations $z_{1:k}$, we can update our previous estimate $P(m^c_{pq}|z_{1:k-1})$ by considering Bayes rule:

$$P(m^c_{pq}|z_{1:k}) \propto P(m^c_{pq}|z_{1:k})P(z_k|m^c_{pq}) \tag{4.12}$$

The term $P\left(z_k|m^c_{pq}\right)$ is known as the *measurement model* and describes how observations such as LiDAR returns or image patches are generated from the map state m^c_{pq}.. Applying Bayes rule a second time to $P(z_k|m^c_{pq})$ further implies that

$$P(m^c_{pq}|z_{1:k}) \propto P(m^c_{pq}|z_{1:k})\frac{P(m^c_{pq}|z_k)}{P(m^c_{pq})} \tag{4.13}$$

where $P(m^c_{pq}|z_k) = \Phi^c_{pq}(z_k)$ represents the output from the inverse sensor model (which in our case is the PyrOccNet network described in Section 4.3), and $P(m^c_{pq})$ is the prior probability of occupancy for a given semantic class c (which is computed empirically over ground-truth examples). In order to remove the coefficients of proportionality, it is helpful to consider the ratio between the posteriors $P(m^c_{pq} = 1|z_{1:k})$ and $P(m^c_{pq} = 0|z_{1:k})$:

$$\frac{P(m^c_{pq} = 1|z_{1:k})}{P(m^c_{pq} = 0|z_{1:k})} = \frac{P(m^c_{pq} = 1|z_{1:k-1})}{P(m^c_{pq} = 0|z_{1:k-1})}\frac{P(m^c_{pq} = 1|z_k)}{P(m^c_{pq} = 0|z_k)}\frac{P(m^c_{pq} = 0|z_{1:k})}{P(m^c_{pq} = 1)} \tag{4.14}$$

or equivalently

$$\frac{P(m^c_{pq} = 1|z_{1:k})}{1 - P(m^c_{pq} = 1|z_{1:k})} = \frac{P(m^c_{pq} = 1|z_{1:k-1})}{1 - P(m^c_{pq} = 1|z_{1:k-1})}\frac{P(m^c_{pq} = 1|z_k)}{1 - P(m^c_{pq} = 1|z_{1:k})}\frac{1 - P(m^c_{pq} = 1)}{P(m^c_{pq} = 1)} \tag{4.15}$$

This equation can be more conveniently expressed using a *log odds ratio* representation, which for a binary random variable X is defined as

$$l(X) = \log \frac{P(X=1)}{P(X=0)} = \log \frac{P(X=1)}{1-P(X=1)} \tag{4.16}$$

In our scenario, the log odds ratio of the posterior is given by

$$\begin{aligned} l_{1:k}(m^c_{pq}) &= \log \frac{P\left(m^c_{pq}=1|z_k\right)}{1-P\left(m^c_{pq}=1|z_k\right)} + \log \frac{P\left(m^c_{pq}=1|z_{1:k-1}\right)}{1-P\left(m^c_{pq}=1|z_{1:k-1}\right)} - \log \frac{P\left(m^c_{pq}=1\right)}{1-P\left(m^c_{pq}\right)=1} \\ &= l_k(m^c_{pq}) + l_{1:k-1}(m^c_{pq}) - l^c_0 \end{aligned} \tag{4.17}$$

Here the terms $l_{1:k}(m^c_{pq})$ and $l_{1:k-1}(m^c_{pq})$ represent the current and previous estimate of the map state, while l^c_0 represents the prior likelihood that an object of class c is present before any observations have been made. The term $l_k(m^c_{pq})$ meanwhile represents the output from the single-frame inverse sensor model, which conveniently corresponds to the pre-sigmoid outputs of the PyrOccNet network described in Section 4.3. Given the log odds expression for the posterior $l_{1:k}(m^c_{pq})$, we can recover the posterior probabilities as

$$P\left(m^c_{pq}=1|z_{1:k}\right) = \frac{1}{1+\exp\left(l_{1:k}(m^c_{pq})\right)} \tag{4.18}$$

By applying the simple update rule in Equation 4.17, the single-frame map predictions from Section 4.5 could be readily accumulated over time to construct large-scale composite maps and to resolve ambiguities due to occlusions or missing information.

4.6.2 Calibration

An implicit assumption of the Bayes filter algorithm described above is that the output of the inverse sensor model $\Phi pq^c(z_k)$ represents the true posterior probability of occupancy $P(m_{pq}=1|z_k)$. In other words, if the network predicted a grid cell to be occupied with confidence 0.9, we would expect it to be correct 90% of the time. We can formalise this as

$$P(m_{pq}=1|\Phi^c_{pq}(z_k)) = \Phi^c_{pq}(z_k) \quad \forall p,q \tag{4.19}$$

A predictive model which fulfils this property is referred to as a calibrated model. Unfortunately, a widely observed phenomenon of deep neural networks is that predicted network confidence scores are often poorly calibrated estimates of the true posterior (Guo et al., 2017).

In particular, the cross-entropy loss function often produces results at the extreme ends of the probability scale, resulting in predictions being overly confident. This presents a problem for the Bayesian occupancy approach because over-confident predictions in areas where the network is uncertain (such as parts of the scene which are occluded) can override more reliable predictions at other timesteps.

Reliability diagrams

We can study the extent of this miscalibration problem through the use of reliability diagrams (DeGroot and Fienberg, 1983; Niculescu-Mizil and Caruana, 2005). A reliability diagram is a plot of the true probability $p\left(m^c_{pq} = 1 | \Phi^c_{pq}(z_k)\right)$ against the network's predicted confidence $\Phi^c_{pq}(z_k)$. Given a set of predictions from the network and corresponding ground truth occupancies, we first assigned each prediction to one of N histogram bins, such that $b_n \leq \Phi^c_{pq}(z_k) < b_{n+1}$, where b_n is the lower boundary of the nth bin. We could then obtain an approximation to the conditional probability $p\left(m_{pq} = 1 | \Phi^c_{pq}(z_k)\right)$ by computing the proportion of samples α^c_m in each bin which correspond to occupied cells of semantic class c:

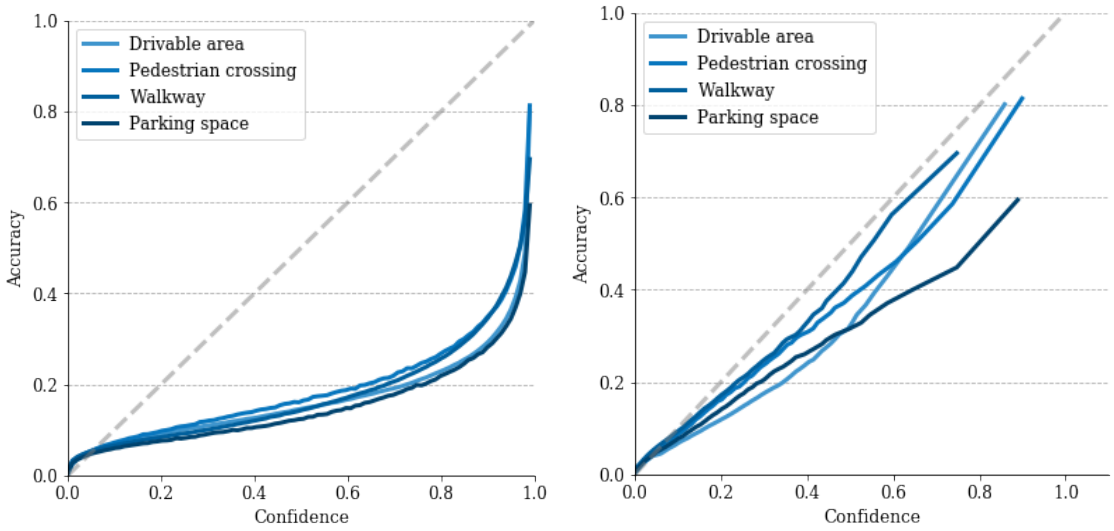
$$\alpha^c_n = \frac{\sum_{pq} \mathbb{1}\left(m_{pq} = 1\right) \mathbb{1}\left(b_n \leq \Phi^c_{pq}(I_k) < b_{n+1}\right)}{\sum_{pq} \mathbb{1}\left(b_n \leq \Phi^c_{pq}(z_k) < b_{n+1}\right)} \tag{4.20}$$

This was equivalent to computing the accuracy of predictions within each bin.

Figure 4.8a shows a reliability diagram for the PyrOccNet network, evaluated over the NuScenes validation set. For a perfectly calibrated network, one would expect the model confidence to be perfectly correlated with prediction accuracy, which corresponds a diagonal line on the reliability diagram. From Figure 4.8 it is evident that the calibration curves diverged significantly from the diagonal, indicating that the model-predicted confidences significantly overestimated the true likelihood that a given grid cell contained an object.

Isotonic regression

To obtain better-calibrated estimates of the true posterior, a number of calibration procedures have been proposed, an overview of which is provided by Guo et al. (2017). For this work, we adopted the isotonic regression approach of Zadrozny and Elkan (2001) to calibrate the PyrOccNet outputs. Isotonic regression is a non-parametric calibration approach which generates a mapping from predicted confidences $\Phi^c_{pq}(z_k)$ to calibrated confidences as a piecewise-constant monotonic function $f\left(\Phi^c_{pq}(z_k)\right)$. The form of f was obtained by first computing a reliability diagram as described above using a held-out subset of the training data. The value q_n of the piecewise-constant function f corresponding to histogram bin n is



(a) Before calibration

(b) After calibration

Figure 4.8. Reliability diagrams showing model accuracy against confidence, evaluated on the NuScenes validation set. Dashed line represents a perfectly calibrated model.

found by minimising the expression

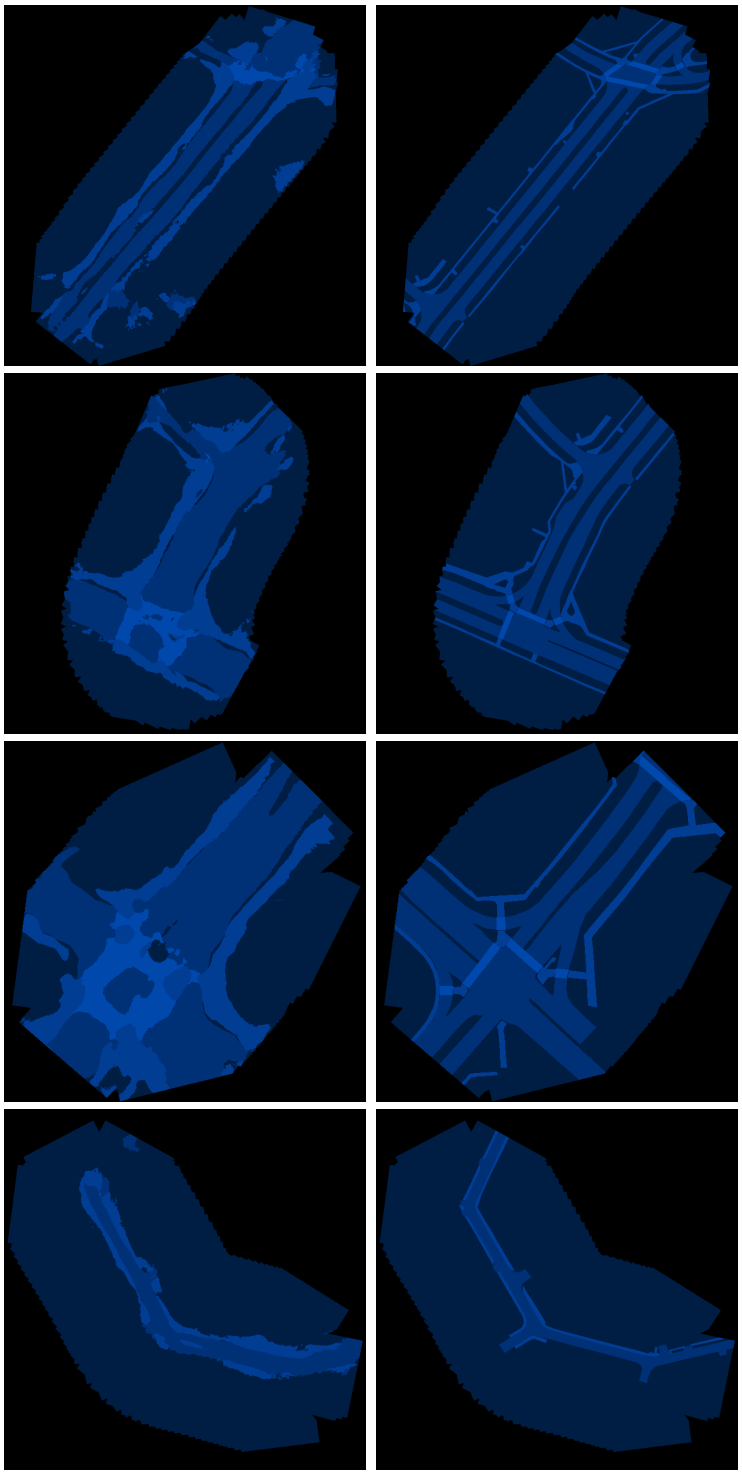
$$\min_{f} \sum_{n} (f_n - a_n)^2 \quad \text{subject to } f_n \leq f_{n+1} \tag{4.21}$$

where a_n is the accuracy of histogram bin n as described above. The calibrated probability for a prediction $\Phi^c_{pq}(z_k)$ was then given by returning the value f_n corresponding to the histogram bin n into which $\Phi^c_{pq}(z_k)$ falls.

Figure 4.8b shows a reliability diagram for the same PyrOccNet network as before, after calibration using the isotonic regression approach. After calibration, each reliability curve adhered much more closely to the diagonal, indicating that the returned network confidences were a more reliable estimate of the true posterior $P\left(m^c_{pq} = 1 | z_k\right)$ that an object of class c existed at location pq.

4.6.3 Multi-frame fusion

Given the calibrated inverse sensor model from Section 4.6.2, it was then possible to apply the Bayesian filtering framework from Section 4.6.1 to generate large-scale occupancy grid maps by fusing the predictions from multiple images over time. Each ‘scene’ of the NuScenes dataset consisted of a 20s long sequence of video, captured from six surround view cameras. Figure 4.9 shows several examples of maps produced by applying the Bayesian filtering



Multi-frame Ground truth

Figure 4.9. Examples of large-scale maps from the NuScenes validation set. The left column shows maps generated by applying the Bayesian filtering algorithm described in Section 4.2, using a calibrated PyrOccNet network as the inverse sensor model. The right column shows the corresponding ground truth map. Only the four static NuScenes classes (*drivable area*, *walkway*, *pedestrian crossing* and *parking space*) are shown. See Figure 4.10 for a comparison to single-frame results.

Table 4.5. Intersection over union scores for single-frame and multi-frame fusion methods on the NuScenes validation set. *Calibrated* indicates that the inverse sensor model was calibrated using isotonic regression before applying the Bayesian filtering algorithm.

Method	Drivable	Ped. crossing	Walkway	Carpark	MEAN
Single frame	46.7	22.9	22.4	11.7	25.9
Multi-frame (uncalibrated)	47.5	19.9	23.7	12.2	25.8
Multi-frame (calibrated)	50.3	20.3	25.3	12.4	27.1

approach of Section 4.6.1 to each of these sequences. Only the four static NuScenes classes (*drivable area*, *walkway*, *pedestrian crossing* and *parking space*) were considered for this analysis, since the occupancy state of dynamic objects such as cars changes over time. Since we were primarily interested in the map-building capabilities of the method rather than localisation, ground-truth pose information was used to obtain the mapping from camera coordinates to world coordinates at each timestep. The resulting maps, as shown in Figure 4.9, provided useful representations of large-scale road scenes, capturing relevant and accurate information about the road topology, locations of pedestrian crossings etc., making them suitable inputs to further downstream tasks.

In addition to enabling the creation of large-scale maps, the Bayesian filtering algorithm also provided quantitative advantages over generating local maps for each frame individually. To assess the benefits of this approach, some of the evaluation from Section 4.5 was repeated for the multiple frame, and the predicted local occupancy grid maps were compared against local maps extracted from scene-level occupancy grids constructed using Bayesian filtering. Unlike in previous experiments, parts of the scene which were occluded by foreground objects were not excluded from the computation of accuracy.

Intersection over union scores for single-frame predictions, as well as predictions produced by accumulating information across multiple frames (using both a calibrated and uncalibrated inverse sensor model), are shown in Table 4.5. Figure 4.10 also provides a qualitative comparison between the two methods. The results in Table 4.5 confirmed that applying the Bayesian filtering method with a calibrated inverse sensor model did improve the accuracy of predictions compared to those obtained from a single image, particularly over the *drivable area* and *walkway* classes, which achieved considerable improvement over the single-image baseline. This was however slightly offset by lower performance on the *pedestrian crossing* class.

The results in Table 4.5 also highlighted the importance of the calibration procedure described in Section 4.6.2. Applying the Bayesian filtering algorithm naïvely to the uncalibrated PyrOccNet model actually resulted in slightly worse average IoU score than the single-frame case. Using a correctly calibrated model on the other hand, resulted in a moder-

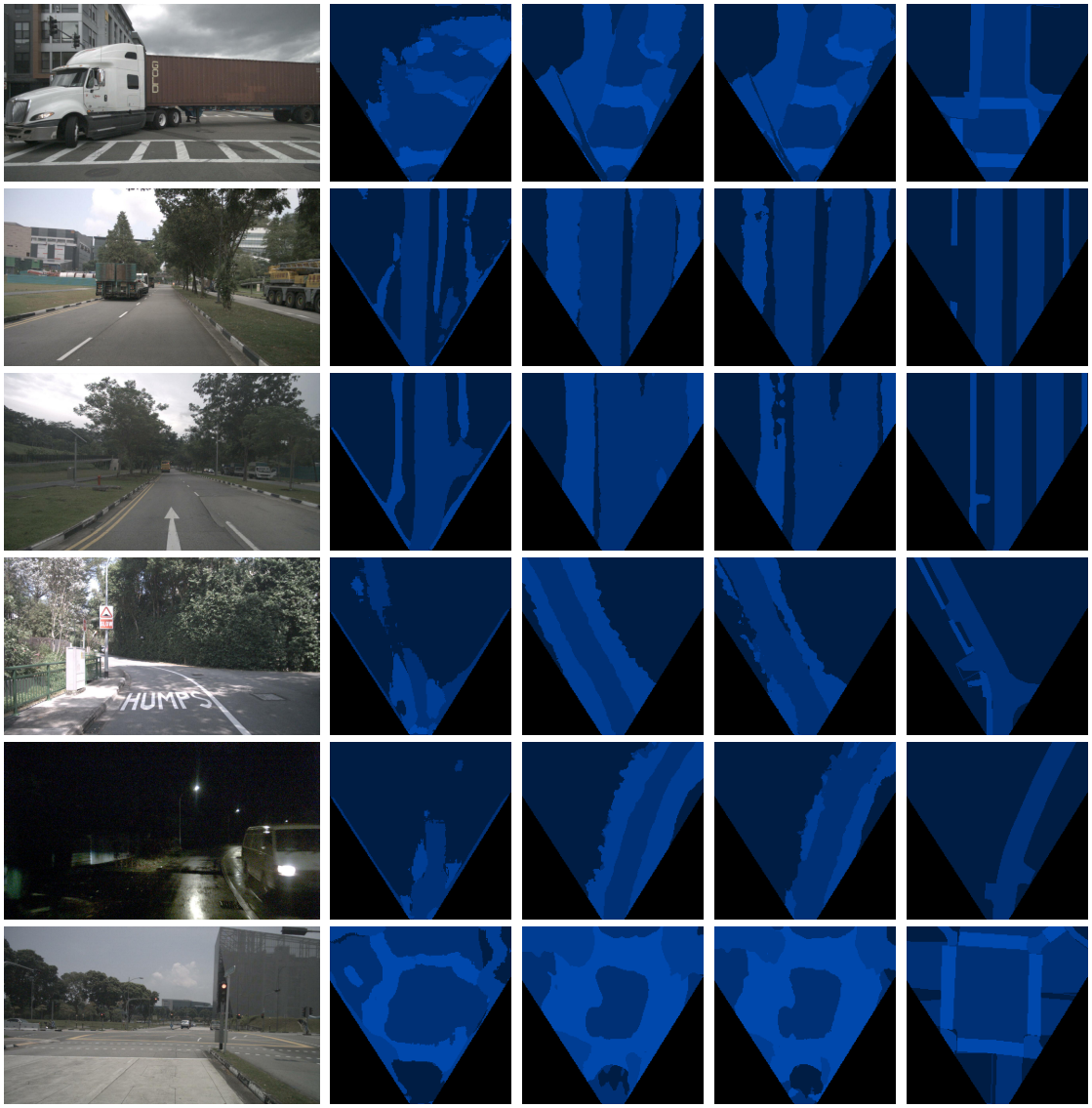


Figure 4.10. Qualitative comparison of single-frame vs multi-frame fusion methods on the NuScenes validation set. Multi-frame results are warped back into the single-frame field of view for comparison.

ate improvement in performance. This implied that correctly calibrated probabilities were essential to the success of the Bayesian filtering approach.

The advantage of the multi-frame approach is particularly apparent from the qualitative examples in Figure 4.10. In row 1 of the figure, the road scene in front of the camera was temporarily occluded by a large truck. In the single image case, the PyrOccNet predictions broke down in the unseen part of the image. In the multi-frame scenario however, information from previous and subsequent frames could be incorporated to help resolve the ambiguity. Similar situations arose in rows 4 and 5 where distant parts of the scene were incorrectly classified by the single frame due to occlusion as the car rounded the bend in row 4 or due to the low light conditions in row 5. In rows 2 and 3, the single image model falsely assumed only a single carriageway, whereas the additional context provided by the Bayesian filtering method correctly identified the dual carriageway. Row 6 of the figure shows an example of a failure case of the Bayesian approach: combining multiple inconsistent predictions sometimes led to narrow structures, such as the pedestrian crossings, being blurred over a large area. This may explain the reduced performance on this class reported in Table 4.5. It can be seen however by comparing the calibrated and uncalibrated cases in rows 4 and 5 of Figure 4.10 that the use of a calibrated inverse sensor model helped alleviate this problem to some extent.

4.6.4 Sensor fusion

As discussed above, one of the limitations of the Bayesian filtering approach is that it does not provide a natural mechanism for fusing observations of dynamic objects over time. A special case, however, was where multiple observations were captured simultaneously. The NuScenes data collection platform featured a set of six cameras arranged to provide full 360°coverage around the vehicle, and a top-mounted LiDAR rotating at a speed of 20Hz. The cameras were synchronised such that each shutter was triggered as the rotating LiDAR beam swept over them. This meant that the six cameras were synchronised to within 50ms of one another: a small enough time interval that the occupancy state of the world would not change substantially between observations. It was therefore possible to apply the Bayesian filtering framework to create composite local maps of the ego vehicle's surroundings for a given moment in time. Examples of such maps, which captured both the static scene layout and dynamic objects, are illustrated in Figure 4.11. The white lines in the figure indicate the boundaries of each camera's field of view. As can be seen from Figure 4.11, the overlap between adjacent cameras was fairly small, which made any quantitative improvements in accuracy from merging multiple observations minimal. However, this fused representation may be highly useful for future downstream tasks. For example, if it were required to extract

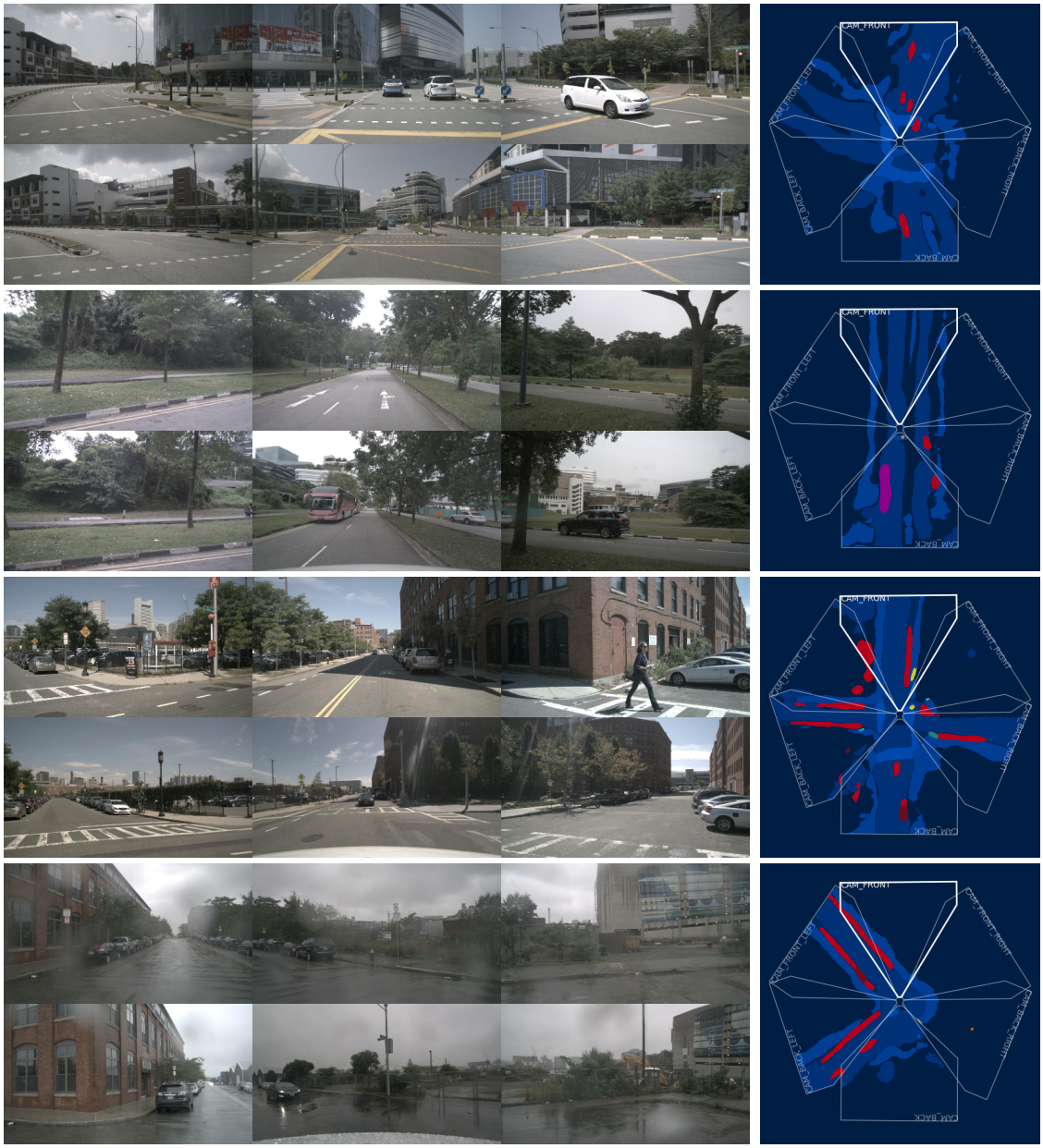


Figure 4.11. Composite local maps (right) generated by applying Bayesian filtering to images from six surround-view cameras (left).

individual instances from the occupancy grids, using a fused representation would avoid counting objects which span multiple images twice.

4.7 Conclusions

The aim of this chapter was to solve the problem of estimating dense birds-eye view maps of a traffic scene using monocular cameras alone. We began by defining the problem in terms of a powerful and flexible framework called a semantic Bayesian occupancy grid. This framework provided a representation which provided a comprehensive summary of the scene at a given moment in time: capturing both moving objects such as cars and pedestrians, as well as static regions such as the road area and pedestrian crossings. Furthermore, the framework provided a powerful mechanism for fusing predictions from multiple sensors and accumulating information over time.

Within this framework, a novel neural network called the Pyramid Occupancy Network was introduced. The PyrOccNet took a monocular image as input, transformed the image to a birds-eye view representation, and processed it to produce the final occupancy grid map. The architecture was inspired by its predecessor from the previous chapter, but incorporated several new innovations which enabled it to function more efficiently. In particular, a new Dense Transformer Layer provided the means to transform features to the birds-eye view without the need to explicitly build a memory-intensive 3D voxel grid. This was combined with a feature pyramid structure to extract more expressive image-based features.

The proposed algorithm was evaluated in two distinct settings. For the first task, we considered the problem of using a single monocular image to generate a local map of the scene visible from a single camera. The evaluation was based on two large-scale autonomous driving datasets: NuScenes and Argoverse, which were adapted to the map prediction task. An ablation study confirmed the effectiveness of the Dense Transformer Layer, and further experiments against both state-of-the-art published methods and hand-crafted baselines showed that the PyrOccNet was able to capture the scene in greater detail than alternative approaches. An examination of the generalisability of the various methods found that a PyrOccNet trained on one dataset could achieve moderate success on another, although this remained a challenge for all approaches discussed. This finding suggested that in spite of the availability of large-scale datasets such as NuScenes, more work must be done to obtain reliable generalisation across methods.

In the second scenario, the Bayes filtering algorithm was applied to the single frame predictions from the first scenario to generate larger-scale maps of entire streets or neighbourhoods. We were able to demonstrate that accumulating predictions over multiple frames

produced more accurate maps than the single frames alone. This achievement however relied on careful calibration of the output probabilities to overcome the neural network's tendency towards overconfident predictions. Work in this section addressed the problem of generating large-scale maps of the static world, but left open the question of incorporating dynamic motion into the occupancy grid framework.

4.7.1 Limitations

The work presented in this chapter represented an initial step towards the goal of building high-definition semantic maps of scenes from monocular video. Whilst the results were promising, the accuracy of the method currently (for example the results in Table 4.2 and Table 4.3) were not yet sufficient to enable full self-driving from the map representation alone. In particular smaller objects such as pedestrians, which only occupy a few pixels in the occupancy grid, were still challenging to reliably detect. This may in part be explained by the challenges of operating purely from monocular images, which, as established in Chapter 3, cannot yet match the performance of methods relying on LiDAR and other sensors. However there is a representation issue as well: dividing the world into discrete grid cells necessarily places a limitation on the minimum size of objects that can be detected. It is likely that for these types of objects, representing them within a rasterised semantic map may always remain a challenge and that other methods, such as more traditional object detection approach, may be more appropriate.

Another significant significant limitation of the proposed method was the inability to represent dynamic objects using the multi-frame fusion strategy discussed in Section 4.6. This capability would be a valuable asset in an online setting, where it is necessary to track and predict the motion of objects such as cars in real time. Extensions to the occupancy grid framework which can handle dynamic objects do exists, and are referred to as DOGMAs (dynamic occupancy grid maps) (Nuss, Reuter, et al., 2018). However such methods rely on accurate estimates of the dynamic motion of every grid cell, and determining this information from monocular video sequences represents a challenging research problem in itself. A simpler solution would be to use a 3D object detection algorithm such as the one proposed in Chapter 3 to detect moving objects and to track them at the instance level. This may represent a more natural way to express motion in road scenes than the grid-based approach described in this thesis.

4.7.2 Current context

Since the time of the work conducted in this chapter, interest in the problem of predicting semantic birds-eye view maps from monocular images has grown (Mani et al., 2020; Reiher et al., 2020; Can et al., 2020; Philion and Fidler, 2020). Encouragingly, a common trend among more recent works is a move away from purely learned transformations to the birds-eye view (such as those used by the VED (Lu, Molengraft, et al., 2019) and VPN (Pan, Sun, et al., 2020) networks discussed in this chapter) and instead sharing our approach in placing greater emphasis on the underlying projective camera geometry. For example Reiher et al. (2020) and Can et al. (2020) used an inverse perspective mappings (Hartley and Zisserman, 2003) to transform features to the birds-eye view, and then trained networks to correct the errors introduced by the assumption of a flat plane. Philion and Fidler (2020) meanwhile used an approach similar to the OFT layer described in Chapter 3 to lift pixels to 2D points, before accumulating them over vertical pillars to obtain a birds-eye view. Among these works there is increased focus on the multi-frame setting discussed in Section 4.6, with Can et al. (2020) having used a symmetric max-pooling operation to combine predictions across multiple frames. In addition to predicting birds-eye view maps, Philion and Fidler (2020) also learned a differentiable cost map which was directly used for path planning and future trajectory prediction. The success of their work offers further compelling evidence of the advantages of the birds-eye view representation when only monocular video is available.

Chapter 5

Conclusions

5.1 Summary

This thesis has addressed two fundamental tasks which are central to the future success of autonomous driving technology: 3D object detection and semantic map prediction. These tasks were constrained to the case where only monocular camera imagery is available: an important case study for lowering the cost of future autonomous vehicles, extending their perception range and improving sensor redundancy. The central philosophy of this thesis was that such camera images, whilst a rich medium, are a challenging representation to work with. To the greatest extent possible, therefore, all reasoning about the 3D structure of the world should take place in a separate, metric representation space called the birds-eye view. To this end, we introduced two neural network components: the orthographic feature transform and the dense transformer layer; which were capable of manipulating image-based feature representations into the birds-eye view using a combination of deep learning and geometry. These were integrated into two end-to-end deep neural network architectures: the OFTNet and the Pyramid Occupancy Network.

For the problem of 3D object detection, we showed that applying the OFTNet network resulted in state-of-the-art monocular performance on the KITTI object detection benchmark: the largest autonomous driving dataset at the time. Extensive analysis of the types of errors encountered by the OFTNet and other leading methods was conducted, and found that the OFTNet shared many features in common with other approaches which had direct access to depth information via stereo or LiDAR.

The second part of this thesis proceeded to formulate the problem of semantic map prediction: a task which had previously been difficult to study due to lack of appropriate data. This limitation was overcome by leveraging two recent two recent datasets: Argoverse and NuScenes; to generate camera-centric birds-eye view map labels which combined

both static road geometry information and dynamic agent bounding boxes. The Pyramid Occupancy Network was then used to predict these local maps directly from monocular images, outperforming both existing works and hand-crafted baselines. A framework called Bayesian occupancy grid mapping, which was extended to the case of semantic mapping, was then used to accumulate the predictions from multiple cameras and timesteps. We was able to show that this approach could be used to generate maps of entire streets or regions, and achieved greater accuracy than simply predicting each frame independently.

5.2 Future Work

Naturally, over the course of this PhD there were numerous ideas and extensions which have not been discussed which remain open as future research opportunities. A few of these ideas are summarised below:

Explicit knowledge of depth This thesis introduced two neural network modules for performing transformations between an image-based and birds-eye view perspective: the orthographic feature transform and the dense transformer layer. In both cases, the modules relied on the network's ability to implicitly reason about the depth of points in the scene. However as established in the previous section, a common theme among more recent works (Xu and Chen (2018), Wang, Chao, et al. (2019), and You et al. (2019)) has been a more explicit representation of depth, where the network is trained to directly output a geometric representation of the world before passing this on for further processing. Combining ideas of both explicit depth estimation and end-to-end learning has the potential for further impacts on object detection and map prediction performance.

Alternative output representations The notion of transforming monocular images into a birds-eye view representation is a powerful and flexible one, and many other problems may benefit outside of the two that have been considered in this thesis. Examples of other potential tasks which lend themselves well to a similar framework include digital elevation map estimation (Malartre et al., 2010), lane detection (Tang et al., 2021), and cost map estimation for online path planning (Ferguson and Likhachev, 2008). Furthermore, the same environmental structure which makes birds-eye view representations applicable for autonomous driving is also present in many other applications: indoor domestic robotics, agricultural robotics and some augmented reality applications to name just a few.

Improved models of uncertainty Section 4.6 illustrated the importance of neural networks which produce correctly calibrated probabilities to the Bayesian filtering algorithm. More generally within scene understanding task, predicting confidences which better reflect a system's true probability of error is important for decision making; for example by identifying objects that the system hasn't seen before. The work discussed in Chapter 4 adopted the relatively crude approach of isotonic regression to obtain calibrated probabilities. However other ways of dealing with uncertainty have been proposed, such as Bayesian neural networks (Gal, 2016; Kendall and Gal, 2017), which may provide a more elegant solution to the problem of model uncertainty and could be explored further.

Path planning and prediction The results in this thesis have demonstrated the value of the birds-eye view representation as applied to 3D structure problems such as 3D object detection and map prediction. However perhaps the most valuable application of the birds-eye view is to tasks involving motion such as trajectory planning and future prediction. Performing such tasks in the image space is hugely challenging: small ego-vehicle movements can have a profound effect on appearance, and the motion of other objects varies wildly depending on the distance from and direction relative to the camera. Since the birds-eye view is metric, motion in this space reflects the true velocities of objects in 3D. As a result of this property, most existing works which tackle these types of problems (see Bansal et al., 2018; Lee, Choi, et al., 2017; Salzmann et al., 2020; Rhinehart et al., 2019 for examples) operate in this space. However, no existing works which adopt this strategy operate exclusively from monocular images, presenting considerable scope for further research.

References

Adelson, Edward H et al. (1984). “Pyramid methods in image processing”. In: *RCA engineer* 29.6, pp. 33–41.

Alatise, Mary B and Gerhard P Hancke (2020). “A review on challenges of autonomous mobile robot and sensor fusion methods”. In: *IEEE Access* 8, pp. 39830–39846.

Alshehhi, Rasha et al. (2017). “Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 130, pp. 139–149.

Anderson, JM et al. (2014). “The Promise and Perils of Autonomous Vehicle Technology”. In: *Autonomous Vehicle Technology: A Guide for Policymakers*, pp. 12–16.

Andrews, Stuart, Ioannis Tsochantaridis, and Thomas Hofmann (2003). “Support vector machines for multiple-instance learning”. In: *Advances in neural information processing systems*, pp. 577–584.

Ayache, Nicholas and Olivier Faugeras (1989). “Maintaining representations of the environment of a mobile robot”. In:

Badino, Hernán, Uwe Franke, and Rudolf Mester (2007). “Free space computation using stochastic occupancy grids and dynamic programming”. In: *Workshop on Dynamical Vision, ICCV, Rio de Janeiro, Brazil*. Vol. 20. Citeseer.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.12, pp. 2481–2495.

Baker, Simon et al. (2011). “A database and evaluation methodology for optical flow”. In: *International journal of computer vision* 92.1, pp. 1–31.

Bansal, Mayank, Alex Krizhevsky, and Abhijit Ogale (2018). “Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst”. In: *arXiv preprint arXiv:1812.03079*.

Bao, Sid Yingze and Silvio Savarese (2011). “Semantic structure from motion”. In: *CVPR 2011*. IEEE, pp. 2025–2032.

Barth, Matthew, Kanok Boriboonsomsin, and Guoyuan Wu (2014). "Vehicle automation and its potential impacts on energy and emissions". In: *Road vehicle automation*. Springer, pp. 103–112.

Beltrán, Jorge et al. (2018). "Birdnet: a 3d object detection framework from lidar information". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 3517–3523.

Bosse, Michael and Robert Zlot (2008). "Map matching and data association for large-scale two-dimensional laser scan-based slam". In: *The International Journal of Robotics Research* 27.6, pp. 667–691.

Bradski, G. (2000). "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools*.

Butler, D. J. et al. (Oct. 2012). "A naturalistic open source movie for optical flow evaluation". In: *European Conf. on Computer Vision (ECCV)*. Ed. by A. Fitzgibbon et al. (Eds.) Part IV, LNCS 7577, pp. 611–625.

Caesar, Holger, Varun Bankiti, et al. (2019). "nuScenes: A multimodal dataset for autonomous driving". In: *arXiv preprint arXiv:1903.11027*.

Caesar, Holger, Jasper Uijlings, and Vittorio Ferrari (2018). "Coco-stuff: Thing and stuff classes in context". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1209–1218.

Can, Yigit Baran et al. (2020). "Understanding Bird's-Eye View Semantic HD-Maps Using an Onboard Monocular Camera". In: *arXiv preprint arXiv:2012.03040*.

Cao, Zhe et al. (2017). "Realtime multi-person 2d pose estimation using part affinity fields". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7291–7299.

Casas, Sergio, Wenjie Luo, and Raquel Urtasun (2018). "Intentnet: Learning to predict intention from raw sensor data". In: *Conference on Robot Learning*, pp. 947–956.

Chabot, Florian et al. (2017). "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2040–2049.

Chandra, Rohan et al. (2020). "Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms". In: *IEEE Robotics and Automation Letters* 5.3, pp. 4882–4890.

Chang, Ming-Fang et al. (2019). "Argoverse: 3d tracking and forecasting with rich maps". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8748–8757.

Chatila, Raja and Jean-Paul Laumond (1985). "Position referencing and consistent world modeling for mobile robots". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, pp. 138–145.

Chen, Liang-Chieh, George Papandreou, et al. (2017). "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587*.

Chen, Xiaozhi, Kaustav Kundu, Ziyu Zhang, et al. (2016). "Monocular 3d object detection for autonomous driving". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2156.

Chen, Xiaozhi, Kaustav Kundu, Yukun Zhu, et al. (2015). "3D Object Proposals for Accurate Object Class Detection". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., pp. 424–432. URL: http://papers.nips.cc/paper/5644-3d-object-proposals-for-accurate-object-class-detection.pdf.

Chen, Xiaozhi, Huimin Ma, et al. (2017). "Multi-view 3d object detection network for autonomous driving". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1907–1915.

Chen, Yushi, Zhouhan Lin, et al. (2014). "Deep learning-based classification of hyperspectral data". In: *IEEE Journal of Selected topics in applied earth observations and remote sensing* 7.6, pp. 2094–2107.

Chen, Zhe, Jing Zhang, and Dacheng Tao (2019). "Progressive lidar adaptation for road detection". In: *IEEE/CAA Journal of Automatica Sinica* 6.3, pp. 693–702.

Chiu, Hsu-kuang et al. (2020). "Probabilistic 3d multi-object tracking for autonomous driving". In: *arXiv preprint arXiv:2001.05673*.

Chong, Z. J. et al. (2013). "Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment". In: *2013 IEEE International Conference on Robotics and Automation*, pp. 1554–1559.

Claypool, Henry, Amitai Bin-Nun, and Jeffery Gerlach (2017). "Self-driving cars: The impact on people with disabilities". In: *Newton, MA: Ruderman Family Foundation*.

Cordts, Marius et al. (2016). "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223.

Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297.

Crayton, Travis J and Benjamin Mason Meier (2017). "Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy". In: *Journal of Transport & Health* 6, pp. 245–252.

Cui, Henggang et al. (2019). "Multimodal trajectory predictions for autonomous driving using deep convolutional networks". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2090–2096.

Dai, Jifeng et al. (2016). "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems*, pp. 379–387.

Dalal, Navneet and Bill Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. CVPR '05. USA: IEEE Computer Society, pp. 886–893. ISBN: 0769523722. DOI: 10.1109/CVPR.2005.177. URL: https://doi.org/10.1109/CVPR.2005.177.

Danescu, Radu, Florin Oniga, and Sergiu Nedevschi (2011). "Modeling and tracking the driving environment with a particle-based occupancy grid". In: *IEEE Transactions on Intelligent Transportation Systems* 12.4, pp. 1331–1342.

Daum, Frederick E. (2015). "Extended Kalman Filters". In: *Encyclopedia of Systems and Control*. Ed. by John Baillieul and Tariq Samad. London: Springer London, pp. 411–413. ISBN: 978-1-4471-5058-9. DOI: 10.1007/978-1-4471-5058-9_62. URL: https://doi.org/10.1007/978-1-4471-5058-9_62.

Davison, Andrew J (2003). "Real-time simultaneous localisation and mapping with a single camera". In: IEEE, p. 1403.

DeGroot, Morris H and Stephen E Fienberg (1983). "The comparison and evaluation of forecasters". In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 32.1-2, pp. 12–22.

Del Moral, Pierre (1997). "Nonlinear filtering: Interacting particle resolution". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 325.6, pp. 653–658.

Delakis, Manolis and Christophe Garcia (2008). "text Detection with Convolutional Neural Networks." In: *VISAPP (2)*, pp. 290–294.

Deng, Liuyuan et al. (2019). "Restricted deformable convolution-based road scene semantic segmentation using surround view cameras". In: *IEEE Transactions on Intelligent Transportation Systems*.

Divvala, Santosh K, Alexei A Efros, and Martial Hebert (2012). "How important are "deformable parts" in the deformable parts model?" In: *European Conference on Computer Vision*. Springer, pp. 31–40.

Dixit, Vinayak V, Sai Chand, and Divya J Nair (2016). "Autonomous vehicles: disengagements, accidents and reaction times". In: *PLoS one* 11.12, e0168054.

Djuric, Nemanja et al. (2018). "Motion prediction of traffic actors for autonomous driving using deep convolutional networks". In: *arXiv preprint arXiv:1808.05819* 2.

Eigen, David, Christian Puhrsch, and Rob Fergus (2014). "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., pp. 2366–2374. URL: https://proceedings.neurips.cc/paper/2014/file/7bccfde7714a1ebadf06c5f4cea752c1-Paper.pdf.

Elfes, Alberto (1989). "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6, pp. 46–57.

Elfes, Alberto et al. (1990). "Occupancy grids: A stochastic spatial representation for active robot perception". In: *Proceedings of the Sixth Conference on Uncertainty in AI*. Vol. 2929, p. 6.

Engel, Jakob, Thomas Schöps, and Daniel Cremers (2014). "LSD-SLAM: Large-scale direct monocular SLAM". In: *European conference on computer vision*. Springer, pp. 834–849.

Engelcke, Martin et al. (2017). "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1355–1361.

Erhan, Dumitru et al. (2014). "Scalable object detection using deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2147–2154.

Everingham, M. et al. (2011). *The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results*. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

Felzenszwalb, Pedro F, Ross B Girshick, and David McAllester (2010). "Cascade object detection with deformable part models". In: *2010 IEEE Computer society conference on computer vision and pattern recognition*. IEEE, pp. 2241–2248.

Felzenszwalb, Pedro F, Ross B Girshick, David McAllester, and Deva Ramanan (2009). "Object detection with discriminatively trained part-based models". In: *IEEE transactions on pattern analysis and machine intelligence* 32.9, pp. 1627–1645.

Ferguson, Dave and Maxim Likhachev (2008). "Efficiently using cost maps for planning complex maneuvers". In: *Lab Papers (GRASP)*, p. 20.

Fischler, Martin A. and Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6, pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

Fischler, Martin A and Robert A Elschlager (1973). "The representation and matching of pictorial structures". In: *IEEE Transactions on computers* 100.1, pp. 67–92.

Freund, Yoav and Robert E Schapire (1995). "A desicion-theoretic generalization of on-line learning and an application to boosting". In: *European conference on computational learning theory*. Springer, pp. 23–37.

Fritsch, Jannik, Tobias Kuehnl, and Andreas Geiger (2013). "A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms". In: *International Conference on Intelligent Transportation Systems (ITSC)*.

Fu, Cheng-Yang, Wei Liu, et al. (2017). "Dssd: Deconvolutional single shot detector". In: *arXiv preprint arXiv:1701.06659*.

Fu, Huan, Mingming Gong, et al. (2018). "Deep ordinal regression network for monocular depth estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2002–2011.

Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, pp. 267–285.

Gal, Yarin (2016). "Uncertainty in Deep Learning". PhD thesis. University of Cambridge.

Garcia, Christophe and Manolis Delakis (2004). "Convolutional face finder: A neural architecture for fast and robust face detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 26.11, pp. 1408–1423.

Gauss, Carl Friedrich (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium auctore Carolo Friderico Gauss*. sumtibus Frid. Perthes et IH Besser.

Geiger, Andreas, Philip Lenz, Christoph Stiller, et al. (2013). "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11, pp. 1231–1237.

Geiger, Andreas, Philip Lenz, and Raquel Urtasun (2012). "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3354–3361.

Gindele, Tobias et al. (2009). "Bayesian occupancy grid filter for dynamic environments using prior map knowledge". In: *2009 IEEE Intelligent Vehicles Symposium*. IEEE, pp. 669–676.

Girshick, Ross (2015). "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Girshick, Ross et al. (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Gkioxari, Georgia, Jitendra Malik, and Justin Johnson (2019). "Mesh r-cnn". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9785–9795.

Glennie, Craig and Derek D Lichti (2010). "Static calibration and analysis of the Velodyne HDL-64E S2 for high accuracy mobile scanning". In: *Remote Sensing* 2.6, pp. 1610–1624.

Godard, Clément, Oisin Mac Aodha, and Gabriel J Brostow (2017). "Unsupervised monocular depth estimation with left-right consistency". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 270–279.

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Gu, Chunhui et al. (2009). "Recognition using regions". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1030–1037.

Guo, Chuan et al. (2017). "On calibration of modern neural networks". In: *arXiv preprint arXiv:1706.04599*.

Gustafsson, Fredrik and Erik Linder-Norén (2018). "Automotive 3D Object Detection Without Target Domain Annotations". MA thesis.

Guttman, Antonin (1984). "R-trees: A dynamic index structure for spatial searching". In: *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pp. 47–57.

Hadsell, Raia et al. (2009). "Learning long-range vision for autonomous off-road driving". In: *Journal of Field Robotics* 26.2, pp. 120–144.

Hane, Christian et al. (2013). "Joint 3D scene reconstruction and class segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 97–104.

Hart, Peter E, Nils J Nilsson, and Bertram Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

Hartley, Richard and Andrew Zisserman (2003). *Multiple view geometry in computer vision*. Cambridge university press.

He, Kaiming, Georgia Gkioxari, et al. (2017). "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969.

He, Kaiming, Xiangyu Zhang, et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

— (2015). "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9, pp. 1904–1916.

Hecker, Simon, Dengxin Dai, and Luc Van Gool (2018). "End-to-end learning of driving models with surround-view cameras and route planners". In: *Proceedings of the european conference on computer vision (eccv)*, pp. 435–453.

Heitz, Geremy and Daphne Koller (2008). "Learning Spatial Context: Using Stuff to Find Things". In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 30–43. ISBN: 978-3-540-88682-2.

Hendy, Noureldin et al. (2020). "FISHING Net: Future Inference of Semantic Heatmaps In Grids". In: *arXiv preprint arXiv:2006.09917*.

Hermans, Alexander, Georgios Floros, and Bastian Leibe (2014). "Dense 3d semantic mapping of indoor scenes from rgb-d images". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2631–2638.

Herrmann, Andreas, Walter Brenner, and Rupert Stadler (2018). *Autonomous driving: how the driverless revolution will change the world*. Emerald Group Publishing.

Hillel, Aharon Bar et al. (2014). "Recent progress in road and lane detection: a survey". In: *Machine vision and applications* 25.3, pp. 727–745.

Hoiem, Derek, Yodsawalai Chodpathumwan, and Qieyun Dai (2012). "Diagnosing error in object detectors". In: *European conference on computer vision*. Springer, pp. 340–353.

Hopcroft, John and Robert Tarjan (1973). "Algorithm 447: efficient algorithms for graph manipulation". In: *Communications of the ACM* 16.6, pp. 372–378.

Huang, Gao, Zhuang Liu, et al. (2017). "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Huang, Lichao, Yi Yang, et al. (2015). "Densebox: Unifying landmark localization with end to end object detection". In: *arXiv preprint arXiv:1509.04874*.

Jaderberg, Max et al. (2015). "Spatial transformer networks". In: *arXiv preprint arXiv:1506.02025*.

Johnsen, Annika, Niklas Strand, and Jan Andersson (2018). *Literature Review on the Acceptance and Road Safety, Ethical, Legal, Social and Economic Implications of Automated Vehicles: Deliverable 2.1 from the EU-H2020-project BRAVE-BRidging the Gabs for the Adoption of Automated VEhicles*. Institut für empirische Soziologie an der Universität Erlangen-Nürnberg.

Kalman, Rudolph Emil (1960). "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D, pp. 35–45.

Kampffmeyer, Michael, Arnt-Borre Salberg, and Robert Jenssen (2016). "Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images

using deep convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1–9.

Kavraki, Lydia E et al. (1996). "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4, pp. 566–580.

Kendall, Alex and Yarin Gal (2017). "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *Advances in Neural Information Processing Systems 30 (NIPS)*.

Kendall, Alex, Matthew Grimes, and Roberto Cipolla (2015). "Posenet: A convolutional network for real-time 6-dof camera relocalization". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946.

Kiefer, Jack, Jacob Wolfowitz, et al. (1952). "Stochastic estimation of the maximum of a regression function". In: *The Annals of Mathematical Statistics* 23.3, pp. 462–466.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114*.

Kirchner, Alexander and Christian Ameling (2000). "Integrated obstacle and road tracking using a laser scanner". In: *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No. 00TH8511)*. IEEE, pp. 675–681.

Kopelias, Pantelis et al. (2020). "Connected and autonomous vehicles – Environmental impacts – A review". In: *Science of The Total Environment* 712, p. 135237. ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2019.135237. URL: http://www.sciencedirect.com/science/article/pii/S0048969719352295.

Koppula, Hema S et al. (2011). "Semantic labeling of 3d point clouds for indoor scenes". In: *Advances in neural information processing systems*, pp. 244–252.

Krähenbühl, Philipp and Vladlen Koltun (2011). "Efficient inference in fully connected crfs with gaussian edge potentials". In: *Advances in neural information processing systems*, pp. 109–117.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Ku, Jason, Melissa Mozifian, et al. (2018). "Joint 3d proposal generation and object detection from view aggregation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1–8.

Ku, Jason, Alex D Pon, and Steven L Waslander (2019). "Monocular 3D Object Detection Leveraging Accurate Proposals and Shape Reconstruction". In: *arXiv preprint arXiv:1904.01690*.

Kuipers, Benjamin (2000). "The spatial semantic hierarchy". In: *Artificial intelligence* 119.1-2, pp. 191–233.

Kuipers, Benjamin Jack (1977). "Representing knowledge of large-scale space". In:

Kuipers, Benjamin and Yung-Tai Byun (1991). "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations". In: *Robotics and autonomous systems* 8.1-2, pp. 47–63.

Kundu, Abhijit, Yin Li, and James M Rehg (2018). "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3559–3568.

Ladick, L'ubor et al. (2009). "Associative hierarchical crfs for object class image segmentation". In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, pp. 739–746.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira (2001). "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In:

Lang, Alex H et al. (2019). "Pointpillars: Fast encoders for object detection from point clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12697–12705.

LaValle, Steven M (1998). "Rapidly-exploring random trees: A new tool for path planning". In:

LaValle, Steven M and James J Kuffner Jr (2001). "Randomized kinodynamic planning". In: *The international journal of robotics research* 20.5, pp. 378–400.

Lazebnik, Svetlana, Cordelia Schmid, and Jean Ponce (2006). "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE, pp. 2169–2178.

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.

Lee, Namhoon, Wongun Choi, et al. (2017). "Desire: Distant future prediction in dynamic scenes with interacting agents". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 336–345.

Lee, Seokju, Junsik Kim, et al. (2017). "Vpgnet: Vanishing point guided network for lane and road marking detection and recognition". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1947–1955.

Leonard, John J and Hugh F Durrant-Whyte (1991a). "Mobile robot localization by tracking geometric beacons". In: *IEEE Transactions on robotics and Automation* 7.3, pp. 376–382.

— (1991b). “Simultaneous map building and localization for an autonomous mobile robot.” In: *IROS*. Vol. 3, pp. 1442–1447.

Levin, Anat, Dani Lischinski, and Yair Weiss (2004). “Colorization using optimization”. In: *ACM SIGGRAPH 2004 Papers*, pp. 689–694.

Li, Bo (2017). “3d fully convolutional network for vehicle detection in point cloud”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1513–1518.

Li, Bo, Tianlei Zhang, and Tian Xia (2016). “Vehicle detection from 3d lidar using fully convolutional network”. In: *arXiv preprint arXiv:1608.07916*.

Li, Peiliang, Xiaozhi Chen, and Shaojie Shen (2019). “Stereo r-cnn based 3d object detection for autonomous driving”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7644–7652.

Liang, Justin et al. (2019). “Convolutional recurrent network for road boundary extraction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9512–9521.

Lin, Tsung-Yi, Piotr Dollár, et al. (2017). “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125.

Lin, Tsung-Yi, Priya Goyal, et al. (2017). “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988.

Lin, Tsung-Yi, Michael Maire, et al. (2014). “Microsoft COCO: Common objects in context”. In: *European conference on computer vision*. Springer, pp. 740–755.

Liu, Wei et al. (2016). “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer, pp. 21–37.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Lowe, David G (2004). “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2, pp. 91–110.

Lu, Chenyang, Marinus Jacobus Gerardus van de Molengraft, and Gijs Dubbelman (2019). “Monocular semantic occupancy grid mapping with convolutional variational encoder–decoder networks”. In: *IEEE Robotics and Automation Letters* 4.2, pp. 445–452.

Lu, Feng and Evangelos Milios (1997). “Globally consistent range scan alignment for environment mapping”. In: *Autonomous robots* 4.4, pp. 333–349.

Lundquist, Christian, Thomas B Schön, and Umut Orguner (2009). *Estimation of the free space in front of a moving vehicle*.

Luo, Wenjie, Bin Yang, and Raquel Urtasun (2018). "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 3569–3577.

Ma, Lei et al. (2019). "Deep learning in remote sensing applications: A meta-analysis and review". In: *ISPRS journal of photogrammetry and remote sensing* 152, pp. 166–177.

Maggiori, Emmanuel et al. (2016). "Convolutional neural networks for large-scale remote-sensing image classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 55.2, pp. 645–657.

Malartre, F. et al. (2010). "Real-time dense digital elevation map estimation using laserscanner and camera SLAM process". In: *2010 11th International Conference on Control Automation Robotics Vision*, pp. 1212–1218. DOI: 10.1109/ICARCV.2010.5707900.

Mallat, Stephane G (1989). "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE transactions on pattern analysis and machine intelligence* 11.7, pp. 674–693.

Mani, Kaustubh et al. (2020). "MonoLayout: Amodal scene layout from a single image". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1689–1697.

Martin, Martin C and Hans P Moravec (1996). *Robot Evidence Grids.* Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst.

Masehian, Ellips (Nov. 2015). "The Role of Motion Planning in Robotics". In: *Applied Mechanics and Materials* 811, pp. 311–317. DOI: 10.4028/www.scientific.net/AMM.811.311.

Masters, Dominic and Carlo Luschi (2018). "Revisiting small batch training for deep neural networks". In: *arXiv preprint arXiv:1804.07612*.

Matan, Ofer et al. (1992). "Reading handwritten digits: A zip code recognition system". In: *Computer* 25.7, pp. 59–63.

Máttyus, Gellért, Wenjie Luo, and Raquel Urtasun (2017). "Deeproadmapper: Extracting road topology from aerial images". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3438–3446.

Máttyus, Gellért, Shenlong Wang, et al. (2016). "Hd maps: Fine-grained road segmentation by parsing ground and aerial images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3611–3619.

Maturana, Daniel and Sebastian Scherer (2015). "Voxnet: A 3d convolutional neural network for real-time object recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 922–928.

McFarland, Matt (2019). *Most self-driving companies say this tech is crucial. Elon Musk disagrees.* URL: www.edition.cnn.com/2019/06/17/tech/lidar-self-driving-tesla/index.html (visited on 02/05/2021).

Miller, Shelie A and Brent R Heard (2016). *The environmental impact of autonomous vehicles depends on adoption patterns*.

Minemura, Kazuki et al. (2018). "LMNet: Real-time multiclass object detection on CPU using 3D LiDAR". In: *2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE, pp. 28–34.

Montemerlo, Michael (July 2003). "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association". PhD thesis. Pittsburgh, PA: Carnegie Mellon University.

Moravec, Hans P (1980). "Obstacle avoidance and navigation in the real world by a seeing robot rover." Stanford Univ CA Dept of Computer Science.

— (1979). "Visual mapping by a robot rover". In:

— (1989). "Sensor fusion in certainty grids for mobile robots". In: *Sensor devices and systems for robotics*. Springer, pp. 253–276.

Moravec, Hans and Alberto Elfes (1985). "High resolution maps from wide angle sonar". In: *Proceedings. 1985 IEEE international conference on robotics and automation*. Vol. 2. IEEE, pp. 116–121.

Mousavian, Arsalan et al. (2017). "3d bounding box estimation using deep learning and geometry". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7074–7082.

Moutarlier, Philippe and Raja Chatila (1990). "An experimental system for incremental environment modelling by an autonomous mobile robot". In: *Experimental Robotics I*. Springer, pp. 327–346.

Mozos, Oscar Martinez et al. (2007). "Supervised semantic labeling of places using information extracted from sensor data". In: *Robotics and Autonomous Systems* 55.5, pp. 391–402.

Mur-Artal, Raul, Jose Maria Martinez Montiel, and Juan D Tardos (2015). "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5, pp. 1147–1163.

Mur-Artal, Raúl and Juan D Tardós (2015). "Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM." In: *Robotics: Science and Systems*. Vol. 2015. Rome.

National Highway Traffic Safety Administration, US Department of Transportation (2015). *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. Tech. rep.

— (2008). *National Motor Vehicle Crash Causation Survey, Report to Congress*. Tech. rep.

Neuhold, Gerhard et al. (2017). "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes". In: *International Conference on Computer Vision (ICCV)*. URL: https://www.mapillary.com/dataset/vistas.

Nguyen, Nhat-Duy et al. (2020). "An Evaluation of Deep Learning Methods for Small Object Detection". In: *Journal of Electrical and Computer Engineering* 2020.

Niculescu-Mizil, Alexandru and Rich Caruana (2005). "Predicting good probabilities with supervised learning". In: *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632.

Nilsson, Nils J (1984). *Shakey the robot*. Tech. rep. SRI INTERNATIONAL MENLO PARK CA.

Novak, Libor (2017). "Vehicle Detection and Pose Estimation for Autonomous Driving". MA thesis. Czech Technical University in Prague.

Nowlan, Steven J and John C Platt (1995). "A convolutional neural network hand tracker". In: *Advances in neural information processing systems*, pp. 901–908.

Nuss, Dominik, Stephan Reuter, et al. (2018). "A random finite set approach for dynamic occupancy grid maps with real-time application". In: *The International Journal of Robotics Research* 37.8, pp. 841–866.

Nuss, Dominik, Markus Thom, et al. (2014). "Fusion of laser and monocular camera data in object grid maps for vehicle environment perception". In: *17th International Conference on Information Fusion (FUSION)*. IEEE, pp. 1–8.

Office for National Statistics (2020). *Leading causes of death, UK, 2001-2018*. Tech. rep.

Ohnemus, Michael and Anthony Perl (Dec. 2016). "Shared Autonomous Vehicles: Catalyst of New Mobility for the Last Mile?" In: *Built Environment* 42, pp. 589–602. DOI: 10.2148/benv.42.4.589.

OpenStreetMap contributors (2017). *Planet dump retrieved from https://planet.osm.org*. https://www.openstreetmap.org.

Osadchy, Margarita, Yann Le Cun, and Matthew L Miller (2007). "Synergistic face detection and pose estimation with energy-based models". In: *Journal of Machine Learning Research* 8.May, pp. 1197–1215.

Pan, Bowen, Jiankai Sun, et al. (2020). "Cross-view semantic segmentation for sensing surroundings". In: *IEEE Robotics and Automation Letters* 5.3, pp. 4867–4873.

Pan, Xingang, Jianping Shi, et al. (2017). "Spatial as deep: Spatial cnn for traffic scene understanding". In: *arXiv preprint arXiv:1712.06080*.

Papageorgiou, Constantine P, Michael Oren, and Tomaso Poggio (1998). "A general framework for object detection". In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, pp. 555–562.

Patra, Suvam et al. (2018). "A joint 3d-2d based method for free space detection on roads". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 643–652.

Payen de La Garanderie, Greire, Amir Atapour Abarghouei, and Toby P Breckon (2018). "Eliminating the blind spot: Adapting 3d object detection and monocular depth estimation to 360 panoramic imagery". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 789–807.

Pham, Trung T et al. (2015). "Hierarchical higher-order regression forest fields: An application to 3d indoor scene labelling". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2246–2254.

Philion, Jonah and Sanja Fidler (2020). "Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D". In: *arXiv preprint arXiv:2008.05711*.

Qi, Charles R, Wei Liu, et al. (2018). "Frustum pointnets for 3d object detection from rgb-d data". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 918–927.

Qi, Charles R, Hao Su, et al. (2017). "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660.

Qiao, Jianglin, Dongmo Zhang, and Dave De Jonge (2019). "Graph Representation of Road and Traffic for Autonomous Driving". In: *Pacific Rim International Conference on Artificial Intelligence*. Springer, pp. 377–384.

Redmon, Joseph, Santosh Divvala, et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271.

— (2018). "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767*.

Reiher, Lennart, Bastian Lampe, and Lutz Eckstein (2020). "A Sim2Real Deep Learning Approach for the Transformation of Images from Multiple Vehicle-Mounted Cameras to a Semantically Segmented Image in Bird's Eye View". In: *arXiv preprint arXiv:2005.04078*.

Ren, Shaoqing et al. (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99.

Rhinehart, Nicholas et al. (2019). "Precog: Prediction conditioned on goals in visual multi-agent settings". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2821–2830.

Roddick, Thomas and Roberto Cipolla (2020). "Predicting Semantic Map Representations from Images using Pyramid Occupancy Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11138–11147.

Roddick, Thomas, Alex Kendall, and Roberto Cipolla (2019). "Orthographic feature transform for monocular 3d object detection". In: *Proceedings of the British Machine Vision Conference*.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Sahdev, Raghavender (2017). "Free space estimation using occupancy grids and dynamic object detection". In: *arXiv preprint arXiv:1708.04989*.

Salas-Moreno, Renato F et al. (2013). "Slam++: Simultaneous localisation and mapping at the level of objects". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1352–1359.

Salzmann, Tim et al. (2020). "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data". In: *arXiv preprint arXiv:2001.03093*.

Schulter, Samuel et al. (2018). "Learning to look around objects for top-view representations of outdoor scenes". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 787–802.

Scott, Grant J et al. (2017). "Training deep convolutional neural networks for land–cover classification of high-resolution imagery". In: *IEEE Geoscience and Remote Sensing Letters* 14.4, pp. 549–553.

Sengupta, S. et al. (2012). "Automatic dense visual semantic mapping from street-level imagery". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 857–862.

Sermanet, Pierre et al. (2013). "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229*.

Shatkay, Hagit and Leslie Pack Kaelbling (1997). "Learning topological maps with weak local odometric information". In: *IJCAI (2)*, pp. 920–929.

Shen, Zhiqiang et al. (2017). "Dsod: Learning deeply supervised object detectors from scratch". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1919–1927.

Shotton, Jamie et al. (2006). "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation". In: *European conference on computer vision*. Springer, pp. 1–15.

Shrivastava, Abhinav et al. (2016). "Beyond skip connections: Top-down modulation for object detection". In: *arXiv preprint arXiv:1612.06851*.

Silver, David, J Andrew Bagnell, and Anthony Stentz (2010). "Learning from demonstration for autonomous navigation in complex unstructured terrain". In: *The International Journal of Robotics Research* 29.12, pp. 1565–1592.

Simmons, Reid et al. (2000). "Coordination for multi-robot exploration and mapping". In: *Aaai/Iaai*, pp. 852–858.

Simonelli, Andrea et al. (2019). "Disentangling monocular 3d object detection". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1991–1999.

Smith, Adrian (2013). *Sequential Monte Carlo methods in practice*. Springer Science & Business Media.

Smith, Randall C and Peter Cheeseman (1986). "On the representation and estimation of spatial uncertainty". In: *The international journal of Robotics Research* 5.4, pp. 56–68.

Smith, Randall, Matthew Self, and Peter Cheeseman (1986). "Estimating uncertain spatial relationships in robotics". In:

— (1990). "Estimating uncertain spatial relationships in robotics". In: *Autonomous robot vehicles*. Springer, pp. 167–193.

Sun, Min, Byung-soo Kim, et al. (2013). "Relating things and stuff via objectproperty interactions". In: *IEEE transactions on pattern analysis and machine intelligence* 36.7, pp. 1370–1383.

Sun, Pei, Henrik Kretzschmar, et al. (2020). "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2446–2454.

Sünderhauf, N. et al. (2016). "Place categorization and semantic mapping on a mobile robot". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5729–5736.

Szeliski, Richard (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Tang, Jigang, Songbin Li, and Peng Liu (2021). "A review of lane detection methods based on deep learning". In: *Pattern Recognition* 111, p. 107623. ISSN: 0031-3203. DOI: https:

//doi.org/10.1016/j.patcog.2020.107623. URL: http://www.sciencedirect.com/science/article/pii/S003132032030426X.

Tao, Tong et al. (2010). "Cooperative simultaneous localization and mapping for multi-robot: Approach & experimental validation". In: *2010 8th World Congress on Intelligent Control and Automation*. IEEE, pp. 2888–2893.

Teichmann, Marvin et al. (2018). "Multinet: Real-time joint semantic reasoning for autonomous driving". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 1013–1020.

Thrun, Sebastian (2002). "Probabilistic robotics". In: *Communications of the ACM* 45.3, pp. 52–57.

Thrun, Sebastian and Arno Bücken (1996a). "Integrating grid-based and topological maps for mobile robot navigation". In: *Proceedings of the National Conference on Artificial Intelligence*, pp. 944–951.

— (1996b). *Learning Maps for Indoor Mobile Robot Navigation.* Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Thrun, Sebastian and Yufeng Liu (2005). "Multi-robot SLAM with sparse extended information filters". In: *Robotics Research. The Eleventh International Symposium*. Springer, pp. 254–266.

Thrun, Sebastian, Yufeng Liu, et al. (2004). "Simultaneous localization and mapping with sparse extended information filters". In: *The international journal of robotics research* 23.7-8, pp. 693–716.

Tomasi, Carlo and Takeo Kanade (1992). "Shape and motion from image streams under orthography: a factorization method". In: *International journal of computer vision* 9.2, pp. 137–154.

Transport Systems Catapult (2017). *Market Forecast for Connected and Autonomous Vehicles*.

Triggs, Bill et al. (1999). "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms*. Springer, pp. 298–372.

Uijlings, Jasper RR et al. (2013). "Selective search for object recognition". In: *International journal of computer vision* 104.2, pp. 154–171.

UK Department for Transport (2019). *Reported road casualties in Great Britain: 2019 annual report*. Tech. rep. URL: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/922717/reported-road-casualties-annual-report-2019.pdf.

Ulku, Irem and Erdem Akagunduz (2019). "A survey on deep learning-based architectures for semantic segmentation on 2d images". In: *arXiv preprint arXiv:1912.10230*.

Vaillant, Régis, Christophe Monrocq, and Yann Le Cun (1994). "Original approach for the localisation of objects in images". In: *IEE Proceedings-Vision, Image and Signal Processing* 141.4, pp. 245–250.

Valentin, Julien PC et al. (2013). "Mesh based semantic modelling for indoor and outdoor scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2067–2074.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vineet, Vibhav et al. (2015). "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 75–82.

Viola, Paul and Michael Jones (2001). "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. IEEE, pp. I–I.

Wan, Eric A and Rudolph Van Der Merwe (2000). "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. Ieee, pp. 153–158.

Wang, Chieh-Chih, Charles Thorpe, and Sebastian Thrun (2003). "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. Vol. 1. IEEE, pp. 842–849.

Wang, Dominic Zeng and Ingmar Posner (2015). "Voting for voting in online point cloud object detection." In: *Robotics: Science and Systems*. Vol. 1. 3, pp. 10–15607.

Wang, Jun, Yanjun Huang, and Jian Zhao (2020). "Safety of Autonomous Vehicles". In: *Journal of Advanced Transportation*.

Wang, Yan, Wei-Lun Chao, et al. (2019). "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8445–8453.

Wang, Ziyan, Buyu Liu, et al. (2019). "A parametric top-view representation of complex road scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10325–10333.

Williams, Lance (1983). "Pyramidal parametrics". In: *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pp. 1–11.

Wirges, Sascha et al. (2018). "Object detection and classification in occupancy grid maps using deep convolutional networks". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 3530–3535.

Wolf, Denis F and Gaurav S Sukhatme (2005). "Mobile robot simultaneous localization and mapping in dynamic environments". In: *Autonomous Robots* 19.1, pp. 53–65.

World Health Organization (2018). *Global status report on road safety 2018: Summary*. Tech. rep. World Health Organization.

Wu, Zhirong et al. (2015). "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920.

Xiang, Yu et al. (June 2015). "Data-Driven 3D Voxel Patterns for Object Category Recognition". In: DOI: 10.1109/CVPR.2015.7298800.

Xu, Bin and Zhenzhong Chen (2018). "Multi-level fusion based 3d object detection from monocular images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2345–2353.

Yan, Junjie, Zhen Lei, et al. (2014). "The fastest deformable part model for object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2497–2504.

Yan, Yan, Yuxing Mao, and Bo Li (2018). "Second: Sparsely embedded convolutional detection". In: *Sensors* 18.10, p. 3337.

Yang, Bin, Wenjie Luo, and Raquel Urtasun (2018). "Pixor: Real-time 3d object detection from point clouds". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7652–7660.

Yang, Shichao and Sebastian Scherer (2019). "Cubeslam: Monocular 3-d object slam". In: *IEEE Transactions on Robotics* 35.4, pp. 925–938.

Yao, Jian et al. (2015). "Estimating drivable collision-free space from monocular video". In: *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, pp. 420–427.

You, Yurong et al. (2019). "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving". In: *arXiv preprint arXiv:1906.06310*.

Yu, Chao, Zuxin Liu, et al. (2018). "DS-SLAM: A semantic visual SLAM towards dynamic environments". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1168–1174.

Yu, S., T. Westfechtel, et al. (2017). "Vehicle detection and localization on bird's eye view elevation images using convolutional neural network". In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pp. 102–109. DOI: 10.1109/SSRR.2017.8088147.

Yu, Wonpil and Francesco Amigoni (2014). "IEEE Standard for robot map data representation for navigation". In: *IROS2014 (IEEE/RSJ International Conference on Intelligent Robots and Systems) Workshop on "Standardized Knowledge Representation and Ontologies for Robotics and Automation"*, pp. 3–4.

Yurtsever, Ekim et al. (2020). "A survey of autonomous driving: Common practices and emerging technologies". In: *IEEE Access* 8, pp. 58443–58469.

Zadrozny, Bianca and Charles Elkan (2001). "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In: *Icml*. Vol. 1. Citeseer, pp. 609–616.

Zeeshan Zia, Muhammad, Michael Stark, and Konrad Schindler (2014). "Are cars just 3d boxes?-jointly estimating the 3d shape of multiple objects". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3678–3685.

— (2013). "Explicit occlusion modeling for 3d object class representations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3326–3333.

— (2015). "Towards scene understanding with detailed 3d object representations". In: *International Journal of Computer Vision* 112.2, pp. 188–203.

Zhao, Zhong-Qiu et al. (2019). "Object detection with deep learning: A review". In: *IEEE transactions on neural networks and learning systems* 30.11, pp. 3212–3232.

Zhou, Bolei, Hang Zhao, et al. (2017). "Scene Parsing through ADE20K Dataset". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Zhou, Yin and Oncel Tuzel (2018). "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499.

Zhu, Benjin, Zhengkai Jiang, et al. (2019). "Class-balanced grouping and sampling for point cloud 3d object detection". In: *arXiv preprint arXiv:1908.09492*.

Zhu, Jun-Yan, Taesung Park, et al. (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232.

Zitnick, C Lawrence and Piotr Dollár (2014). "Edge boxes: Locating object proposals from edges". In: *European conference on computer vision*. Springer, pp. 391–405.