Multi-Object Detection, Pose Estimation and Tracking in Panoramic Monocular Imagery for Autonomous Vehicle Perception

GREGOIRE PIERRE HUGUES PAYEN-DE-LA-GARANDERIE

Multi-Object Detection, Pose Estimation and Tracking in Panoramic Monocular Imagery for Autonomous Vehicle Perception

Grégoire Payen de La Garanderie

A thesis presented for the degree of
Doctor of Philosophy at Durham University

Department of Computer Science
Durham University
United Kingdom

8th October 2020

Abstract

While active sensing such as radars, laser-based ranging (LiDAR) and ultrasonic sensors are nearly ubiquitous in modern autonomous vehicle prototypes, cameras are more versatile because they are nonetheless essential for tasks such as road marking detection and road sign reading. Active sensing technologies are widely used because active sensors are, by nature, usually more reliable than cameras to detect objects, however they are lower resolution, break in challenging environmental conditions such as rain and heavy reflections, as well as materials such as black paint. Therefore, in this work, we focus primarily on passive sensing technologies. More specifically, we look at monocular imagery and to what extent, it can be used as replacement for more complex sensing systems such as stereo, multi-view cameras and LiDAR.

Whilst the main strength of LiDAR is its ability to measure distances and naturally enable 3D reasoning; in contrast, camera-based object detection is typically restricted to the 2D image space. We propose a convolutional neural network extending object detection to estimate the 3D pose and velocity of objects from a single monocular camera. Our approach is based on a siamese neural network able to process pair of video frames to integrate temporal information.

While the prior work has focused almost exclusively on the processing of forward-facing rectified rectilinear vehicle mounted cameras, there are no studies of panoramic imagery in the context of

autonomous driving. We introduce an approach to adapt existing convolutional neural networks to unseen 360° panoramic imagery using domain adaptation via style transfer. We also introduce a new synthetic evaluation dataset and benchmark for 3D object detection and depth estimation in automotive panoramic imagery.

Multi-object tracking-by-detection is often split into two parts: a detector and a tracker. In contrast, we investigate the use of end-to-end recurrent convolutional networks to process automotive video sequences to jointly detect and track objects through time. We present a multitask neural network able to track online the 3D pose of objects in panoramic video sequences.

Our work highlights that monocular imagery, in conjunction with the proposed algorithmic approaches, can offer an effective replacement for more expensive active sensors to estimate depth, to estimate and track the 3D pose of objects surrounding the ego-vehicle; thus demonstrating that autonomous driving could be achieved using a limited number of cameras or even a single 360° panoramic camera, akin to a human driver perception.

Declaration

The work in this thesis is based on research carried out within the Innovative Computing Group at the Department of Computer Science at Durham University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all the author's own work unless referenced to the contrary in the text.

Acknowledgements

Foremost, I would like to thanks my supervisor, Prof. Toby Breckon, for his infinite patience, kindness and invaluable guidance throughout my studies. He is, without doubt, the most influential person in my research career, both in his teaching and in his research. I was blessed to be part of such an inspiring and supportive research group.

I must thank my friends and fellow researchers within our research community, the Lovelace Lair, particularly Mr. Mikolaj Kundergorski, without whom, I would never have started this endeavour; Dr. Chas Nelson and Dr. Chris Willcocks who helped me understand the role of a researcher; Dr. Samet Akçay who introduced me to the world of deep learning; Dr. Amir Atapour-Abarghouei, Mr. Philip Jackson, and Mr. Tom Winterbottom for the many conversations and insights in the literature.

I would like to thank all my friends from St. Chad's College who supported me, in particular Dr. Indra Werthmann-Carroll and Mr. Rónán Carroll.

I am grateful to Jaguar Land Rover and the EPSRC for the financial support that they provided me for the elaboration of this thesis; in particular, my main contacts at JLR, Ms. Anna Gaszczak and Mr. Dereck Webster who helped shaping my research topic.

Finally, I must thank my parents and my family for their patience, encouragements and understanding through my PhD studies.

Contents

List of Figures

List of Tables

List of Acronyms

ADAS Advanced Driver Assistance System. 1

AMOTA Average MOTA. 20, 111, 112

AMOTP Average MOTP. 20, 111, 112

AOS Average Orientation Similarity. 103, 110

BN Batch Normalisation. 92

CNN Convolutional Neural Network. 9, 14–17, 21, 23, 24, 32, 51, 62, 68, 70, 84, 85, 87, 88, 101, 111, 113, 115, 120, 121

FoV Field of View. 6, 16, 67, 68, 70, 76, 102

GN Group Normalisation. 91, 92

GPU Graphic Processing Unit. 24, 27, 30, 100

GRU Gated Recurrent Unit. 18

HDR High Dynamic Range. 117, 120

IoU Intersection over Union. 13, 14, 19, 37, 41–43, 45, 46, 79, 103

LiDAR Light Detection and Ranging. 5, 6, 8, 46, 85, 101, 111, 117, 119

LSTM Long Short Term Memory. 18

mAP mean Average Precision. 13, 14, 19, 79, 103, 110

ML mostly lost. 20

MOTA multiple object tracking accuracy. 19, 20, 50, 103, 112

MOTAR recall-normalised MOTA. 20, 112

MOTD multi-object tracking-by-detection. 6, 19, 20, 33, 86, 111

MOTP multiple object tracking precision. 19, 20, 50, 103, 112

MOTS multi-object tracking and segmentation. 19, 122

MRF Markov Random Field. 24

MT mostly tracked. 20

NMS Non Maximum Suppression. 18, 37, 98

ODD operational design domain. 2, 4

OHEM Online Hard Example Mining. 15, 16, 95

ReID Re-Identification. 22, 23, 118

ReLU Rectified Linear Unit. 91

RNN Recurrent Neural Network. 18, 21–23, 121

ROI Region of Interest. 16, 37, 39

RPN Region Proposal Network. 15, 19, 36, 37, 40, 71, 72

SGD Stochastic Gradient Descent. 118

VQA Visual Question Answering. 122

Chapter 1

Introduction

Autonomous vehicles have gained significant coverage in mainstream media in recent years [10, 11]. While fully autonomous vehicles are certainly years away, some autonomous systems have already been integrated into cars in the form of various Advanced Driver Assistance System (ADAS) such as adaptive cruise control, collision alerts and mitigation, automatic high beams [12]. More recently, some commercial cars have featured fully-automated parking assistance, lane keeping assistance, and dynamic driving assistance [12, 13]. The SAE international standard J3016 [14] subsumes the various automation systems under 6 levels of driving automation ranging from *0 – no driving automation* to *5 – full driving automation*. Current production vehicles achieve up to level 2 — the so-called *hands-off* level — while a few manufacturers have plans to release level 3 vehicles — the so-called *eyes-off* level — in the next couple of years [13, 15]. Alas, this kind of partial automation in *level 2* and *level 3* comes with a set challenges regarding safety [16, 17] which have been illustrated by several tragic accidents in both testing vehicles [18] and production vehicles [19–21]. Those challenges of partial automation, revolving around

Figure 1.1: A vehicle equipped with a forward facing stereo camera.

human-computer interactions and driver distraction, stem from the need to keep a human driver involved in the system to take control of the vehicle at short notice in the event of a system failure or limitation [16, 22]. This event is called a disengagement. The annual surveys from the American Automobile Association (AAA) [23–26] produced between 2016 and 2019 show that three quarters of the U.S. drivers are afraid of such automation technology. According to the AAA, the public fear increased in 2018 following the number of high profile accidents where the driver failed to take control of the vehicle during a system failure [25, 26]. In contrast, *level 4* are expected to automatically recover from system failure [14] and would address the current safety concerns related to human-computer interactions [16]. Automation level 1 to 4 are expected to work within a specific operational design domain (ODD). Such domain (e.g. restricted to highway driving) can be defined by the vehicle manufacturer and the vehicle automation is not expected to work outside of this ODD, while automation of *level 5* vehicles are expected to have no ODD limitation and the automation must work at all time. Therefore *level 4* and *level 5* provide a

high-level of automation which does not depend on the driver and would address the current safety concerns [16]. As a result, we see the automation of level 4 and 5 as key requirements for autonomous vehicles.

As early as 1994, the VaMP driverless car [27] was able to drive on the motorway for long distances with little human interaction, however driving in complex urban scenarios remained out of scope. The VaMP car itself followed the work of Dickmanns *et al.* in the 1980s on the VaMoRs lorry [28] and the work of Pomerleau on ALVINN [29], a vehicle guided by a three-layer neural network in 1989. Nowadays, there are several autonomous systems being trialled on public roads by most automotive manufacturers [30]. The DARPA 2007 Urban Challenge [31] featured autonomous vehicles capable of driving in urban traffic. Whilst prototype driverless cars have been involved in very few car crashes; the number of accidents is not a representative metric of how safe a fully autonomous system is. Indeed, Google, a main operator of driverless cars on public roads, reports that their test drivers are expected to take back the control of the car if they themselves discovers any discrepancy between what they see and what the car sees (active disengagement) as well as when an automated alert is reported by the car (passive disengagement) [32]. Hence a better metric is the number of *miles driven per disengagement* (MPD) from automatic mode to manual mode. The report [32] shows that current autonomous vehicles can drive thousands of miles without any disengagements [32]. In 2018, Lv *et al.* [30] published a study of the 2016 annual disengagement reports submitted by manufacturers conducting testing of autonomous vehicles on public roads in California. They set the threshold between *level 2* and *level 3* automation at around 2000 MPD — a target which was achieved by only one manufacturer. For stage 1 automation ($<$ 2000 MPD), active disengagements represent 37% of the overall number of disengagements (active and passive). That is, either the system did not detect the fault, or the driver was not confident that the system would and actively intervened to disengage the vehicle. In contrast, in a passive disengagement, the system successfully detected the fault and was able to alert the driver. The study provides an extensive analysis of the type of

causes for disengagements; in particular, it notes that software failure is the cause of 79.63% of passive disengagement while software limitation is the cause of 87.72% of active disengagement. Software is the primary cause of both passive and active disengagement of autonomous vehicles.

At the core of an autonomous vehicle software stack, there is a situational awareness system which is traditionally divided into three aspects: perception, planning and control [22]. On one hand, the environmental perception aspect gathers data from sensors and produces a coherent and meaningful representation of the real world. On the other hand, the control and planning aspects use this representation to forecast the most likely future events over a short window of time (typically a few seconds), plan the behaviour of the ego-vehicle and consequently control the ego-vehicle. Mapping is also sometime included as a fourth aspect as high definition (HD) maps of the road can be used to alleviate the complexity of the perception system in level 4 vehicles. While mapping can be used as a part of a geo-fenced service such as ridesharing or taxi services, such maps are expensive to create. Besides, they are most likely impossible to maintain on a large scale such as countrywide or worldwide, keeping track of both temporary changes (*e.g.* accidents, road works) and permanent evolutions of the road network. Therefore, it would not be realistic to rely on a HD map in *level 5* vehicles which are expected to have no ODD limitations [14]. Furthermore, Koopman and Fratrik [33] provide a non-exhaustive list of ODD considerations related to objects and events, most of which are not directly related to mapping aspects and would not be addressed by a HD maps, even in a geo-fenced service. Wang and Li [22] showed that the two main causes of disengagements in the 2018 AAA report [25] are undesired behaviours from road users (48%) and computation issues of perception (28%). Those two figures highlight that despite the recent advances, perception remains one of the key challenges of autonomous driving.

1.1 Motivation

Perception is intrinsically linked to the number and kind of sensors used by the vehicle. Indeed, Wang and Li [22] argue that, according to empirical evidence, increasing the number of Light Detection and Ranging (LiDAR) sensors decreases the number of disengagements caused by perception issues. Recently, there has been much research on the use of passive technologies such as stereo cameras (as shown in Figure 1.1) instead of radars, LiDAR and ultrasonic sensors. While active technologies are usually expensive, cameras are much cheaper, smaller and more versatile. Cameras are more versatile because they are essential for tasks such as road marking detection and road sign reading. Active sensing technologies are widely used because active sensors are, by nature, usually more reliable than cameras to detect objects, however they are lower resolution than cameras, break in challenging environmental conditions such as rain and heavy reflections, as well as materials such as black paint — an all too common vehicle body colour. The task of detecting objects using active sensors under significant partial occlusions in complex scenes is no more simpler than its computer vision counterpart [34, 35]. Computer vision is intrinsically a difficult inference problem [36, Section 1.1], however recent advances in computer vision and hardware capabilities enable real-time solutions albeit they still require a lot of processing power. Therefore, in this work, we focus primarily on passive technologies. More specifically, we look at monocular imagery and to what extent, it can be used as replacement for more complex systems such as stereo and multiview setups as well as LiDAR.

LiDAR are primarily used in autonomous driving to provide 3D sensing. A LiDAR provides a dense estimate of the distance (*a.k.a.* depth) of objects to the sensor over a polar coordinate sampling grid. This depth image is subsequently reprojected from polar to cartesian coordinates to form a 3D point cloud or a voxel grid. We study the counterpart problem of depth estimation in monocular colour imagery in Chapter 4. The depth maps produced in Chapter 4 could be substituted in lieu of a LiDAR in any further processing stages which would normally rely on

a LiDAR output. However, the characteristics of the depth map produced by a camera and a LiDAR are not the same. Current camera technologies have a much higher horizontal and vertical resolution than a LiDAR for the same Field of View (FoV), while LiDAR technologies have superior radial (*i.e.* distance) accuracy (see Chapter 4). A camera also provides colour information, not available with a LiDAR, which is useful for many tasks. Therefore, we aim in Chapter 3 to replace a common use of the depth map generated by a LiDAR: object detection and 3D pose estimation. Such 3D detection consists of the size (width, height, length) and 3D pose (translation and rotation) of an object with respect to the egovehicle. In contrast, an object detected by a camera is typically represented by a 2D rectangle in image space. Therefore, the 3D detections provided by a LiDAR are much more useful for self-driving than the 2D detections provided by a camera as 3D detections enable reasoning in the 3D space. In Chapter 3, we show how to estimate the actual 3D position of objects using a monocular camera by leveraging a neural network which is able to learn not only geometrical constraints but also semantic constraints.

While an active sensor might be directional and have a limited FoV like a camera, a typical autonomous vehicle would feature at least one 360° omnidirectional LiDAR. In contrast, the contemporary computer vision work to date has focused almost exclusively on the processing of forward-facing vehicle mounted cameras. In Chapter 4, we introduce an approach to adapt contemporary deep network architectures developed on conventional rectilinear imagery to work on equirectangular 360° panoramic imagery which can be generated by an omni-directional dual fisheye camera or a multi-view camera setup.

In addition to object detection and 3D perception, object tracking is another key aspect of any autonomous driving perception pipeline. Tracking for autonomous driving is accomplished using the multi-object tracking-by-detection (MOTD) approach. In MOTD, the tracking task is divided first into a object detection step in each individual frame followed by grouping of those detections across frames into tracklets. The detection step could be based on any of the detectors

which we develop in Chapters 3, 4, and 5. This approach limits the amount of information available to both the detector and the tracker: the detector does not have access to temporal information and cannot exploit temporal consistency whereas the tracker has access to a limited set of per-detection features instead of a dense representation of the input video sequence. In contrast, in Chapter 3 and 5, we attempt to jointly solve the problem of detection and tracking. Chapter 3 attempts to define a frame-to-frame detection and tracking neural network trained on pairs of images which can link the information of the previous and the next frames together. However this network is not recurrent and therefore is not capable of handling occlusions and more challenging detections. In contrast, Chapter 5 builds upon this work to provide a multi-scale tracking neural network trained on video sequences which maintains an internal recurrent state in order to track objects in more challenging scenarios.

1.2 Thesis Contributions and Structure

The main contributions of this thesis are:

- A novel approach to detect objects and estimate the 3D pose and velocity of those objects in a scene using a single monocular camera. This approach is based on a siamese architecture [37] and the two-stage detector Faster R-CNN [34]. It achieves state-of-the-art results compared to Mousavian *et al.* [4] and Xiang *et al.* [5] for 3D pose estimation and shows that velocity can be more accurately estimated using a siamese network than derived from past and present 3D pose (Chapter 3).

- A method to extend existing neural network architectures to 360° monocular imagery. This extension overcomes the dataset bias problem described in Section 2.2.1 and generalises to 360° imagery by using style transfer [2] to adapt existing automotive datasets to train neural networks suited to 360° imagery. We demonstrate our method by adapting our

3D pose estimation approach described in Chapter 3 and dense depth estimation based on Godard *et al.* [38]. We also introduce the first evaluation dataset for 360° automotive imagery based on synthetic data generated using the Carla simulator [39] (Chapter 4).

- A novel approach for 3D detection and tracking within 360° monocular video sequences based on an end-to-end neural network, extending our work from Chapter 3 and 4. The introduced architecture is a novel end-to-end multi-scale recurrent tracking network based on SSD [40] instead of the conventional two-stages approach of *tracking-by-detection* [41, 42]. We provide one of only two approaches based solely on cameras rather than LiDAR and the only approach using panoramic imagery on the nuScenes tracking benchmark [7] (Chapter 5).

Figures 1.2.A&B show a qualitative example of our object detection and 3D pose estimation approach described in Chapter 3. The images A and B are taken four frames apart and each vehicle is coded using a unique colour that is common between the two frames to illustrate our frame-to-frame tracking results. The cross on the 3D bounding box indicates the front of the vehicle. Figure 1.2.C shows the extension of the 3D pose estimation approach to 360° panoramic imagery in Chapter 4. Figure 1.2.D shows dense depth estimation results using the same domain adaptation approach in Chapter 4, where the colour ranges from light yellow (closest distance to the camera) to deep blue (furthest distance). Figures 1.2.E–G show examples of our multi-class 3D detection and tracking approach within 360° video sequences described in Chapter 5.

These thesis contributions have led to the following publication in the peer-reviewed literature:

- G. Payen de La Garanderie, A. Atapour Abarghouei, and T. P. Breckon, "Eliminating the Blind Spot: Adapting 3D Object Detection and Monocular Depth Estimation to 360° Panoramic Imagery", in Proc. European Conference on Computer Vision, Springer, pp. 812-830, 2018 (Chapter 4 and parts of Chapter 3).

In Chapter 2, we introduce the contemporary state of the art in machine learning and computer vision related to automotive applications which we build upon through this thesis. In Chapter 3, we build a siamese [37] 3D object detection and tracking network based on MS-CNN [43]. We have identified a lack of prior work and datasets on panoramic imagery in automotive application; which we attempt to fill in Chapter 4 by introducing a novel approach to adapt existing training datasets and Convolutional Neural Network (CNN) to panoramic imagery using CycleGAN [2]. We use this methodology to adapt our neural network presented in Chapter 3 as well as Monodepth [38] to 360° panoramic imagery. This approach is based on the concepts of domain adaptation to generalise from the KITTI dataset [8] to our own 360° synthetic imagery testing dataset. We attempt to depart from the tracking literature in Chapter 5 by integrating object detection and tracking together in a single end-to-end CNN. This network is a multi-scale single shot detector inspired by [44, 45].

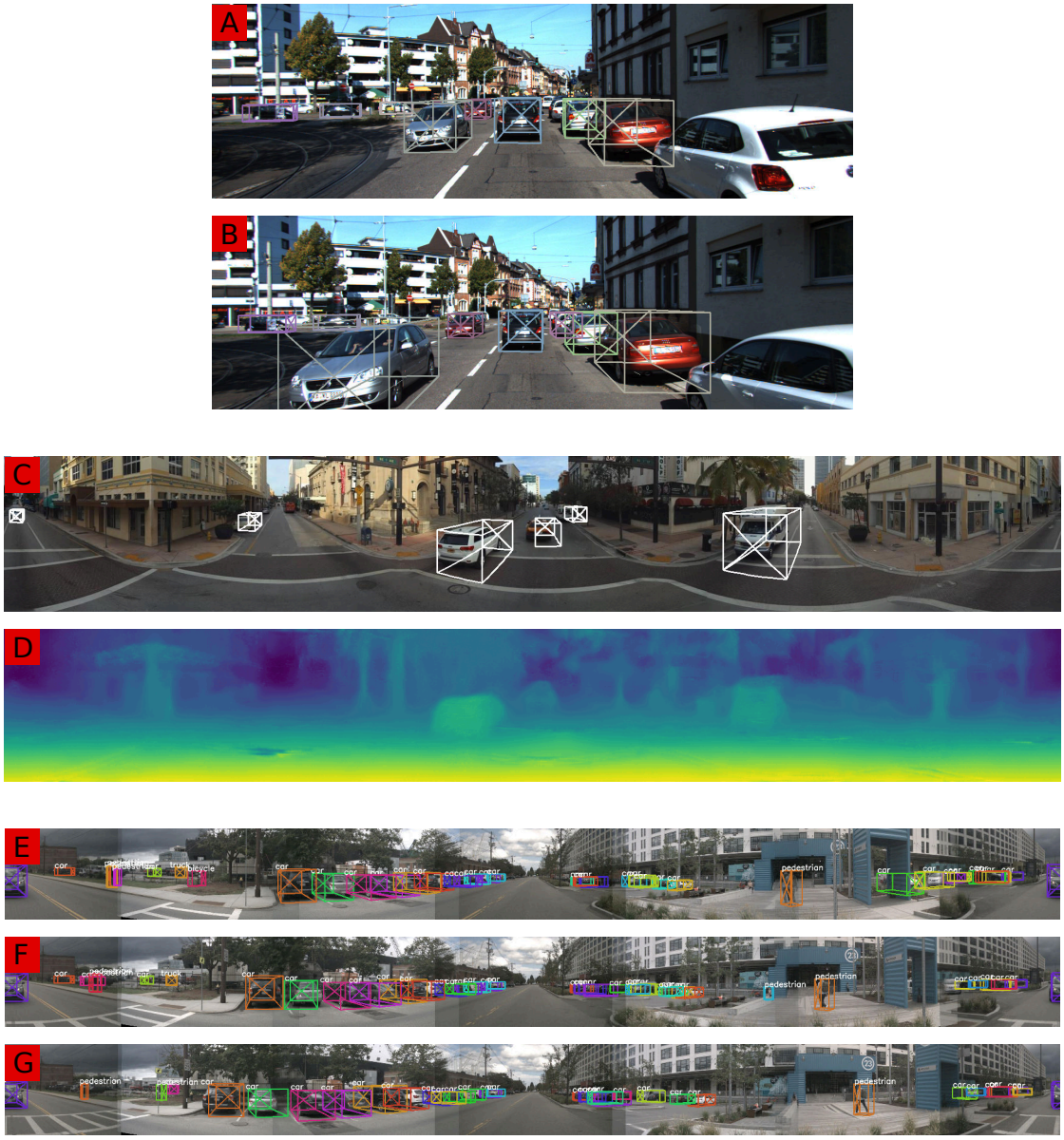


Figure 1.2: A & B: Frame-to-frame 3D detection and tracking (Chapter 3), C: 3D Detection in 360° imagery (Chapter 4), D: Depth estimation in 360° imagery (Chapter 4), E – G: Detection, pose estimation and tracking in 360° imagery (Chapter 5).

Chapter 2

Literature Review

As mentioned briefly in the introduction, we split the automotive sensing literature on situational awareness in two parts. The first part is the perception system (object detection, object tracking, depth estimation). The second part presents the key concepts of machine learning and more specifically of deep learning which we build upon for both domain adaptation and for scaling neural network training to video sequences.

2.1 Visual Perception

Visual perception is a very broad subject, which we attempt to narrow down to the topics relevant to autonomous driving. There is no clear definition of what the output of a perception system is or ought to be. In recent times, this has been defined by example via a number of benchmark-based challenges defined by their associated dataset and perception task. Traditionally, the main autonomous driving benchmark, the KITTI collection of benchmarks [8], released in 2012 based

on the dataset of the same name [1], identified several autonomous driving tasks: stereo vision, optical flow, odometry, object detection, object tracking, and road detection [46]. In 2015, a scene flow benchmark [47] and in 2017, a 3D object detection, depth estimation and depth completion benchmarks were also incorporated into KITTI.

Meanwhile, the Cityscapes dataset [48, 49] released in 2016 introduced dense pixel-wise semantic and instance segmentation. While the KITTI dataset was recorded on the streets of Karlsruhe, Germany — shown in Figure 2.1; the Cityscapes dataset [48, 49] expands to multiple cities in Germany and the recent CCSAD dataset [50], recorded in Mexico, presents some new challenges found in developing countries. Furthermore, the Mapillary Vistas instance segmentation dataset [51] based on crowdsourced imagery covers a large number of countries on all five continents [51]. In addition to this dataset, Mapillary provides a vast collection of unannotated images including 360° panoramic imagery from all over the world. In Chapter 4, we attempt to adapt existing neural networks and weights to this set of 360° panoramic imagery.

While new datasets have been released, the KITTI dataset remained, until recently, the largest dataset both in breadth and in scope for object detection and tracking. The nuScenes dataset [7] from nuTonomy released in 2019, is the first dataset to include features comparable to the KITTI dataset (*e.g.* 3D annotations) while providing a much higher number of scenes and richer set of sensors. The nuScenes dataset is also the first dataset to provide a HD map alongside the data. Similar datasets from Lyft [52], ApolloScape [53], Waymo [54, 55] have been released in 2019 with an emphasis on multi-cameras multi-sensors configurations as well as 3D object tracking.

Among the tasks proposed by those benchmarks, there are two main categories: the first provides a pixelwise labelling of the scene (segmentation, optical flow, depth estimation) which augment the existing imagery with additional per-pixel information while the second extracts information from those images (object localisation and classification, object tracking, road extraction). Interestingly, none of the aforementionned datasets and benchmarks integrates road sign reading,

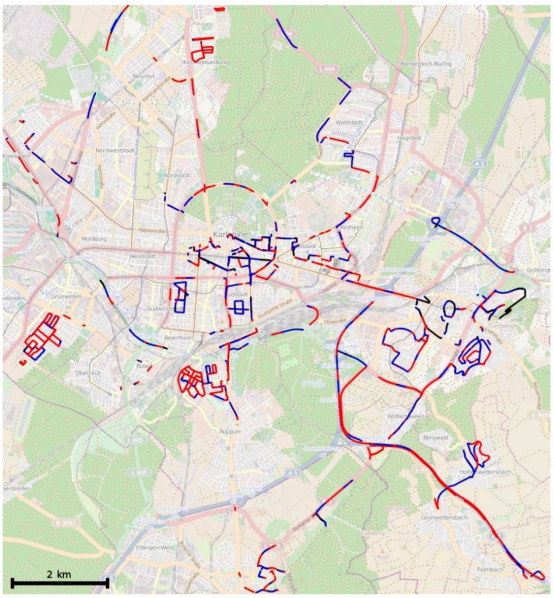


Figure 2.1: Map of the KITTI dataset captured on the streets of Karlsruhe, Germany (reproduced from Geiger *et al.* [1]).

road marking and road layout understanding. In this section, we focus on the prior work in object detection, object tracking and depth estimation which we build upon in Chapters 3, 4, and 5.

2.1.1 Object Detection

The object detection task, as shown in Figure 2.2, is, at least in automotive, concerned with finding objects, recovering their orientation and labelling them with predefined classes. In the KITTI Object benchmark, the classes are cars, pedestrians and bicycles. Immovable objects (e.g. buildings, trees, lamp posts, poles and road signs) are generally ignored. The recent Waymo Open Dataset [54] also includes road signs in addition to moving objects.

Object detection performance is evaluated using the mean Average Precision (mAP). The overlap between the ground truth and the detected objects is measured using the Intersection over Union

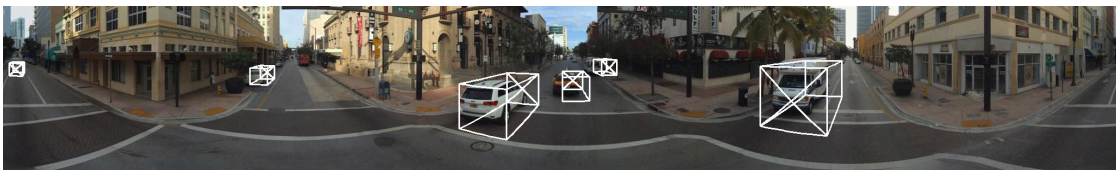


Figure 2.2: 3D Object detection example within a 360° equirectangular image from our work on panoramic imagery (see Chapter 4).

(IoU), also called Jaccard index, over the area of the 2D bounding boxes. The ground truth and predicted boxes are matched together using the Kuhn-Munkres algorithm [56] based on the computed IoU as a cost matrix. The assignments are subsequently used to compute true and false positives and negatives, precision and recall. The mAP is defined as the average precision over the recall thresholds. The KITTI dataset [8] requires a minimum overlap of respectively 0.5 and 0.7 for pedestrians/cyclists and cars for positive detections. Object detection in an automotive context presents some unique challenges due to a high number of objects, heavy occlusions, poor lighting conditions such as strong shadows and very bright regions [57]. Furthermore, safety requires a high recall especially for objects which are in the immediate vicinity of the egovehicle to guarantee that important objects will not be missed at the expense of a higher false positive rate.

Prior work relies on various inputs such as monocular imagery [34], stereo imagery [58], super-pixels segmentation [59], disparity maps and point clouds [57]. Some older approaches assume that the road had already been segmented by a prior processing step to simplify the problem; however, recent approaches based on CNN [60] do not require as much preprocessing.

Early object detection approaches based on machine learning perform an exhaustive search with a sliding window [61] to find potential objects; however, this is very computationally expensive; thus, van de Sande *et al.* introduced the concept of Selective Search [61] to identify potential candidate regions instead of searching the entire space. Sande *et al.* use Selective Search together with bag-of-words features [62] and the SVM classifier [63].

Most contemporary end-to-end CNN driven detection approaches are based on the R-CNN architecture introduced by Girshick [64] which is composed of a CNN classifier used jointly with Selective Search [61]. Successive improvements from Fast R-CNN [65] and Faster R-CNN [34] increased the performance by respectively sharing feature maps across proposals and generating the proposals using a Region Proposal Network (RPN) instead of traditional techniques based on a sliding window. The RPN is based on the principle demonstrated by Sermanet *et al.* [66] that CNNs are inherently efficient at computing a sliding window over the image. This allowed unified end-to-end training of the network to solve the combined detection and classification task. Online Hard Example Mining (OHEM) was proposed by Shrivastava *et al.* [67] to improve the training of region-based neural networks. More recently, Yang *et al.* [68] and Cai *et al.* [43] introduced a multi-scale extension of Faster R-CNN by pooling the region proposals from multiple layers in order to reduce the number of proposals needed as well as to improve performance on smaller objects such as distant objects. Our approaches, presented in Chapter 3 and 4, are based on the multi-scale approach of Cai *et al.* [43]. Lin *et al.* [69] takes the multi-scale approach a step further by transforming the network into an hourglass network.

Faster R-CNN and the aformentionned detectors are based on AlexNet [60] and VGG [3] architectures which were originally designed for image classification and later adapted to object detection [64]. Those architectures consist of a stack of convolutional layers followed by a fully-connected layer. In contrast, newer architectures such as ResNet [70] and GoogLeNet [71] are fully convolutional which does not fit with the original model of Faster R-CNN [34]. Dai *et al.* designed R-FCN [72], a fully convolutional network which fits more naturally with those new architectures.

The two stage architecture used by Faster R-CNN-based approaches presents several drawbacks. Due to the two stages, the architecture is difficult to combine with a multi-scale approach. While Cai *et al.* [43] proposed such a multi-scale architecture to generate proposals at multiple scales,

the Region of Interest (ROI) are still pooled from the last CNN trunk layer. The second stage of the network is applied to each region proposal separately, creating a large amount of duplicated work due to the lack of information sharing between proposals. This impairs the scaling of Faster R-CNN to scenes containing a large number of detections and slows the inference. In contrast, Liu *et al.* [40] introduced SSD, a single stage approach to object detection. Subsequent work by Redmon and Farhadi [73–75] on the YOLO detector provides a single stage detector which is much faster than two-stages approaches. Data imbalance between foreground and background classes is a severe problem in single stage networks which is addressed in SSD [40] using OHEM. In constrast, the RetinaNet [76] tackles the problem by adapting the cross entropy loss function to down weight easy positive and negative examples. Ren *et al.* [44] turned SSD into a recurrent network which iteratively refines the detections over time. The iterative approach of Ren *et al.* [44] adapts SSD [40] by stacking multiple hourglass networks [69]. This hourglass model can be applied iteratively to refine the quality of the detections. Zhang *et al.* [45] use a similar structure to refine the detections. Our multi-frame tracking approach presented in Chapter 5 is inspired by those stacked single shot detectors [44, 45].

Object Detection within Panoramic Imagery

Even though significant strides have been made using rectilinear imagery to generate object proposals [77] and detections by deep networks [34, 43, 64–66, 78], comparatively limited literature exists within 360° panoramic imagery. A rectilinear image is produced by projecting a scene onto an image plane using a pinhole camera model [79, Chapter 1] and the pinhole model can be approximated by rectifying the image produced by most lenses, except for fisheye lenses with a very small focal length (FoV close to 180° or more). Rectilinear imagery has the advantage of preserving straight lines and reducing distortions as perceived by humans. In contrast, 360° panoramic views cannot be represented using rectilinear images. The most common representation of panoramic views are equirectangular images. The pixel coordinates inside an

equirectangular image are specified by their latitude and longitude; in contrast to pixels in a rectilinear image which are represented using their horizontal and vertical position within the image projection onto the 2D focal plane.

Deng *et al.* [80] adapted, trained and evaluated Faster R-CNN [34] on a new dataset of 2,000 indoor panoramic images for 2D object detection. However, their approach did not handle the special case of object wrap-around at the equirectangular image boundaries.

Recently object detection and segmentation has been applied directly to equirectangular panoramic images to provide object detection and saliency in the context of virtual cinematography [81, 82] using pre-trained detectors such as Faster R-CNN [34]. Su and Grauman [83] introduce a Flat2Sphere technique to train a spherical CNN to imitate the results of an existing CNN facilitating large object detection at any angle.

In contemporary automotive sensing problems, the required vertical field of view is small as neither the view above the horizon nor the view directly underneath the camera have any useful information for those problems. Therefore, the additional complexity of the spherical CNN introduced by [83] is not needed in the specific automotive context. Instead we show in Chapter 4 how to reuse existing deep architectures built for rectilinear imagery without requiring any significant architectural changes.

Object Detection within Videos

The object detectors defined above work on still images however there are a few authors who have addressed the task of detecting objects within videos. This task overlaps the traditional definition of object detection as well as tracking. In this section, we focus on the detection aspects rather than the tracking aspects whilst tracking is surveyed in Section 2.1.2. Object detectors in videos face one challenge: motion blur and compression artifacts which are not present in the still

images used in most image classification and object detection benchmarks such as ImageNet [84] and Pascal VOC [85] however this domain bias does not exist in automotive because models are usually trained and tested on the same automotive datasets [7,8], which have been extracted from video sequences recorded on road rather than still imagery. In addition, video-based detectors exploit temporal consistency to improve detection quality.

Tripathi *et al.* [86] defines a Recurrent Neural Network (RNN) based on a Gated Recurrent Unit (GRU) [87] to refine the location of detections over time. It is based on a weakly-supervised loss function which enforce consistency across frames rather than accuracy with the ground truth. It assumes that the detections have been grouped into tracklets by an external tracker. Meanwhile, SeqNMS [88] extends the Non Maximum Suppression (NMS) commonly used in detectors to work across time. Kang *et al.* [89–91] introduce the notion of tubelet as an extension of the concept of object proposals to videos and use a Long Short Term Memory (LSTM) [92] to refine the confidence score and location of each object across frames. Similarly, Galteri *et al.* [93] uses the object detections from the previous frame as a prior for object proposal generation. Ning [94] introduce ROLO, a single object tracker refining the location of objects using an LSTM. Unlike the prior work, it also integrates the last feature map of YOLO into the tracker to provide a richer input to the temporal network. Lu *et al.* [95] introduce an association LSTM network which jointly associate detections and tracklets as well as refining the object location.

Chen *et al.* [96] introduce TSSD, a network based on VGG16 [3] as well as a new type of recurrent unit called convLSTM, an extension of LSTM [92] using convolutions rather than matrix multiplications. Unlike the prior work, TSSD is a single end-to-end network. The two recurrent layers work on the convolutional feature maps rather than a sparse set of detections. The network consists of an attention mechanism which relies on both the current frame and the hidden state to select useful parts of the image. While the original network is only a detector, Chen *et al.* subsequently extended it to a tracker [97].

In 2019, Voigtlaender *et al.* [98] introduce an extension of the KITTI dataset for multi-object tracking and segmentation (MOTS). In addition to the dataset, they introduce a baseline method based on Mask R-CNN [78] for joint tracking and segmentation. Instead of using a recurrent network, they use 3D convolution to merge the information of 8 frames before the RPN. Zhang and Kim [99] extend the notion of temporal convLSTM [96, 97] by warping the hidden state feature maps using the optical flow generated using FlowNet [100].

Most of the aforementioned approaches add an extra temporal layer at the end of the network [86, 94–97] while more recent approaches integrate temporal knowledge earlier before the creation of object proposals [89–91, 93, 98] for two-stages detectors — such as Faster R-CNN [34]. In contrast, in Chapter 5, we propose a new single-shot architecture which integrates the temporal information at different scales throughout the network.

2.1.2 Object Tracking

Tracking algorithms can be divided into two broad categories: the first are template-based approaches which learn a complex representation of an object to be able to recognize it in subsequent frames while the second category called multi-object tracking-by-detection (MOTD), as shown in Figure 2.3, leverages motion continuity to link detections across frames to form tracklets. In this section, we focus on the later which is more relevant to autonomous driving. Leal-Taixé *et al.* [41] and more recently, Ciaparrone *et al.* [42] provide a comprehensive overview of the state of the art in multi-object detection.

MOTD is most commonly evaluated using the multiple object tracking precision (MOTP) and multiple object tracking accuracy (MOTA) metrics defined by Bernardin and Stiefelhagen [101]. MOTP is a measure of the quality of the generated bounding box position based on the IoU while MOTA is an aggregate of a ratio of misses in the sequences, a ratio of false positives and a ratio of mismatches. Unlike the mAP, the confidence score of the detections is not taken

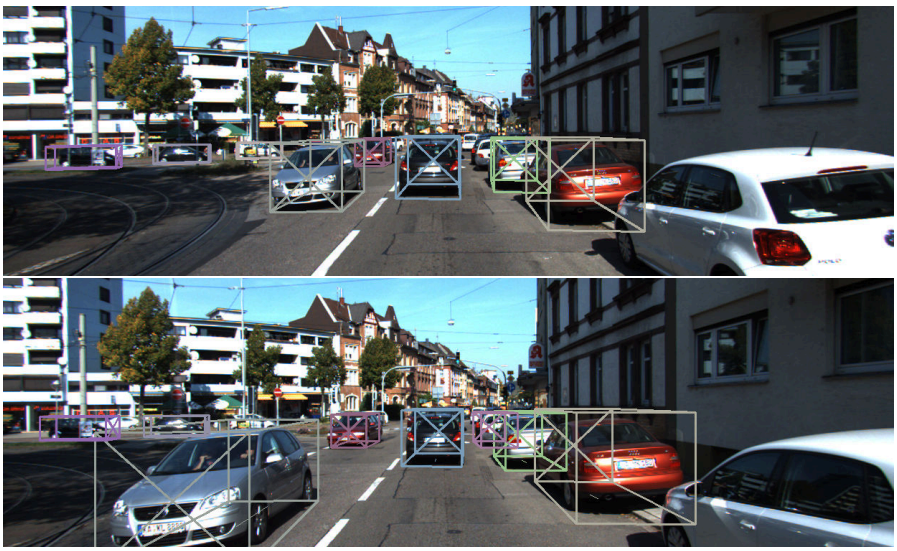


Figure 2.3: Object tracking and 3D pose estimation example from our work in Chapter 3. The two images are four frames apart, that is 400 ms apart.

into account in the definition of the metric. Even if such a confidence scores exist, MOTA and MOTP expect that low confidence detections would have already been filtered out. Contrarily, the recently released nuScenes tracking benchmark [7] uses the Average MOTP (AMOTP) and Average MOTA (AMOTA) respectively defined by Weng and Kitani [102] which are averages of the MOTP and the MOTA over the recall thresholds. Weng and Kitani [102] also define a variant of MOTA called recall-normalised MOTA (MOTAR) which introduce a recall-normalisation term to prevent the MOTA from being negative. Li *et al.* [103] introduce two other metrics: mostly tracked (MT) and mostly lost (ML) which are the number of groundtruth tracklets successfully tracked for more than respectively 80% and 20% of their length.

Graph-based and Feature-based Approaches

Early work addresses the MOTD task using Bayesian inference [103]. A set of studies have shown that the Bayesian formulation can be expressed as a min-cost flow graph solvable in polynomial

time [104, 105] online [106, 107] however there is no guarantee that the formulated model is an accurate representation of the problem. This has led to numerous further publications on graph-based pedestrian tracking [108–113]. The models are usually based on a pairwise detection association cost using features such as 2D bounding box position, size and visual appearance statistics such as colour histogram [109], pedestrian shape template [108], and LBPH [114]. Osep *et al.* [115] and Sharma *et al.* [116] have recently shown that the performances can be improved using 3D pose as an additional feature. To compute this 3D pose, they merge 2D detections with 3D proposals generated from stereo imagery. Sharma *et al.* [116] also use a pairwise appearance cost based on an appearance feature descriptor generated by an hourglass CNN. Instead of using appearance, Tang *et al.* [112] rely on optical flow [117] in the pairwise costs of the minimum cost multicut formulation. Wang *et al.* [118] combines a graph-based approach with deep learning using convolutions. The drawback of this approach is that it works on a fixed time window of 64 frames.

RNN-based Approaches

The LP-SSVM algorithm, introduced by Xiang *et al.* [119], learns to track objects using a Markov Decision Process trained using Reinforcement Learning. Deep learning has been used for template-based single target tracking [120–122] however the first use of deep learning to solve multi-target tracking is a RNN by Milan *et al.* [123]. While they note that this is a step toward end-to-end tracking using neural networks, it still relies on the output of a separate detector. Lu *et al.* [95] use a similar approach based on two stacked LSTM layers [92]. Conversely, Gaidon *et al.* [124] use a probabilistic framework based on the last feature maps produced by a CNN-based detector. Gaidon *et al.* still train the detection network separately from the tracking linear classifier. Similarly, Zhou *et al.* [125] integrate CNN feature map from a small region of interest into a tracking framework called Deep Continuous Conditional Random Field (DCCRF). While not an end-to-end solution, the detector is still able to use information-rich feature maps rather

than a limited set of object attributes.

Zhang *et al.* [126] present an end-to-end network for single-object tracking trained with back-propagation and reinforcement learning. Sadeghian *et al.* [127] model the interactions between objects over an occupancy grid using an LSTM [127]. Ma *et al.* [128] takes a different approach: using tracklets already pre-built using a simpler approach such as the Munkres-Kuhn algorithm [56], they introduce a network able to correct the mismatches and improve the quality of the detections. Kim *et al.* [129] use bilinear LSTM, which is based on the intuition that while the additive coupling of LSTM is efficient at storing motions, it is far less efficient at storing object representations.

Tracking by Re-Identification

Graph-based and RNN-based approaches primarily rely on the analysis of motion and the behaviour of the targets to associate detections and tracklets. Those approaches are not robust to targets which leave the camera field of view then re-enter at a subsequent time. This task is called Re-Identification (ReID) and can be used instead of tracking the motion of object to build a tracker. Tang *et al.* [113] introduce the concept of lifted edges as a new type of edge in a tracking graph to integrate person re-identification within a graph-based tracker. Ristani *et al.* [130] learn object appearance embeddings using a triplet loss and use those embeddings to cluster detections together. Zhang *et al.* [131] generates short tracklets and subsequently merge them across multiple views using ReID based on a hierarchical clustering. Luiten *et al.* [132] propose a 3D tracking approach through occlusions which does not use any appearance model but rather interpolate the trajectory of objects through occlusions.

Re-Identification is most useful to track targets over a long time across multiple views. In contrast, tracking for autonomous vehicle is only concerned with a more immediate timescale of a few seconds. If a vehicle leave the field of view and re-enter later, a new ID can be assigned

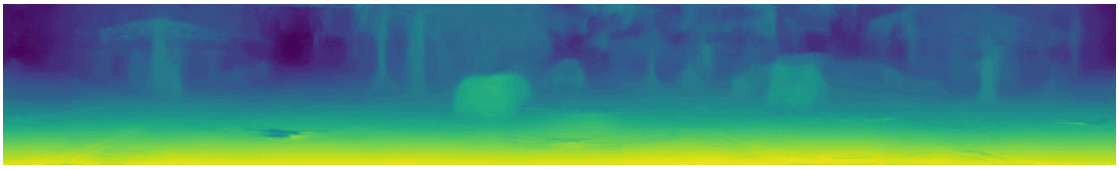


Figure 2.4: Monocular depth estimation example within an equirectangular image from our work on 360° panoramic images (see Chapter 4).

without any consequences in terms of driving. In addition, strong shadow, poor lighting and similar colours make the ReID task quite challenging in autonomous driving scenarios.

End-to-end Approaches

In contrast, there are few approaches based on end-to-end tracking. Leal-Taixé *et al.* [133] use a siamese CNN to compute the association cost of each pair of detections. Feichtenhofer *et al.* [134] contruct a siamese network based on R-FCN [72] with an additional correlation stage to compute the association cost matrix between the objects of two images. In 2019, Voigtlaender *et al.* [98] use an end-to-end network with a layer of 3D convolutions. For each frame, it learns the location of the object in the previous frame and uses the information to link each frame together. The downside of end-to-end tracking is the difficulty of acquiring the large amount of annotated video sequences as well as the cost of training a large network relative to the shallower networks used for ReID and RNN-based strategies. In Chapter 5, we propose a new end-to-end approach to tracking with current features at multiple scales and a frame-to-frame association approach similar to Voigtlaender *et al.* [98].

2.1.3 Depth Estimation

Traditionally, depth estimation, as shown in Figure 2.4 is recovered using multi-view approaches such as structure-from-motion [47, 135] and stereo vision [136], relying on an explicit handling of geometrical constraints between multiple calibrated views. In multi-view approaches, the overall

process is typically split into two separate steps: keypoint matching and geometric optimisation [136]. The keypoint matching is based on a range of visual cues: geometric cues are found in sparse keypoints such as SIFT [137], SURF [138] feature descriptors as well as dense descriptors such as HOG [139, 140]; visual similarity cues are found in dense approaches based on cross-correlation such as Normalised Cross Correlation (NCC) [136]. Knowing a set of either dense or sparse matches between a set of multi-view images, the most likely depth estimate can be determined using geometric reasoning. Since the problem is typically both under-constrained (textureless areas, poor lighting) and subject to noise (reflections, specularity), it is framed as a minimisation problem by regularising the geometric constraints and adding smoothness penalty terms. This problem can be solved locally using Dynamic Programming or globally using approaches such as Graph Cuts or Belief Propagation. However, many approaches rely on Semi-Global Matching (SGM) [141] which is a compromise between local and global optimization using an astute backtracking technique. SGM is relatively fast and can be parallelized on the Graphic Processing Unit (GPU) for real-time efficiency. Modern stereo approaches such as Kendall *et al.* [142] take advantage of the expressiveness of CNN while retaining geometric considerations such as the disparity-space cost function.

In contrast, geometric considerations can be used for depth estimation in a single monocular image up to a scale factor and even then this is a severely underconstrained and challenging task. In addition to geometric cues, it is necessary to rely on contextual or semantic cues in monocular imagery. Saxena *et al.* [143, 144] provides the first approach that exploits such cues. It is based on machine learning using a Markov Random Field (MRF) model to estimate dense depth maps.

Dense Monocular Depth Estimation After the initial success of classical learning-based techniques [143, 144], depth recovery was first approached as a supervised learning problem by

the depth classifier of Ladický *et al*. [145] and deep learning-based approaches such as [9, 146]. However, these techniques are based on the availability of high-quality ground truth depth maps, which are difficult to obtain. In order to combat the ground truth data issue, the method of Atapour-Abarghouei and Breckon [147] relies on readily-available high-resolution synthetic depth maps captured from a virtual environment and domain transfer to resolve the problem of domain bias.

On the other hand, other monocular depth estimation methods have recently emerged that are capable of performing depth recovery without the need for large quantities of ground truth depth data. Zhou *et al*. [148] estimate monocular depth and ego-motion using depth and pose prediction networks which are trained via view synthesis. Kuznietsov *et al*. [149] utilises a deep network semi-supervised by sparse ground truth depth and subsequently reinforced within a stereo framework to recover dense depth information.

Godard *et al*. [38] train their model based on left-right consistency inside a stereo image pair during training. At inference time, however, the model solely relies on a single monocular image to estimate a dense depth map. Even though the said approach is primarily designed to deal with rectilinear images, in this thesis, we further adapt this model to perform depth estimation on equirectangular panoramic images in Chapter 4.

Sparse Monocular 3D Object Detection In contrast to dense methods, it is also possible to recover the 3D pose of objects of interest in monocular imagery.

Prior work on 3D pose regression in panoramic images is mostly focused on indoor scene reconstruction such as PanoContext by Zhang *et al*. [150] and Pano2CAD by Xu *et al*. [151]. The latter retrieves the object poses by regression using a bank of known CAD (Computer-Aided

Design) models. In contrast, our method does not require any *a priori* knowledge of the object geometry.

While most of the work has focused on 2D detection, the work of Chen *et al.* [57,152] leverages 3D pointcloud information gained either from stereo on LIDAR modalities to generate 3D proposals which are pruned using Fast R-CNN. Whereas these works use complex arrangements using stereo vision, handcrafted features or 3D model regression, recent advances [4, 153, 154] show that it is actually possible to recover the 3D pose from monocular imagery. Chen *et al.* [153] use post-processing of the proposals within an energy minimization framework assuming that the ground plane is known. Chabot *et al.* [154] use 3D CAD models as templates to regress the 3D pose of an object given part detections. While Mousavian *et al.* [4] show the 3D pose can be recovered without any template assumptions using carefully-expressed geometric constraints. In this thesis, we propose a new approach in Chapter 3, similar to Mousavian *et al.* [4], without explicitly-expressed geometric constraints. It performs well on both rectilinear and equirectangular panoramic imagery without any knowledge of the ground plane position with respect to the camera. We further extend this approach in Chapter 5 to a single-shot detector.

2.1.4 Perception Summary

The previous sections have provided an insight in the current state-of-the-art of perception applied to the automotive domain. Some algorithms such as stereo vision, optical flow and tracking have been studied for a long time; while object detection and monocular depth estimation algorithms have advanced rapidly in the last few years. 360° panoramic imagery is a relatively new field which has not been studied in the context of automotive. Furthermore, the work of Kendal *et al.* [155] has shown that it is possible to jointly learned multiple tasks efficiently with a single neural network. Our work hinges on the study of 3D object detection, tracking, and depth estimation in both rectilinear and 360° panoramic imagery.

2.2 Machine Learning and Deep Learning

Computer vision and in particular in the field of autonomous driving has proven to be a particularly challenging if not intractable task to solve exclusively using hand-crafted features and algorithms. While such algorithms can achieve reasonable results, especially on toy examples, they are ill-suited to the complexity of real-world tasks. In order to hand design such algorithm, one must consider an enormous number of interrelated aspects and in particular the interaction between each of those aspects. It is extremely difficult to achieve a separation of concerns in computer vision. The lack of separation of concerns and the exponential number of interactions leads to the intractability. The introductory chapter of the seminal text by Szeliski [36] provides an in-depth explanation and discussion about this challenge. Koller and Friedman [156] demonstrate in probabilistic terms that exact inference on models of such complexity is intractable and one must instead rely on approximate inference. This result is particularly important for the mission-critical and safety systems required in autonomous vehicles where exact inference would have been desirable but an unachievable property. Instead, one must turn to machine learning to efficiently solve these problems.

This thesis is based on the concepts of deep learning. Schmidhuber produced an extensive survey of deep learning [157] while LeCun published an extensive overview [158]. While shallow neural networks have been around since the 1960s, deep learning was considered impractical in the 1980s due to the computational cost, however it became viable in the mid-2000s with the advent of fast GPUs [157], eventually breaking records in the MNIST handwritten digit recognition challenge [159] in 2010 and ImageNet image classification challenge [60] in 2012. Since then, deep learning-based approaches have surpassed the existing state-of-the-art on many tasks as illustrated in Section 2.1.

In this section, we discuss two aspects of machine learning which are particularly relevant to this thesis: dataset bias and domain adaptation as well as training scaling to video sequences.

2.2.1 Domain Adaptation and Style Transfer

Machine learning models trained on one dataset do not necessarily transfer well to a new dataset — a problem known as dataset bias [160] or covariate shift [161]. A simple solution to dataset bias would be fine-tuning the trained model using the new data, however that often requires large quantities of ground truth, which is not always readily-available.

While many strategies have been proposed to reduce the feature distributions between the two data domains [162–165], a novel solution was recently proposed in [147] which uses image style transfer as a means to circumvent the data domain bias.

Image style transfer was first proposed by Gatys *et al.* [166] but since then remarkable advances have been made in the field [167–170]. In this work, we attempt to transform existing rectilinear training images (such as KITTI [1, 8]) to share the same style as our panoramic destination domain (Mapillary [171]). However, these two datasets have been captured in different places and share no registration relationship. As demonstrated in [147], unpaired image style transfer solved by CycleGAN [2], can be used to transfer the style between two data domains that possess approximately similar content.

In Chapter 4, we use style transfer and more specifically CycleGAN [2] to adapt existing neural networks to 360° panoramic imagery (as shown in Figure 2.5).

2.2.2 Training at Scale

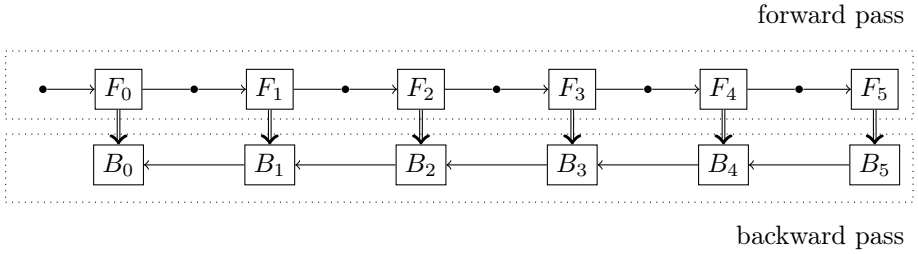
While significant work has focused on scene understanding from still images, video processing is much less studied [172, 173]. Typical video tasks are video classification (*e.g.* type of video or action recognition) [172] and captioning [173]. Deep learning-based video object detection and tracking are often split into a still image object detector followed by a recurrent temporal network such as [95, 128, 129] to reduce complexity. The main difficulty is the limited amount

Figure 2.5: Domain adaptation example using CycleGan [2]. A: Original image. B: Image transferred to the second domain. C: Reconstructed image back to the original domain.

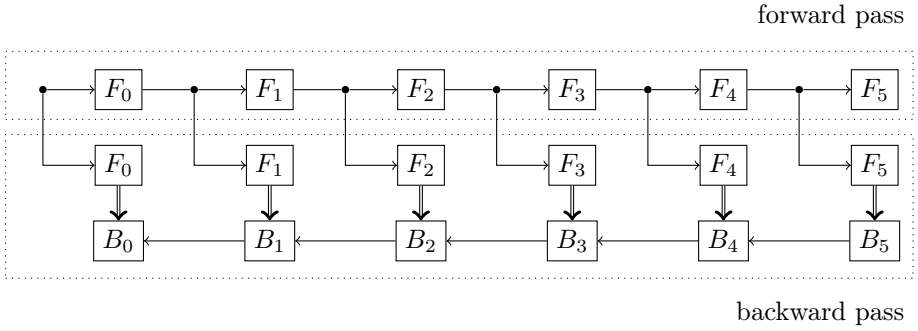
of memory which is available in the processing units (*e.g.* typically 12Gb RAM on a standard GPU). Classical neural network training approaches extensively rely on gradient backpropagation throughout the network. The gradient computation is based on two phases: a forward phase through the entire network followed by a backward phase. All the intermediaries results from the forward phase must be kept in order to perform the backward phase. Therefore, the memory required to train a deep learning model increases linearly with the number of frames processed by the model. Therefore, the typical algorithm would sample a small fixed number of frames from the sequences, such that the model can be trained within the memory constraints. Furthermore, in order to train using more than 2 or 3 frames at a time, the frames must be shrunk to a small size such as 512×512, 224×224 or sometime even smaller [172].

Recent works [174–176] have shown that it is possible to implement strategies to trade-off computation time for memory. Such strategies curb the memory requirements from linear complexity to sublinear complexity. Those techniques have been successfully applied to train larger networks [177–180]. Furthermore, it is possible to store intermediaries gradients either on the main memory or disk storage. For instance, Figure 2.6 shows, without loss of generality, how a neural network processing a video sequence of 5 frames can be optimised using this method to reduce the amount of memory which must be simultaneously stored between the forward and the backward passes.

Those difficulties are only present during training. Backpropagation is not required for inference using the trained models. Hence the memory-wise inference complexity is typically constant regardless of the number of frames and regardless of the training complexity. We exploit this paradigm in Chapter 5 to train a large memory-intensive tracking network which scales to an arbitrary number of video frames.



(a) Naive implementation. The internal state of all F_t must be stored during the forward pass to be used in the backward pass.



(b) Scalable implementation. The internal state is not stored in the forward pass. Instead each forward function F_t is replayed as required in the backward pass, saving much memory at the expense of a slight increase in computation time.

Figure 2.6: Backpropagation through time for a recurrent network over 5 frames. F_t represents the forward pass and B_t, the associated backward pass, at frame t. Each small dots is a hidden state between frames. Each double arrow is the internal state of the forward function F_t which is required by the complementary backward function B_t. The internal state is much larger than the hidden state.

2.3 Summary

We have identified that 360° panoramic imagery has a new set of challenging problems in automotive visual sensing. In particular, we address in this thesis the problems of object detection, 3D pose estimation, and multi-object detection and tracking within monocular 360° panoramic imagery.

We have introduced the contemporary state of the art in object detection in Section 2.1.1 and object tracking in Section 2.1.2 which we build upon throughout this thesis. In Chapter 3, we build a siamese [37] 3D object detection and tracking network based on MS-CNN [43]. We have identified a lack of prior work and datasets on panoramic imagery in automotive application; which we attempt to fill this gap in Chapter 4 by introducing a novel approach to adapt existing training datasets and CNN to panoramic imagery using CycleGAN [2]. We use this methodology to adapt our neural network presented in Chapter 3 as well as Monodepth [38] to 360° panoramic imagery. This approach is based on the concepts of domain adaptation presented in Section 2.2.1 to generalise from the KITTI dataset [8] to our own 360° synthetic imagery testing dataset. We attempt to depart from the tracking literature in Chapter 5 by integrating object detection and tracking together in a single end-to-end CNN using the concepts of neural network training presented in Section 2.2.2. This network is a multi-scale single shot detector inspired by [44, 45].

Chapter 3

Combined Multi-Object Frame-to-Frame Detection, Tracking and 3D Pose Estimation within Monocular Video Imagery

3.1 Introduction

Object detection and tracking is an essential component of the perception subsystem of any self-driving vehicle. Tracking can be based on either LIDAR and RADAR information or camera video streams. We will focus on the later. The MOTD literature treats the object detection and tracking as two distinct problems whereby the output of the former is used as the input of the

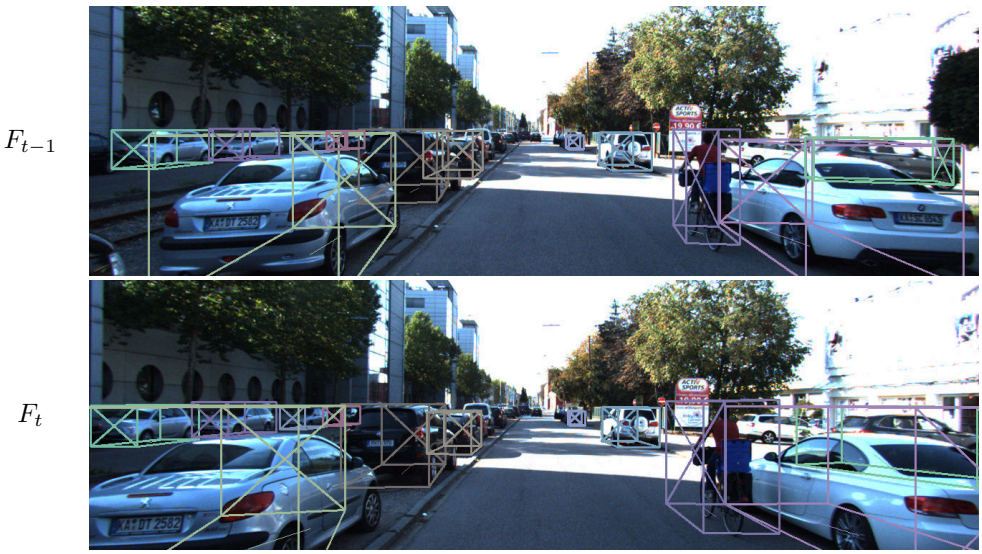


Figure 3.1: Examples of detection results. Top row shows the results on the previous frame and bottom row shows the result on the current frame.

later [104,105,107]. As presented in Section 2.1.2, such approaches limit the amount of knowledge transferred between the detector and the tracker. The tracker input detections are typically represented using a 2D bounding box and some visual appearance embedding. In challenging yet common driving scenarios, the information provided by the appearance embedding is somewhat limited because distant objects under shadow or strong contrast due to sunlight have very similar appearances. Besides such a detector has no notion of time and a tracker has no notion of geometric cues which are not contained in the detection themselves.

Another essential aspect of the perception subsystem is the 3D reasoning ability. Specifically, 3D pose estimation is relevant to object detection and tracking. While 3D reasoning is an intrinsic part of LIDAR- and RADAR-based approaches, it is absent of most detection approaches based on monocular imagery. Depth perception from imagery is traditionally approached using stereo vision [136]. In contrast, as explained in Section 2.1.3, depth and 3D pose estimation by geometric reasoning in monocular imagery is a severely underconstrained task. Machine learning

is particularly suited to integrate semantic reasoning to resolve the geometrical ambiguities as proven by the work of Mousavian *et al.* [4] and Xiang *et al.* [5] presented in Section 2.1.3. We propose an improved 3D pose estimation method using a simpler formulation than earlier work.

In this chapter, we present a novel algorithm which jointly solves object detection, 3D pose estimation and frame-to-frame object tracking. We show that frame-to-frame tracking can be formulated by teaching an object detector based on Faster R-CNN [34] to process two consecutive frames to jointly output the 3D pose, size and velocity of each detections in each frame as well as a mapping of object positions between the two frames (as shown for example in Figure 3.1). Our frame-to-frame approach enables us to directly estimate the velocity of detected objects. We also show that such velocity estimates are more accurate than calculating them as the difference between past and present locations.

3.1.1 Proposed Contributions

The key contributions, against the state of the art [4, 5] outlined in this chapter are:

- improved monocular 3D pose regression of vehicles, pedestrians and cyclists using a simpler formulation than earlier work [4, 5] which achieve state-of-the-art results;
- a novel frame-to-frame detector based on Faster R-CNN to recover the velocity of objects as well as a frame-to-frame mapping of those objects.

3.2 Approach

We first describe the overall network architecture (Section 3.2.1). Subsequently, we describe how we leverage this architecture to estimate 3D pose (Section 3.2.2) and velocity (Section 3.2.3). Finally, we show how we exploit this information for object tracking (Section 3.2.4).

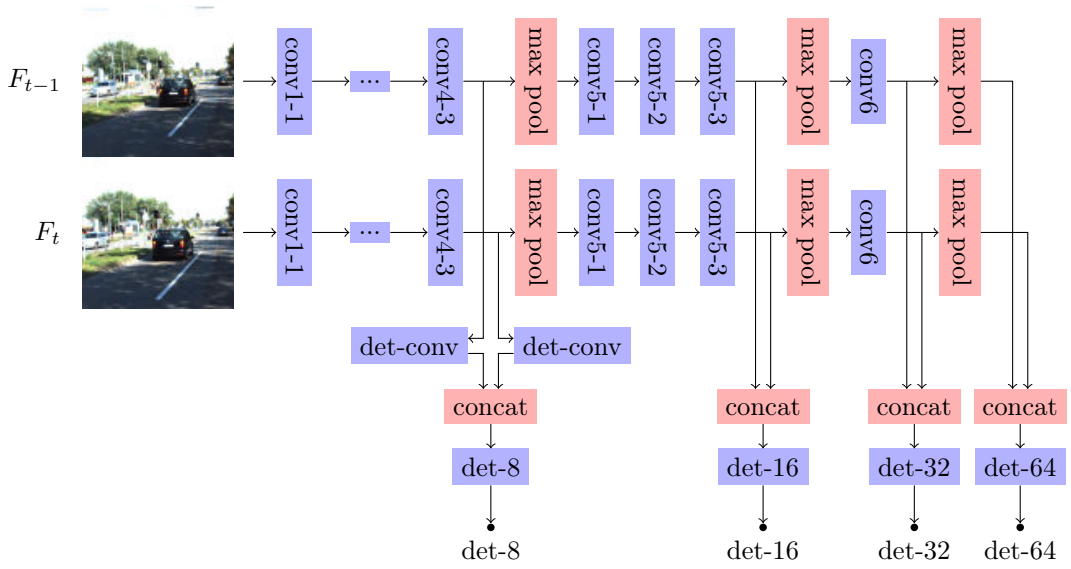


Figure 3.2: RPN subnetwork. Convolutional are shown in blue, pooling and concatenation layers in red. Layers with the same names share the same weights if any. The layers *conv1-1* up to *conv6* are pre-trained VGG16 [3] layers.

3.2.1 Network Architecture

Primarily we leverage the existing multiscale architecture of Cai *et al.* [43] called MS-CNN, however this work can be adapted to any of the derivatives of Faster R-CNN [34]. This network generates a sequence of detection proposals using an RPN and then, in the second-stage detection network, pools a subregion around each proposal to further regress the proposal 2D location. We extend this network to work on two successive frames at a time using a siamese architecture [37] inside the RPN then concatenate the feature maps of both frames in the detection network to regress the 2D location as well as 3D pose and velocity estimation.

As illustrated in Figure 3.2, the RPN network takes an input of both the current (F_t) and previous (F_{t-1}) frame from a monocular video stream. Images are upsampled by a factor of 1.5 to the resolution 1920×576 then each frame of this pair (F_t, F_{t-1}) is processed via a dedicated

trunk of VGG16 [3] 3×3 convolution layers (Figure 3.2, *conv1-1* to *conv5-3*) and 2×2 max-pooling layers (Figure 3.2, *max pool*). Those trunks form a siamese network with shared weights for each convolution layer. Subsequently at four different scales, CNN feature maps (Figure 3.2, *conv4-3*, *conv5-3*, *conv6* and the last *max pool*) from each trunk are pulled and concatenated together and then fed into a set 5×5 convolution layers (Figure 3.2, layers *det-8* to *det-64*). The concatenation followed by a convolution is designed to capture relationships between the two consecutive frames by merging the feature maps from the frames F_t and F_{t-1} together. For each location, the detection convolutional layers (*det-8* to *det-64*) produce the 2D bounding box of two ROI named $\mathcal{P}_t$ and $\mathcal{P}_{t-1}$ for respectively the current and the previous frame (F_t, F_{t-1}) as well as an objectness score (as introduced in Faster R-CNN [34]). Those ROI which may contain an object of interest are called region proposals.

The position of the region proposal at each anchor in the RPN is regressed using a smooth L_1 loss [65] and the objectness score is regressed using a binary cross entropy loss function as described by Ren *et al.* [34]. As explained in Section 3.3, we use a two stage training. In the first stage, the RPN is trained on its own using the smooth L_1 loss and cross entropy; while in the second stage, it is further trained jointly with the detection network described below.

As most anchors do not contain any object at all and others are for the same objects; we apply NMS with a 2D IoU threshold of 0.65 and select the 3000 highest scoring boxes.

Subsequently, the detection network (shown in Figure 3.3) further regresses the exact bounding box and class label of each of the 3000 remaining proposals as well as their full 3D pose and velocity. As shown in Figure 3.3, at the beginning of the detection network, for each proposal, we pool a ROI from *conv4-3* defined by the bounding box encompassing the regions $\mathcal{P}_t$ and $\mathcal{P}_{t-1}$ for respectively the current and previous frames because it is desirable to use a single common ROI for both frames rather than disjoint ROIs per frame to accurately compute the

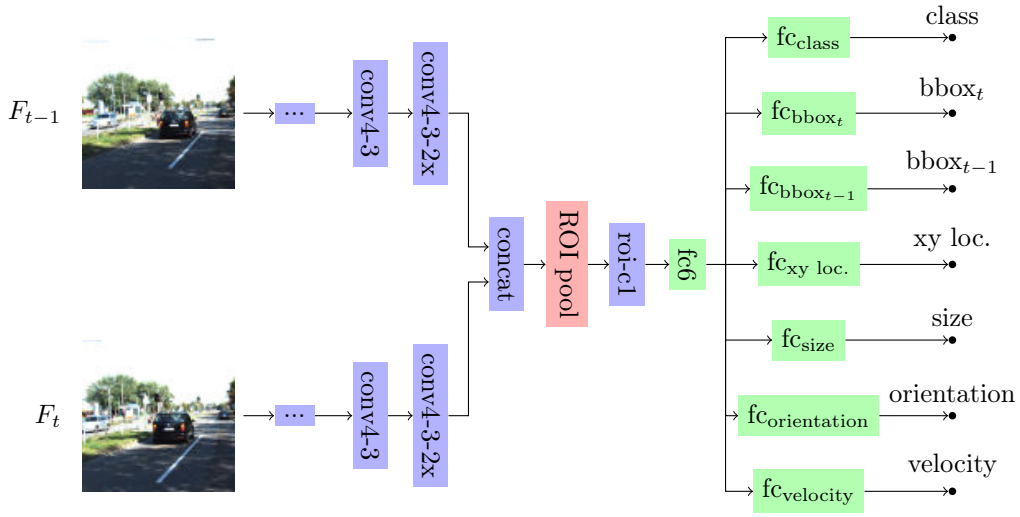


Figure 3.3: Detection network. Feature maps from two ROI pooling layers (one for each frame) are concatenated together then processing through a deconvolution layer (*roi-c1*) and a fully-connected layer (*fc6*). Each output is predicted using one further separate fully-connected layer.

velocity of an object. As shown in Figure 3.3, the two *conv4-3* layers are upsampled using a transposed convolution layer (*conv4-3-2x*) of kernel size 4 and stride 2 applied on each channel independently, then concatenated together feature-wise then pooled. The pooled feature map is then fed through the fully-connected layers (Figure 3.3, *roi-c1*, *fc6*). The layer *roi-c1* has 512 input channels (same as the layer *conv4-3* of VGG16 [3]) and 1024 output channels; while *fc6* has both 1024 input and output channels. We use the last common feature map *fc6* to learn features (shown in green in Figure 3.3) using a separate subsequent fully-connected layer per feature.

While the approach aggregates information from two siamese VGG [3] trunks, during inference on a video stream, we can reuse the VGG trunks (Figure 3.2, *conv1-1* up to *conv6*) from one frame to the next at each time step because the weights are shared across the two VGG trunks to save runtime.

As illustrated on Figure 3.3, the network outputs for each object, its current 2D bounding box and 3D pose in frame F_t, its previous 2D bounding box and 3D pose in F_{t-1} (hence connecting objects in F_t to F_{t-1}), its size, and its velocity. Each of the output and associated loss function is described in the next sections and the method to jointly train this set of loss is defined in Section 3.3.

3.2.2 3D Pose Recovery

While Mousavian *et al.* [4] shows that 3D pose can be estimated without any assumptions of known 3D templates, their algorithm relies on geometric properties. In contrast, we regress the 3D pose directly, simplifying the computation and making it easier to adapt to panoramic images in the subsequent work in Chapter 4.

Here, we directly regress the 3D dimension (width, length and height) in meters of each vehicle using a fully-connected layer as well as the orientation as per [4]. Moreover instead of relying on geometric assumptions, we also regress a quantity which we name the object disparity $d = \frac{f}{z}$ which is the inverse of the distance z multiplied by the focal length f. This definition of the disparity is analogous to the definition used in stereo vision ($d = \frac{fB}{z}$) [36, Chapter 11]. The size of a given object in image space depends on its physical size, distance and the focal length of the camera therefore it is not possible for a neural network to learn the distance z based on object appearance in image space unless the focal length is a constant. This assumption is not desirable because it would prevent the network from generalising to other cameras and lenses. Besides, since the focal length associated with a given ROI depends on the size of the ROI, it is desirable to learn a quantity which is independent from the focal length. In constrast, the disparity d given the object appearance in image space is independent from the focal length. Using a fully-connected layer connected to the last fully-connected layer *fc6*, we learn coefficients a, b such that:

$$d = ah_{roi} + b \tag{3.1}$$

where h_{roi} is the height of the region proposal generated by the RPN in pixels. To simplify the computation, we also learn the 2D projection of the centre of the vehicle onto the image (u, v) using another fully-connected layer. As a result, we can recover the 3D position (x, y, z) using:

$$[x, y, z]^T = zP^{-1}[u, v, 1]^T \tag{3.2}$$

where P is the camera matrix obtained from earlier camera calibration [79, Chapter 1].

For network training of our model, we additionally use data augmentation including image cropping and resizing as defined by [43]. Any of those operations on the image must be accompanied by the corresponding transformation of the camera matrix P in order to facilitate effective training.

As noted by Mousavian *et al.* [4], estimating the angle of an object pose a significant challenge because the signed angle of $+\pi$ radians and of $-\pi$ radians along an axis of rotation represent the same 2D rotation; thus when regressing angles, there is a discontinuity at π radians. Confronted with such an ambiguity, a naive regression using the mean-square error would choose the average of the two extremas rather than the actual angle for objects with an orientation of $\pm\pi$. To circumvent this problem, given the object yaw θ (orientation on the ground plane), we instead learn $c = \cos^2\theta$ and $s = \sin^2\theta$ which are both independent of the directionality. Noting that $\cos^2\theta + \sin^2\theta = 1$, c and s can be conveniently learned with a fully-connected layer followed by a *softmax* layer. For each pair (c, s), there are four possible angles each in a different quadrant depending on the sign of the sine and cosine:

$$\hat{\theta} = \arg\left(\pm\sqrt{c} \;\pm \sqrt{s}\,\mathbf{i}\right) \tag{3.3}$$

We further discriminate between the four quadrants using a separate classifier consisting of a fully-connected layer followed by a *softmax* classification layer.

Following the convention of the KITTI dataset [8], we assume that the pitch and roll are null however it is possible to extend this representation to arbitrary 3D rotation θ around a vector $\mathbf{u}$ given that it can be expressed as the quaternion $q = \cos\frac{\theta}{2} - (u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k})sin\frac{\theta}{2}$. This adds an extra 3 parameters to learn $\mathbf{u}$ which can also be learned with a *softmax* layer because $|\mathbf{u}| = 1$.

3.2.3 Temporal Continuity Information

For each object, we directly learn the 3D velocity vector v in meters relative to the egovehicle between frames F_{t-1} and F_t using a fully-connected layer. We also regress the 2D bounding box and 3D pose of the object in frame F_{t-1} using the same method as for F_t.

3.2.4 Object Tracking

In the previous section, we have shown that for a given detection a at time t, we can recover its bounding box $\mathcal{B}^a_{t-1}$ and $\mathcal{B}^a_t$ in respectively the previous frame F_{t-1} and the current frame F_t. As we aimed to test our object detector as a tracker, we adopted a simple scheme to match the detections on the pair of frames (F_{t-2}, F_{t-1}) of the previous time step with detections of the current time step (F_{t-1}, F_t). We say that two detections a and b from respectively time step $t-1$ and t matches if:

$$\begin{aligned} \text{IoU}(\mathcal{B}^a_{t-1}, \mathcal{B}^b_{t-1}) &< \epsilon \\ \text{dist}(\mathcal{B}^a_{t-1}, \mathcal{B}^b_{t-1}) &< \mu \end{aligned} \tag{3.4}$$

where IoU is the Intersection over Union (also called Jaccard index) over the area of the 2D bounding boxes, *dist* is the euclidean distance between the centre of the two boxes in the 3D

space, ϵ and μ are two thresholds. In our experiments, we choose $\epsilon = 0.5$ and $\mu = 2\,\text{m}$. If a detection at time step t can be matched to more than one detection at time $t-1$, we pick the detection with the highest overlap measured by the 2D IoU. We ignore any detection with a class score below 0.2 and any new detection with a score below 0.6. Using this scheme, we are able track the object frame-to-frame using our detector without relying on any additional tracker.

3.3 Network Training

We trained and evaluated our algorithm on the KITTI detection benchmark [8] which does not provide any ground truth for the previous frame F_{t-1}; however, we were able to annotate 4400 images out of the 7400 available images using the tracklets provided in the raw dataset. When no ground truth is available for F_{t-1} for a given sample (either a new object in the sequence or missing ground truth mentionned above), we ignore the output of the loss functions related to the previous frame as well as the velocity loss; and instead only learn the outputs associated with the current frame F_t.

We use the two-phase training of MS-CNN as described in [43]. In the first phase, we train the region proposal network (shown in Figure 3.2) to generate region proposals $\mathcal{P}_t$ and $\mathcal{P}_{t-1}$ on respectively the current and the previous images using a multitask loss.

In the second phase, our entire network, comprising the architecture of [43] and our 3D pose regression extension, is fine-tuned end-to-end using a multi-task loss over 7 sets of network outputs: *class* and *quadrant classification* are learned via cross entropy loss while *bounding-box position*, *object centre*, *distance*, *orientation*, *velocity* are dependent on a mean-square loss. As a result, it would be time-consuming to manually tune the weights of the weighted sum of losses in a multi-task loss, therefore we use the methodology of Kendall *et al.* [155] to dynamically adjust the multi-task weights during training based on homoscedastic uncertainty without any use of manual hyperparameters.

For each phase, we trained the network using stochastic gradient descent with a batch size of 4 samples, hence 8 frames per batch and a learning rate of 1×10^{-4} for 25000 iterations. The remaining parameters of MS-CNN as set as described in [43].

3.4 Evaluation

We evaluate our approach on the KITTI object detection and tracking benchmark [8]. In addition to the metrics used in the benchmark, we provide our own metrics for the extension to 3D pose and velocity estimation.

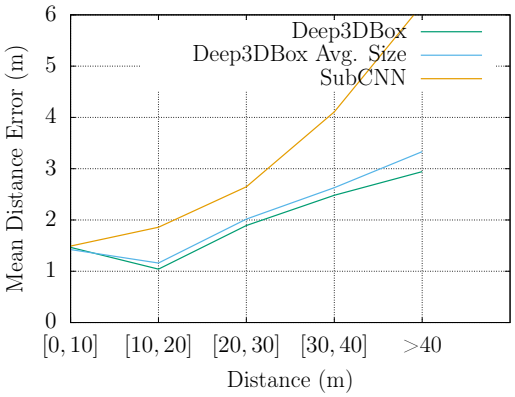
3.4.1 Distance Evaluation

We compare our results (shown in Figures 3.9 and 3.10) to the current state-of-the-art monocular 3D object detection method Deep3DBox [4] with both methods trained on the training/validation split of [181]. The Figure 3.4a shows the original results of Deep3DBox [4] compared to SubCNN [5]. Those results were generated on a common subset of ground truth objects which were successfully detected by both methods. As shown in Figures 3.4b and 3.4c, our method has a 17% decrease of average distance error compared to Deep3DBox and a 10% increase in 3D IoU performances for the common set of detections between the two methods.

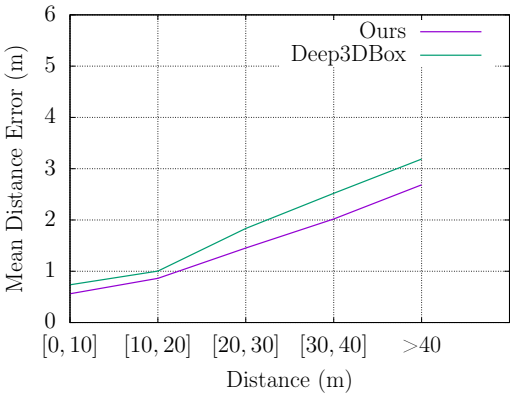
As shown in Figure 3.5, we were also able to learn the distance of more difficult objects such as pedestrians and cyclists which have very few samples in the dataset. The much higher error on pedestrians and cyclists more than 30m away are due to the lack of training data and their small size.

3.4.2 3D Detection Benchmark

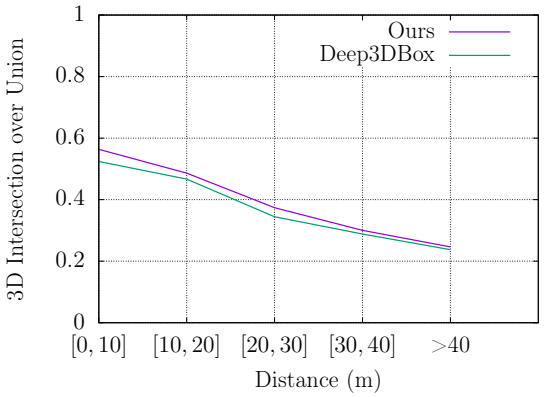
Following the work on Chen *et al.* [152] on multi-view 3D detection, this task was recently added to the KITTI benchmark. In this section, we finetune our network using the entire KITTI



(a) Comparison of Deep3DBox [4] and SubCNN [5] using the average distance error per distance bracket.

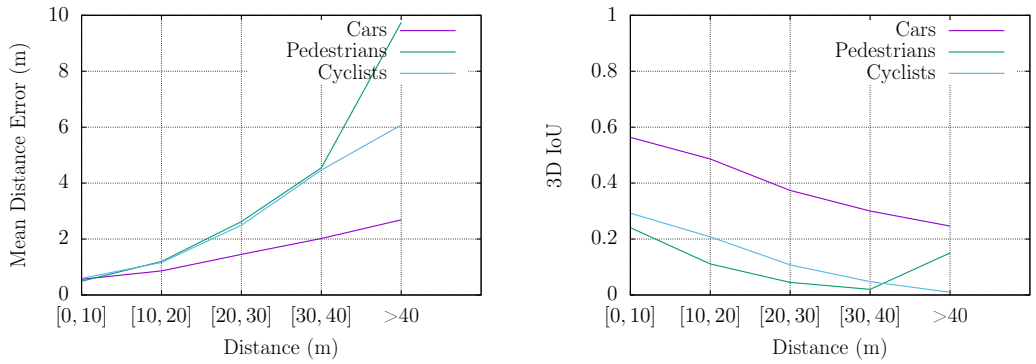


(b) Comparison of Deep3DBox [4] and our approach using the average distance error per distance bracket.



(c) Comparison of Deep3DBox [4] and our approach using the 3D IoU per distance bracket.

Figure 3.4: Pairwise comparison of our method, Deep3DBox [4] and SubCNN [5]. Each comparison is performed on the common subset of successful detections between the two approaches.



(a) Mean distance error per distance bracket.

(b) 3D IoU per distance bracket.

Figure 3.5: Statistics for all categories of the KITTI benchmark.

training set and present the results on the testing set. The 2D IoU matching between ground truth and detections was replaced with 3D IoU. Detections must score an IoU greater than 0.70 to be valid. Our performances shown in Table 3.1 are much lower than other methods published in the benchmark [152, 182, 183] because we rely solely on monocular imagery instead of multiview or LIDAR data. We achieved 2.62%, 1.30% and 1.25% for respectively easy, moderate and hard detections. An IoU above 0.70 requires a distance error below 2 m because cars are on average 4 m long. On average, we achieved a distance error of 1.5 m and a 3D IoU of 0.40.

Our score on the pedestrian and cyclist 3D detection benchmark are shown in Table 3.1. Due to their small size, as shown in Figure 3.5b, it is much more difficult to have a high overlap for pedestrian and cyclist than cars. The KITTI benchmark requires IoU$\geq$ 0.5 for cyclist and pedestrians. This means that the actual position of the object must be pinpointed down to a few decimetres.

As shown in Figure 3.6, the performances obtained with a 3D IoU are heavily dependent on the overlap threshold used in the evaluation. Decreasing the required threshold leads to a dramatic improvement of the Average Precision. In contrast, the performances of the 2D detections are

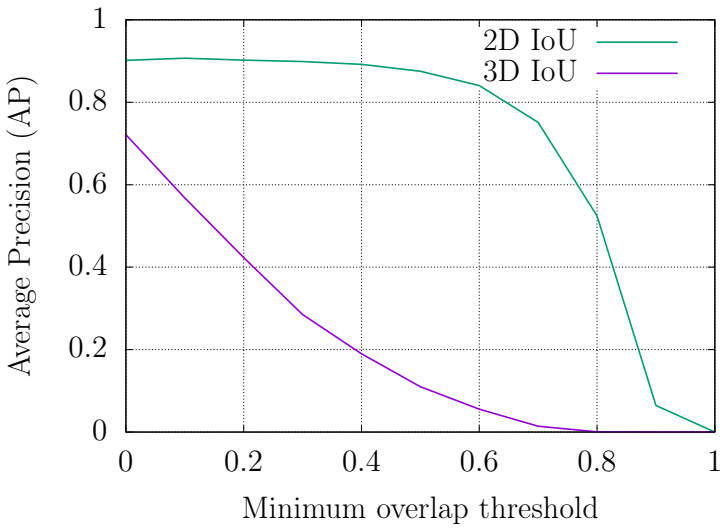


Figure 3.6: Average precision as the required overlap threshold is changed (vehicles, validation dataset)

not as sensitive to the threshold. This difference can be explained by the 3D IoU accumulating the error along both the 2D position and distance dimensions, whereas the 2D IoU excludes the distance error. This sensitivity shows that the low performances on the KITTI benchmark can be attributed to the quality of the 3D bounding boxes, and particularly the difficulty of measuring the distance of each vehicle in monocular imagery.

While LiDAR can estimate the distance of objects within a few centimetres accuracy, monocular approaches have a much higher error margin as shown in Figure 3.5. As the KITTI 3D detection benchmark use 3D IoU to match ground truths and detections, the monocular approaches are particularly penalised due to the lack of high accuracy, especially for small size objects. In contrast, newer benchmarks such as the nuScenes benchmark [7], which we use in Chapter 5, replaces the 3D IoU with euclidean distance between the ground truths and detections.

Method	Modality	Car			Pedestrian			Cyclist		
		Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard
MV3D [152]	multiview	62.35%	71.09%	55.12%	-	-	-	-	-	-
MV3D [152]	lidar	52.73%	66.77%	51.31%	-	-	-	-	-	-
CSoR [182]	multiview	6.79%	6.76%	6.14%	-	-	-	-	-	-
RefinedMPL [184]	monocular	11.14%	18.09%	8.94%	7.18%	11.14%	5.84%	1.82%	3.23%	1.77%
AM3D [185]	monocular	10.74%	16.50%	9.52%	-	-	-	-	-	-
M3D-RPN [186]	monocular	9.71%	14.76%	7.42%	3.48%	4.92%	2.94%	0.65%	0.94%	0.47%
Ours	**monocular**	**1.30%**	**2.62%**	**1.25%**	**0.45%**	**0.52%**	**0.65%**	**1.14%**	**1.14%**	**1.14%**

(a) 3D object evaluation

Method	Modality	Car			Pedestrian			Cyclist		
		Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard
MV3D [152]	lidar	77.00%	85.82%	68.94 %	-	-	-	-	-	-
MV3D [152]	multiview	76.90%	86.02%	68.49%	-	-	-	-	-	-
3D FCN [187]	lidar	76.90%	86.02%	68.49%	-	-	-	-	-	-
CSoR [182]	multiview	18.69%	23.94%	16.30%	-	-	-	-	-	-
RefinedMPL [184]	monocular	17.60%	28.08%	13.95%	7.92%	13.09%	7.25%	2.42%	4.23%	2.14%
AM3D [185]	monocular	17.32%	25.03%	14.91%	-	-	-	-	-	-
Ours	**monocular**	**3.65%**	**5.71%**	**3.50%**	**0.67%**	**0.65%**	**0.65%**	**2.27%**	**2.27%**	**2.27%**
VeloFCN [183]	lidar	0.33%	0.15%	0.47%	-	-	-	-	-	-

(b) Bird's eye view evaluation

Table 3.1: Statistical results on the KITTI 3D detection benchmark [8] using mAP. Higher is better.

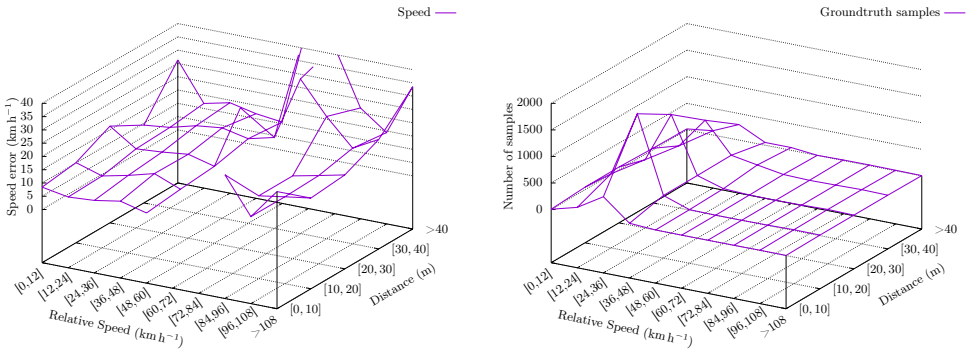
3.4.3 Velocity Evaluation

Figures 3.7a and 3.7c show the average relative speed error for detected vehicles. Each detection is grouped by distance and relative speed from the egovehicle and the average error is calculated for each of those groups. We can see that on average, the error is around $10\,\mathrm{km\,h^{-1}}$ (6.2 mph) for relative speeds below $60\,\mathrm{km\,h^{-1}}$ (37 mph). The error slightly increases with the distance. In two places, the error becomes particularly high: for vehicles at low speed ($<12\,\mathrm{km\,h^{-1}}$, 7.4 mph) in the regions of [10 m,20 m] and above 40 m in distance; and for vehicles at high speeds above $60\,\mathrm{km\,h^{-1}}$ (37 mph). This can be attributed to the lack of ground truth data both for training and evaluation in the KITTI dataset for such higher speed. This is particularly visible in the distribution of the ground truth instances as shown in Figure 3.7b & 3.7d. In particular, there is no ground truth data for speeds in the range [$60\,\mathrm{km\,h^{-1}}$,$96\,\mathrm{km\,h^{-1}}$] ([37 mph,60 mph]) at distances below 30 m, thus leaving a gap in Figure 3.7a & 3.7c. The KITTI data has been mostly captured at low speed in a urban environment with very few examples of highway driving. The distribution of speeds is also influenced by the speed limits and the layout of the roads. At high speed, median strips are typically used to separate opposing lanes and distance opposing vehicles from each other. Besides, on a highway most vehicles drive at very similar speeds and the actual relative speed is quite small. Therefore, measurements of high speeds are not particularly good but are not particularly relevant either in this context.

The relative speed can also be calculated as the difference between the position of the object in the current (p_t) and previous frames ($p_{t-\Delta t}$) divided by the time difference Δt as defined by:

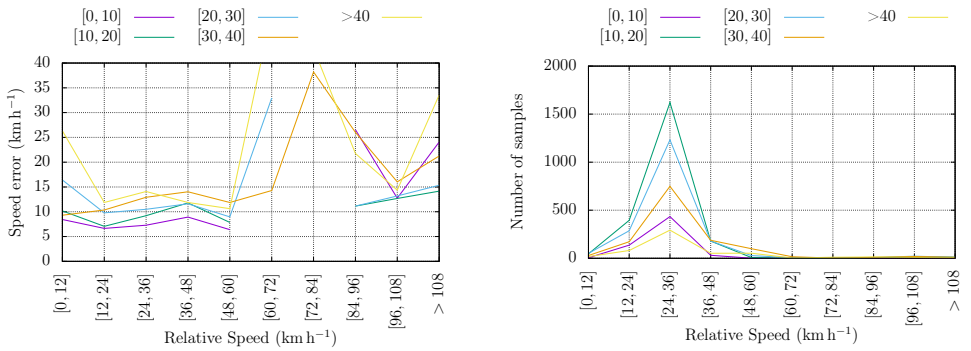
$$v = \frac{p_t - p_{t-\Delta t}}{\Delta t} \tag{3.5}$$

The results are shown in Figure 3.8a & 3.8b. On average, the error generated by this method (Figure 3.8a & 3.8b) is about twice the error of directly learned speeds (Figure 3.7a & 3.7c).



(a) Average speed error per relative speed and distance bucket.

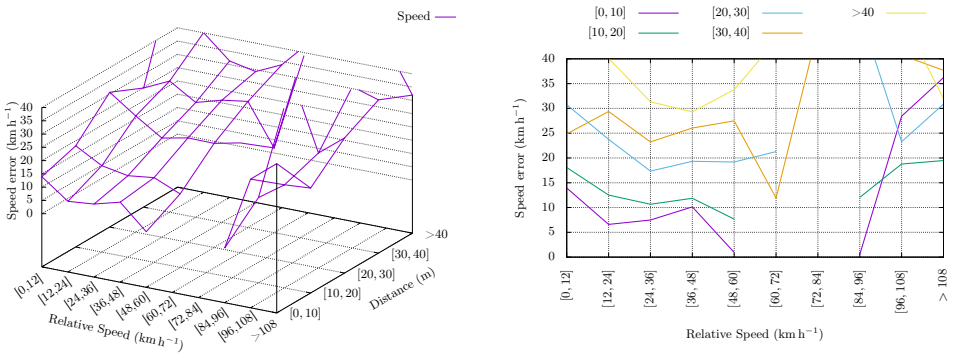
(b) Histogram of the number of ground truth samples per relative speed and distance bucket.



(c) Average speed error per relative speed at different distances (in m).

(d) Histogram of the number of ground truth samples per relative speed at different distances (in m).

Figure 3.7: Speed measurement error and number of ground truth samples per relative speed and distance bucket. The gaps in Figure 3.7a & 3.7c means that there is no ground truth samples in those buckets.



(a) Average speed error per relative speed and distance bucket.

(b) Average speed error per relative speed at different distances (in m).

Figure 3.8: Speed calculated as the difference of the previous and current relative vehicle position. Speed measurement error and number of ground truth samples per relative speed and distance bucket. The gaps in Figure 3.8a & 3.8b means that there is no ground truth samples in those buckets.

Besides, the results are much more sensitive to distance. This illustrates two aspects. The first one is that at higher distances, it is easier to calculate the relative motion than absolute positions. The second one is that we are accumulating the uncertainty surrounding the measurements in the current and previous frames. Therefore, we can conclude that for the purpose of studying relative positions such as speed measurements, it is more efficient to directly regress them than calculating them from other outputs. This is enabled by the two streams neural network architecture, which allows our approach to directly learn the velocity of an object between two frames.

3.4.4 Tracking Benchmark

While we trained our network using the training dataset provided by the KITTI detection benchmark (Figure 3.11), we also show our results on the KITTI tracking benchmark [8]. We evaluate our work using the MOTA and MOTP metrics defined by Bernardin and Stiefelhagen [101]. On the testing dataset, we achieved MOTA of 66.96% and MOTP of 79.51%. Those results are slightly below the state-of-the-art because we have compromised the quality of the neural net-

work by using a 1.5x upsampling as described in Section 3.2.1 instead of 2x upsampling of the input used in the final results of MS-CNN [43]. This compromise was necessary to fit the model within the memory requirements (12Gb) of available GPU hardware.

3.5 Summary

We have shown how to recover the 3D pose, size and velocity of objects from monocular video imagery and use this information to track objects frame-to-frame without the need of an additional tracker, using a siamese neural network [37] based on MS-CNN [43] which is able to process a pair of frames at a time. Our pose estimation approach achieves state of the art results compared to Mousavian *et al*. [4] and Xiang *et al*. [5]. Besides the approach can further be used to enrich the input data of an existing tracking framework such as the online tracker [107] with full 3D pose, size and velocity of the detection. As we are able to process a pair of frames through our two streams architecture, we are able to directly regress the velocity as an output of the network rather than calculating from the previous and current 3D locations. We have shown that this leads to more accurate measurements of the velocity of objects. By its frame-to-frame nature with no memory, our work is not able to track objects through occlusions and is not robust to false negatives. Chapter 5 attempts to address this issue by transforming the detection CNN into a recurrent network to introduce memory inside the network.

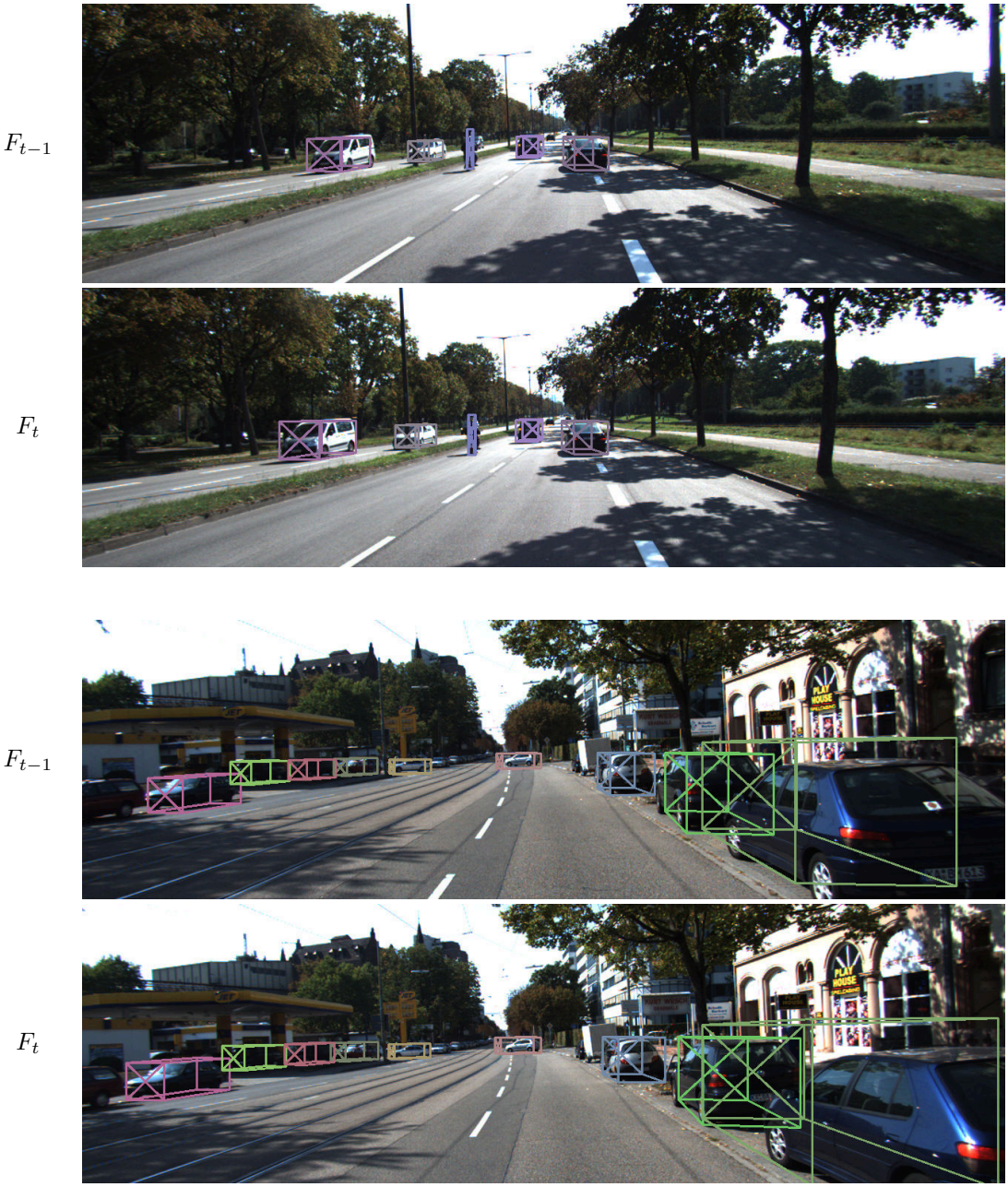


Figure 3.9: Examples of detection results. Top row shows the results on the previous frame and bottom row shows the result on the current frame.

Figure 3.10: Examples of detection results. Top row shows the results on the previous frame and bottom row shows the result on the current frame. (cont.)

Figure 3.11: Tracking results. Each images was captured at 4 frames interval.

Figure 3.11: Tracking results. Each images was captured at 4 frames interval. (cont.)

Figure 3.12: Tracking results. Each images was captured at 4 frames interval. (cont.)

Figure 3.12: Tracking results. Each images was captured at 4 frames interval. (cont.)

Figure 3.13: Tracking results. Each images was captured at 4 frames interval. (cont.)

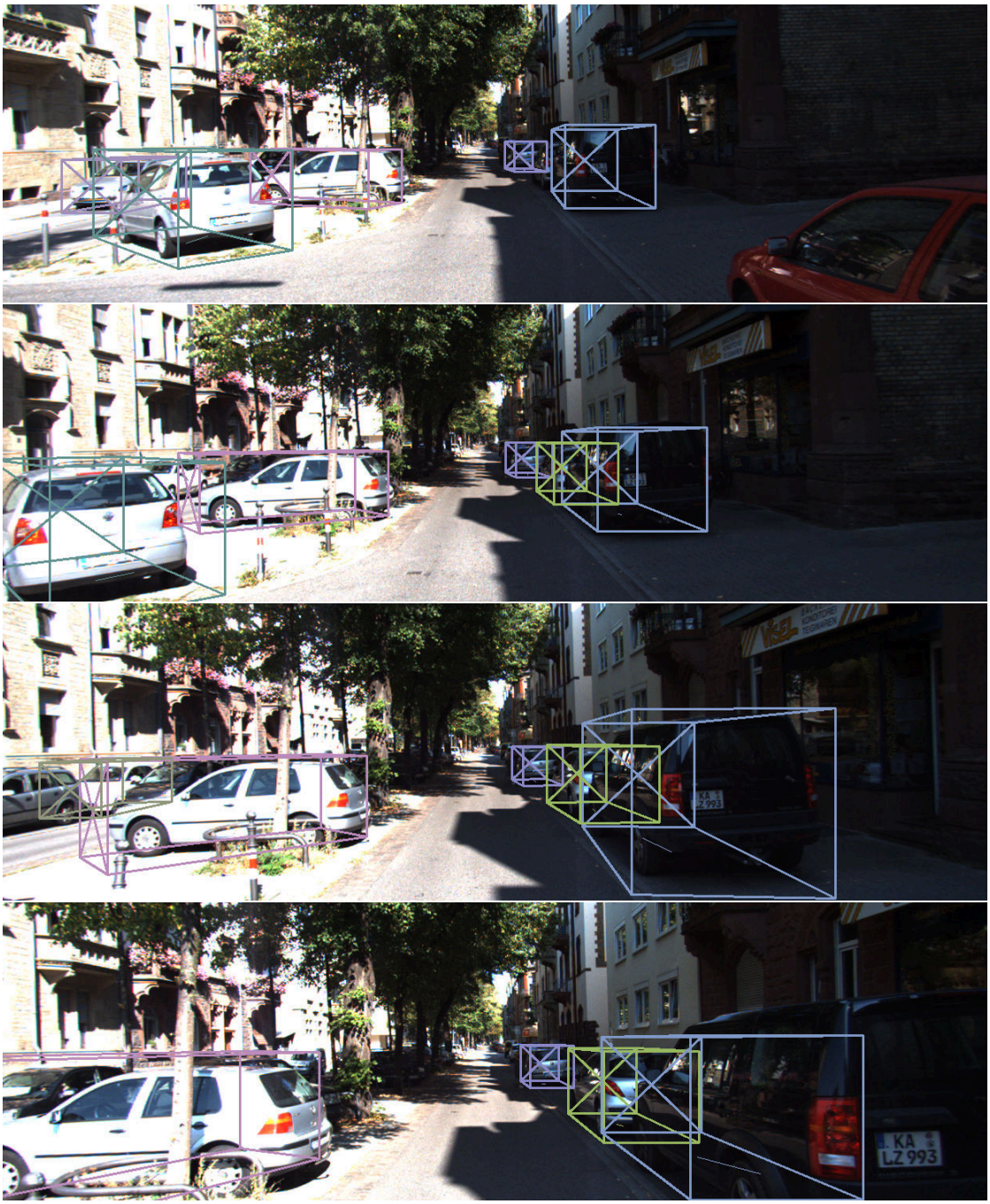


Figure 3.13: Tracking results. Each images was captured at 4 frames interval. (cont.)

Chapter 4

Adapting 3D Object Detection and Monocular Depth Estimation to 360° Panoramic Imagery

4.1 Introduction

Recent automotive computer vision research (object detection [34,44], segmentation [188], stereo vision [142,189], monocular depth estimation [38,145,147]) and indeed our own work in Chapter 3 have focused almost exclusively on the processing of forward-facing rectified rectilinear vehicle mounted cameras. Indeed, by sharp contrast to the abundance of common evaluation criteria and datasets for forward-facing camera imagery [1,8,51,190–193], there are no annotated evaluation dataset or frameworks for any of these tasks using panoramic camera.

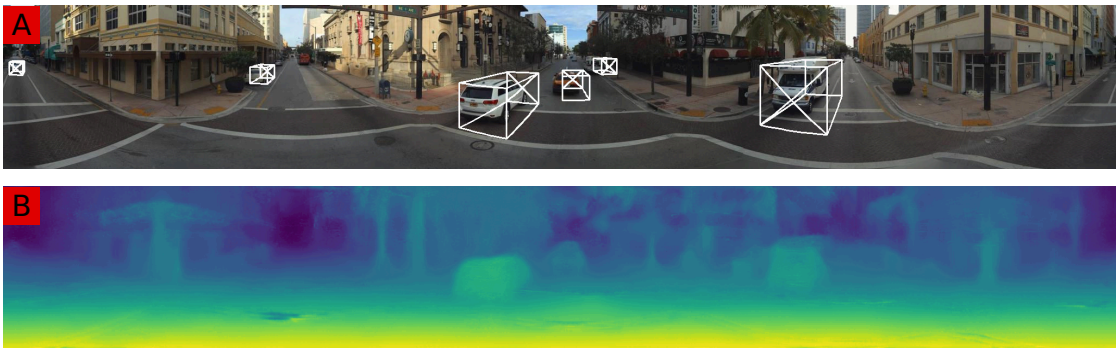


Figure 4.1: Our monocular panoramic image approach. A: 3D object detection. B: depth recovery.

However, varying levels of future vehicle autonomy will require full 360° situational awareness, akin to that of the human driver of today, in order to be able to function across complex and challenging driving environments. One popularly conceived idea of capturing this awareness is to use active sensing in the form of 360° LIDAR, however this is currently an expensive, low-resolution method which does not encompass the richness of visual information required for high fidelity semantic scene understanding. An alternative is to fuse the information from multiple cameras surrounding the vehicle [194] and such methods have been used to fuse between a forward-facing camera and LIDAR [152, 195]. However, here opportunities are lost to share visual information in early stages of the pipeline with further computational redundancy due to overlapping fields of view. Alternatively, the imagery from a multiview setup can be stitched into a 360° panorama [196]. A roof mounted on-vehicle panoramic camera offers superior angular resolution compared to any LIDAR, is 1-2 orders of magnitude lower cost and provides rich scene colour and texture information that enables full semantic scene understanding [197].

Panoramic images are typically represented using an equirectangular projection (Figure 4.1A); in contrast a conventional camera uses a rectilinear projection. In this projection, the image-space coordinates are proportional to latitude and longitude of observed points rather than the usual projection onto a focal plane. As shown in Figure 4.1A, straight 3D lines are represented as

curves and the panoramic representation of objects exhibits spherical distortion as an intrinsic property of the equirectangular projection. This property has limited the success of applying existing contemporary object detection approaches to panoramic images.

Recent work on panoramic images has largely focused on indoor scene understanding [150, 151], panoramic to rectilinear video conversion [81, 82, 198] and dual camera 360° stereo depth recovery [199, 200]. However, no work to date has explicitly tackled contemporary automotive sensing problems.

By contrast, we present an approach to adapt existing deep architectures, such as a CNN [38, 43] developed on rectilinear imagery to operate on equirectangular panoramic imagery. Due to the lack of explicit annotated panoramic automotive training datasets, we show how to reuse existing non-panoramic datasets such as KITTI [1, 8] using style and projection transformations, to facilitate the cross-domain retraining of contemporary algorithms for panoramic imagery. We apply this technique on panoramic imagery to estimate the dense monocular depth (see example in Figure 4.1B) and to recover the full 3D pose of vehicles (Figure 4.1B) based on our approach presented in Chapter 3. Additionally, our work provides the first performance benchmark for the use of these techniques on 360° panoramic imagery acting as a key driver for future research on this topic. Our technique is evaluated qualitatively on crowd-sourced 360° panoramic images from Mapillary [171] and quantitatively using ground truth from the CARLA [39] high fidelity automotive environment simulator. For future comparison our code, models and evaluation data is publicly available[1].

4.1.1 Proposed Contributions

Overall the main contributions, against the state of the art [1, 4, 8, 38, 39, 43], presented in this work are:

[1] https://gdlg.github.io/panoramic

- a novel approach to convert deep network architectures [38, 43] operating on rectilinear images for equirectangular panoramic images based on style and projection transformations;
- a novel approach to reuse and adapt existing datasets [1, 8] in order to train models for panoramic imagery;
- the subsequent application of these approaches for monocular 3D object detection using an adaptation of our work in Chapter 3, additionally operable on conventional imagery without modification;
- further application of these techniques to monocular depth recovery using an adaptation of the rectilinear imagery approach of Godard *et al.* [38];
- provision of the first performance benchmark based on a new synthetic evaluation dataset (based on CARLA [39]) for this new challenging task of automotive panoramic imagery depth recovery and object detection evaluation.

4.2 Approach

We first describe the mathematical projections underlining rectilinear and equirectangular projections and the relationship between the two required to enable our approach within panoramic imagery (Section 4.2.1). Subsequently, we describe the dataset adaptation (Section 4.2.2), its application to monocular 3D pose recovery (Section 4.2.3) and depth estimation (Section 4.2.4) and finally the architectural modifications required for inference within panoramic imagery (Section 4.2.5).

4.2.1 Rectilinear and Equirectangular Projections

Projection using a classical rectified rectilinear camera is typically defined in terms of its camera matrix P. Given the Cartesian coordinates (x, y, z) of a 3D scene point in camera space, its

projection (u_{lin}, v_{lin}) is defined as:

$$\begin{bmatrix} u_{lin} \\ v_{lin} \end{bmatrix} = \left\lfloor P \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right\rfloor \tag{4.1}$$

where $\lfloor \cdot \rfloor$ denotes the homogeneous normalization of the vector by its last component. Assuming the pinhole camera model shown in Figure 4.2, the camera matrix P is conventionally defined as:

$$P = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

where f and (c_x, c_y) are respectively the focal length and the principal point of the camera [79, Chapter 1].

The rectilinear projection as defined in Equation 4.1 is advantageous because the camera matrix P can be combined with further image and object space transformations into a single linear transformation followed by an homogeneous normalization. However, this transformation can also be written as:

$$\begin{bmatrix} u_{lin} \\ v_{lin} \end{bmatrix} = P \cdot \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \tag{4.3}$$

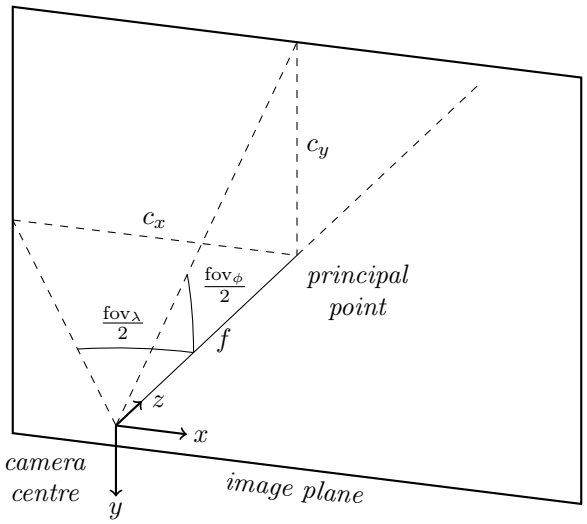


Figure 4.2: Pinhole camera model of principal point (c_x, c_y) and focal length f. fov_x and fov_y are respectively the horizontal and vertical field of view.

This formulation (Equation 4.3) is convenient because the image-space coordinates are expressed in terms of the ratio x/z and y/z which are the same regardless of the distance from the 3D scene point to the camera.

In contrast, the equirectangular projection is defined in terms of the longitude and latitude of the point using the coordinate system shown in Figure 4.3. The longitude and latitude, respectively (λ, ϕ), are defined as:

$$\lambda = \arctan\, x/z \tag{4.4}$$

$$\phi = \arcsin\, y/r \quad \text{where } r = (x^2 + y^2 + z^2)^{\frac{1}{2}} \tag{4.5}$$

The latitude definition in Equation 4.5 can be conveniently rewritten in terms of the ratios x/z

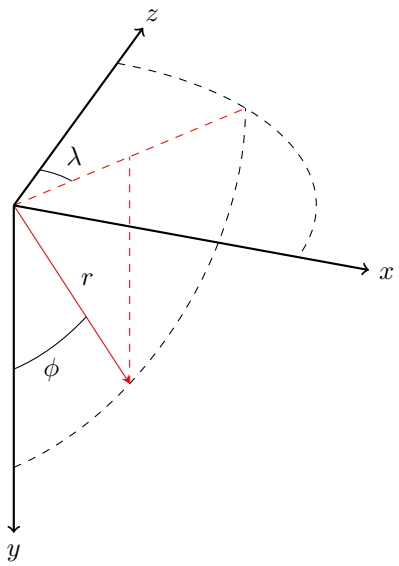


Figure 4.3: The spherical coordinate space using longitude, latitude and radius: (λ, ϕ, r).

and y/z as in Equation 4.3 for rectilinear projections:

$$\phi = \arcsin \frac{y/z}{r} \quad \text{where } r = (x/z^2 + y/z^2 + 1^2)^{\frac{1}{2}} \tag{4.6}$$

For the sake of simplicity, this computation of the latitude and longitude from the Cartesian coordinates can be represented as a function Γ:

$$\begin{bmatrix} \lambda \\ \phi \end{bmatrix} = \Gamma \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} \right) = \Gamma \left(\begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \right) \tag{4.7}$$

Finally, we define an image transformation matrix T_{equi} which transforms the longitude and

latitude to image space coordinates (u_{equi}, v_{equi}):

$$\begin{bmatrix} u_{equi} \\ v_{equi} \\ 1 \end{bmatrix} = T_{equi} \cdot \begin{bmatrix} \lambda \\ \phi \\ 1 \end{bmatrix} = T_{equi} \cdot \Gamma \left(\begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \right) \tag{4.8}$$

The matrix T_{equi} can be defined as:

$$T_{equi} = \begin{bmatrix} \alpha & 0 & c_\lambda \\ 0 & \alpha & c_\phi \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

where α is an angular resolution parameter akin to the focal length. Like the focal length, it can be defined in terms of the field of view:

$$\alpha = \text{fov}_\lambda / w = \text{fov}_\phi / h \tag{4.10}$$

where $\text{fov}_\lambda, \text{fov}_\phi, w, h$ are respectively the image horizontal FoV, vertical FoV, width and height. In contrast to rectilinear imagery, where the focal length is difficult to determine without any kind of camera calibration, the equirectangular imagery, commonly generated by panoramic cameras from the raw dual-fisheye pair, can be readily used without any prior calibration because the angular resolution $\alpha = 2\pi/w$ depends only on the image width. Therefore, approaches that would require some knowledge of the camera intrinsics of rectilinear images (*e.g.* monocular depth estimation) can be readily used on any 360° panoramic image without any prior calibration.

By coupling the definitions of both the rectilinear and equirectangular projections in terms of the ratios x/z and y/z (Equation 4.3 & 4.8), we establish the relationship between the coordinates

in the rectilinear projection and equirectangular projection for the given matrices P and T_{equi}:

$$\begin{bmatrix} u_{equi} \\ v_{equi} \\ 1 \end{bmatrix} = T_{equi} \cdot \Gamma \left(P^{-1} \cdot \begin{bmatrix} u_{lin} \\ v_{lin} \\ 1 \end{bmatrix} \right) \tag{4.11}$$

This enables us to reproject an image from one projection to another, such as from the rectilinear image (Figure 4.4A) to an equirectangular image (Figure 4.4C) and vice versa — a key enabler for the application of our approach within panoramic imagery.

4.2.2 Dataset Adaptation

In our approach, the source domain is the KITTI [1,8] dataset of rectilinear images captured using a front-facing camera rig (1242×375 image resolution; 82.5° horizontal FoV and 29.7° vertical FoV); while our target domain consist of 30,000 images from the Mapillary [171] crowd-sourced street-level imagery (2048×300 image resolution; 360° × 52.7° FoV). These latter images are cropped vertically from 180° down to 52.7° which is more suitable for automotive domain applications. This reduced panorama has an angular coverage 7.7 times larger than our source KITTI imagery. Due to the lack of annotated labels for our target domain, we adapt the source domain dataset to train deep CNN for panoramic imagery via a methodology based on projection and style transformations.

Due to dataset bias [160], training on the original source domain is unlikely to perform well on the target domain. Furthermore, our target is relatively low resolution and has numerous compression artefacts not present in the source domain, which are, however, present in our 360° target domain due to the practicality of 360° image transmission and storage. To improve generalization to the target domain, we transform the source domain to look similar to imagery from our target domain via a two-step process.

Figure 4.4: Output of each step of the adaptation of an image from the KITTI dataset: A: No tranformation, B: Style transfer, C: Projection transfer, D: Style and projection

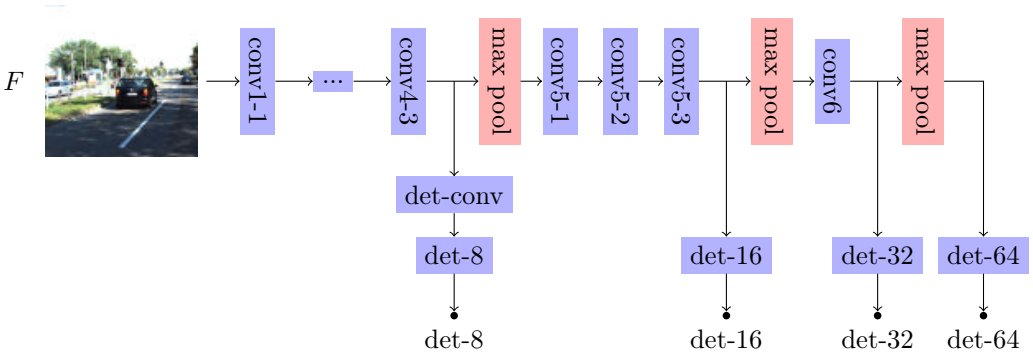


Figure 4.5: RPN subnetwork. Convolutional layers are shown in blue; pooling and concatenation layers in red. The layers *conv1-1* up to *conv6* are pre-trained VGG16 [3] layers.

The first step transfers the style of our target domain (reprojected as rectilinear images) onto each image from the source domain (Figure 4.4A); resulting images are shown in Figure 4.4B. We use the work of Zhu *et al*. on CycleGAN [2] to learn a transformation back and forth between the two unpaired domains. Subsequently, this transformation model is used to transfer the style of our target domain onto all the images from our source domain. Despite the style transfer, the actual geometry of the scene is preserved. In essence, the style transfer introduces a tone mapping and imitates compression artifacts present in most panoramic images while preserving the actual geometry. Without the use of style transfer, the weights are biased toward high-quality imagery and perform poorly on low-quality images.

The second step reprojects the style-transferred images (Figure 4.4B) and annotations from the source domain rectilinear projection to an equirectangular projection (Figure 4.4D). The transformed images represent small subregions (FoV: 82.5° ×29.7°) of a larger panorama. While this set of transformed images covers only a small 82.5° horizontal FoV, we find that they are sufficient to train a deep CNN that perform well on the full 360° FoV.

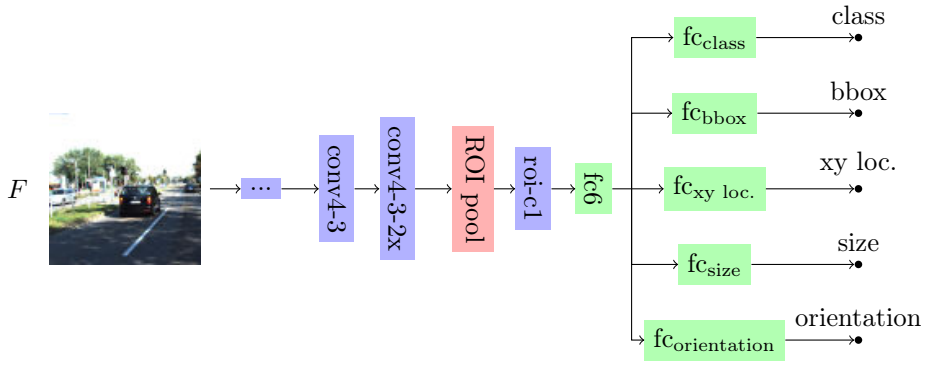


Figure 4.6: Detection network. Feature maps from the ROI pooling layer fed through a deconvolution layer (*roi-c1*) and a fully-connected layer (*fc6*). Each output is predicted using one further separate fully-connected layer.

4.2.3 3D Object Detection

For 3D detection, we leverage the same network by Cai *et al.* [43] which we used in Chapter 3, including our 3D pose regression extension; however, without loss of generality, we leave out our frame-to-frame temporal extensions as shown in Figure 4.5 (RPN) and Figure 4.6 (detection network), because the network is evaluated on a dataset of still panoramic imagery rather than video sequences in Section 4.3. Uniquely, our extended network can be used on either rectilinear or equirectangular imagery without any changes to the network itself, instead only requiring a change to the interpretation of the output for subsequent rectilinear or equirectangular imagery use.

As per Section 3.2.2, we regress the object disparity $d_{lin} = \frac{r}{f}$ which is the inverse of the distance r multiplied by the focal length f. For equirectangular imagery, we use a similar definition $d_{equi} = \frac{r}{\alpha}$ substituting the angular resolution for the focal length. Using a fully-connected layer connected to the last common layer defined in [43], we learn coefficients a, b such that the disparity d can be expressed as:

$$d = ah_{roi} + b \tag{4.12}$$

where h_{roi} is the height of the region proposal in pixels generated by the RPN (as defined by Cai *et al.* [43]). To simplify the computation, we also learn the 2D projection of the centre of the object onto the image (u, v) using another fully-connected layer. As a result, we can recover the actual 3D position (x, y, z) using:

$$[x, y, z]^T = r \cdot u[P^{-1} \cdot [u_{lin}, v_{lin}, 1]^T] \qquad \textit{for rectilinear images} \tag{4.13}$$

$$[x, y, z]^T = r \cdot u[\Gamma^{-1}(T_{equi}^{-1} \cdot [u_{equi}, v_{equi}, 1]^T)] \qquad \textit{for equirectangular images} \tag{4.14}$$

where $u[v] = \frac{v}{\|v\|}$ is the unit vector in the direction of v.

For network training of our model, we additionally use data augmentation including image cropping and resizing as defined by Cai *et al.* [43]. Any of those operations on the image must be accompanied by the corresponding transformation of the corresponding camera matrix P or T_{equi} in order to facilitate effective training.

We learn the remaining size and orientation parameters using the methodology described in Section 3.2.2. Therefore our entire network, comprising the architecture of [43] and our 3D pose regression extension, is fine-tuned end-to-end using a multi-task loss over 6 sets of network outputs: *class* and *quadrant classification* are learned via cross entropy loss while *bounding-box position*, *object centre*, *distance*, *orientation* are dependent on a mean-square loss.

4.2.4 Monocular Depth Recovery

We rely on the approach of Godard *et al.* [38] which was originally trained and tested on the rectilinear stereo imagery of the KITTI dataset [8]. We reuse the same architecture and retrain it on our domain-adapted KITTI dataset constructed using the methodology of Section 4.2.2.

Following the original work [38], the loss function is based on a left-right consistency check between a pair of stereo images. In our new dataset, both stereo images have been warped to an equirectangular projection as well as depth smoothness constraints. While Godard *et al.* uses the stereo disparity $d_{stereo} = \frac{fB}{zw}$ where f is the focal length, B the stereo baseline and w the width of the image, we replace the focal length with the angular resolution: $d_{equi} = \frac{\alpha B}{rw}$.

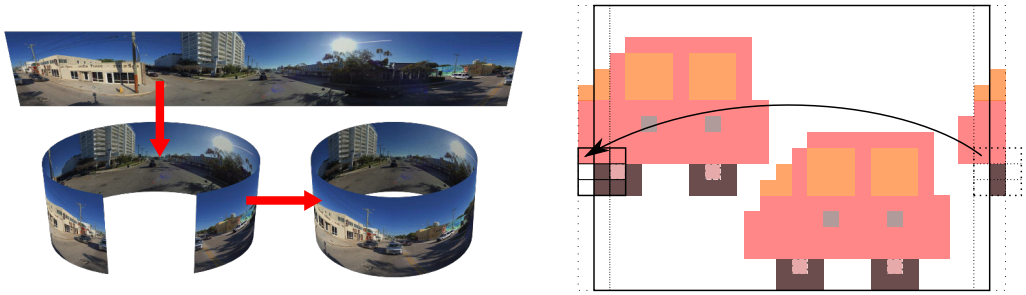
Given a point $p_l = (u_l, v_l)^T$, the corresponding point $p_r = (u_r, v_r)^T$ for a given disparity d can be calculated as:

$$p_r = T_{equi} \cdot \Gamma \left[u \left[\Gamma^{-1} (T_{equi}^{-1} \cdot p_l) \right] + \left[\frac{d_{equi} w}{\alpha}, 0, 0 \right]^T \right] \tag{4.15}$$

with definitions as per Section 4.2.1. The corresponding point p_r in Equation 4.15 is differentiable w.r.t. d_{equi} and is used for the left/right consistency check instead of the original formulation presented in [38]. This alternative formulation (Equation 4.2.1) explicitly takes into account that the epipolar lines in a conventional rectilinear stereo setup are transformed to epipolar curves within panoramic imagery, hence enabling the adaptation of monocular depth prediction [38] to this case.

4.2.5 360° Network Adaptation

While the trained network can be used as is [80, 81] without any further modification, objects overlapping the left and right extremities of the equirectangular image would be split into two objects; one on the left, and one on the right (as depicted in Figure 4.7a, bottom left). Moreover, information would not flow from one side of the image to the other side of the image — at least in the early feature detection layers. As a result, the deep architecture would '*see*' those objects as if heavily occluded. Therefore, it is more difficult to detect objects overlapping the image boundary leading to decreased overall detection accuracy and recall.



(a) A 360° equirectangular image can be folded over itself until the ends meet.

(b) A 3×3 convolution kernel, a column of padding copied from the other side is added at each extremity

Figure 4.7: Convolutions are computed seamlessly across horizontal image boundaries using our proposed padding approach.

A cropped equirectangular panorama can be folded into a 360° ring shown in Figure 4.7a by stitching the left and right edges together. A 2D convolution on this ring is equivalent to padding the left and right side of the equirectangular image with respective pixels from the right and left side as if the image was tiled (as illustrated on Figure 4.7b for 3×3 convolutions). This horizontal ring-padding is hence used on all convolutional layers instead of the conventional zero-padding to eliminate these otherwise undesirable boundary effects.

For 3D detection, our proposed approach based on Faster R-CNN [34] generates a sequence of detection proposals and subsequently pools a subregion around each proposal to further regress the final proposal location, class and 3D pose. To adapt this operation, instead of clamping subregion coordinates by the equirectangular image extremities, we instead wrap horizontally the coordinates of each pixel within the box:

$$u_{wrap} \equiv u \pmod{w} \tag{4.16}$$

where u is the horizontal coordinate of the pixel, u_{wrap} the wrapped horizontal coordinate within

the image and w the image width.

As a result of this approach, we are hence able to hide the image boundary, as a result, enabling a true 360° processing of the equirectangular imagery.

4.3 Evaluation

We evaluate our approach both qualitatively on panoramic images from the crowd-sourced street-level imagery of Mapillary [171] as well as quantitatively using synthetic data generated using the CARLA [39] automotive environment simulator. For future comparison, our code, models and evaluation data is publicly available[2].

4.3.1 Qualitative Evaluation

As discussed in Section 2.2.1, we qualitatively evaluate our method using 30,000 panoramic images (Miami, USA) from the crowd-sourced street-level imagery of Mapillary [171]. Figure 4.8 shows our depth recovery and 3D object detection results on a selection of images of representative scenes from the data. Ring-padding naturally enforces continuity across the right/left boundary; for instance, zero-padding can prevent detection of vehicles crossing the image boundary (Figure 4.9A) whereas ring-padding seamlessly detects such vehicle (Figure 4.9C). Similarly zero-padding introduces depth discontinuities on the boundary (Figure 4.9B) whereas ring-padding enforces depth continuity (Figure 4.9D).

The proposed approach is able to successfully estimate the 3D pose of vehicles and recover scene depth as shown in Figure 4.8. However, the approach fails on vehicles which are too close to the camera (as shown in Figure 4.8F and 4.8H), almost underneath the camera. Those vehicles are challenging to detect, because there are no vehicle with a similar pose relative to the egovehicle in

[2] https://gdlg.github.io/panoramic

the KITTI dataset. For vehicles within a couple of metres of the camera, the view of the vehicle is heavily influenced by the position of the camera on the egovehicle. In particular, the camera is placed higher on the roof of the vehicle in the Mapillary dataset than in the KITTI dataset. Secondly, such vehicle are viewed sideway (as shown in Figures 4.8 F, G, and H); however, such vehicles is usually seen on the sides of the egovehicle, driving in adjacent lanes rather than in front of the egovehicle, thus there are no example of such vehicle in the KITTI dataset. This issue is also present to some extent for depth estimation (as shown in Figure 4.8D) for parts of the scene on the side of the egovehicle. Following the conventions of the KITTI dataset, any vehicles less than 25 pixels in image height were ignored during training. Due to the lower resolutions of the panoramic images, an average-size vehicle (about $2m$ height) with an apparent height of 25 pixels in KITTI is approximately at a distance of $56.6m$, whereas the same vehicle in a panoramic image will stand at $26m$. As a result, the range of the algorithm is reduced even though this is not a fundamental limitation of the approach itself (*e.g.* in Figure 4.8A and 4.8H). Rather, we expect this maximum distance to be increased as the resolution of panoramic imagery is increased.

4.3.2 Quantitative Evaluation Methodology

Due to the lack of available annotated automotive panoramic imagery dataset, we evaluate our algorithm on synthetic data generated using the CARLA automotive environment simulator [39] adapted for panoramic imagery rendering using the same format as our qualitative dataset.

We extended the CARLA simulator [39] to output four different views of 90° horizontal FoV at right angle from each other, as well as the depth map for each of those views. Those views were stitch together into a single equirectangular image. We generated a validation set and testing set using two distinct scenes, such that there is no overlap between the validation and testing sets, both in terms of imagery and content. We automatically generated a list of vehicles in the scene;

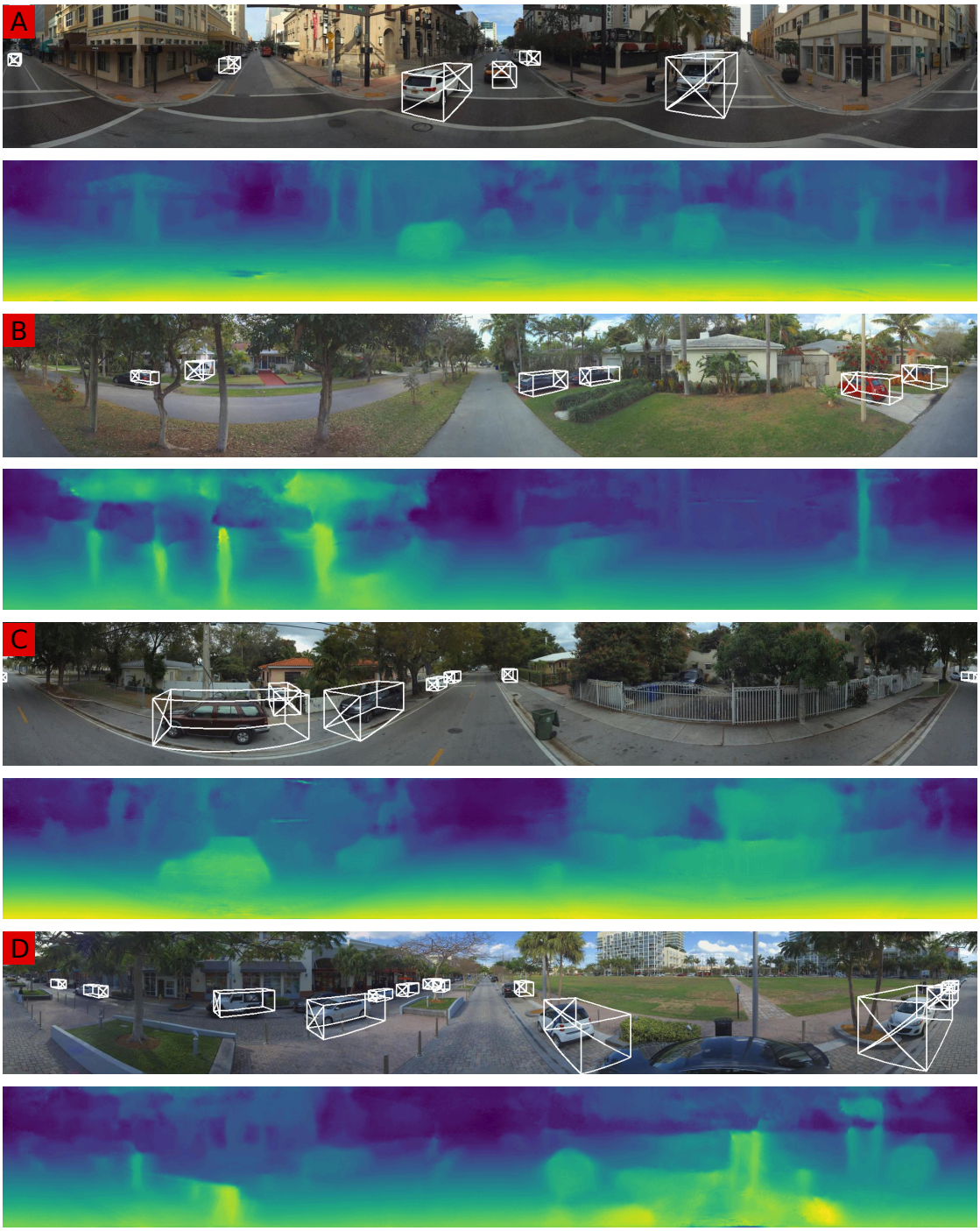


Figure 4.8: Monocular depth recovery and 3D object detection with our approach.

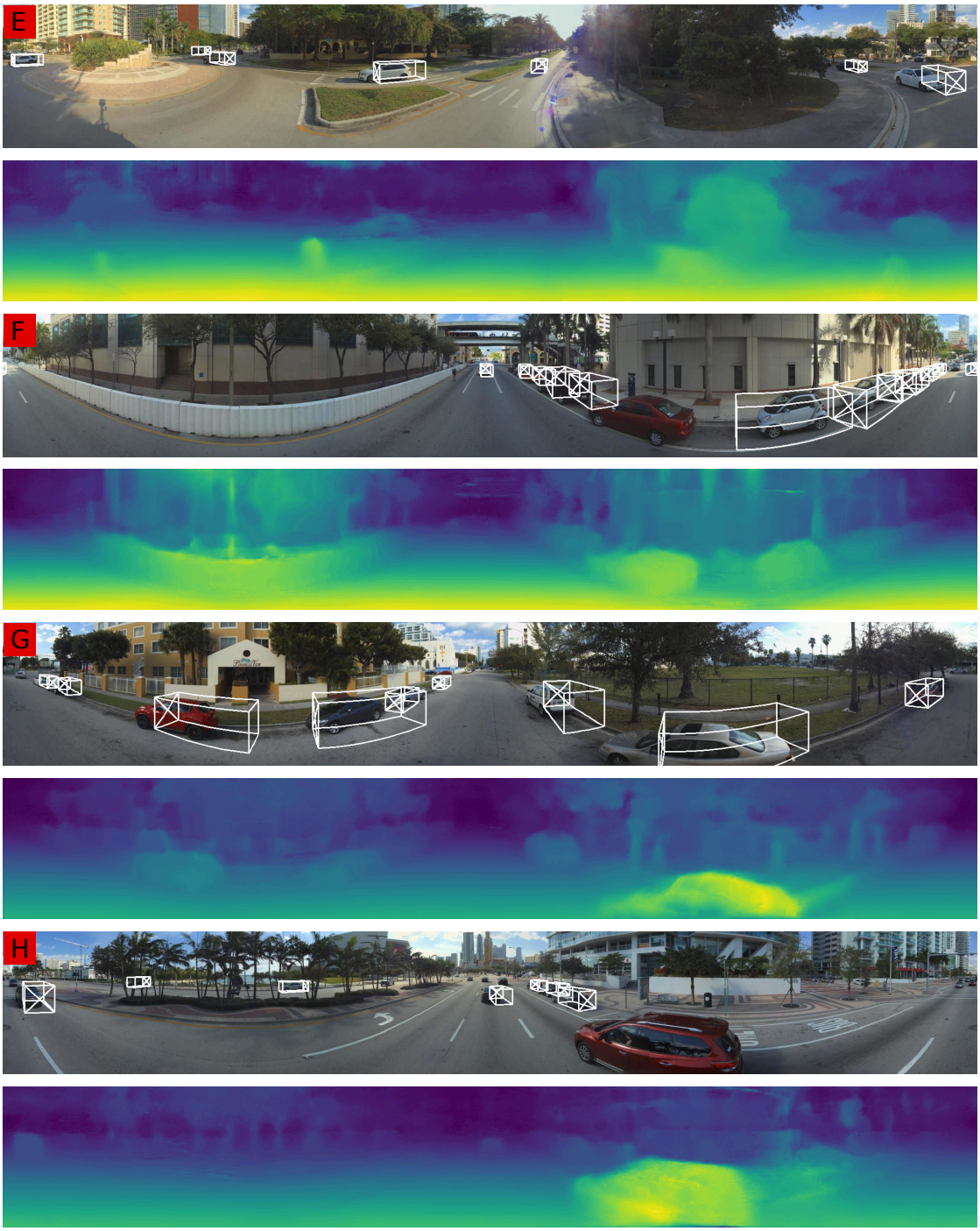


Figure 4.8: Monocular depth recovery and 3D object detection with our approach. (cont.)

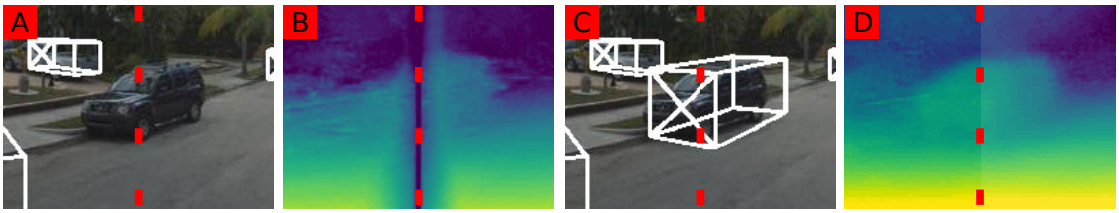


Figure 4.9: Right/left boundary effect. A,B: Zero-padding; C,D: Ring-padding.

however, as automated occlusion measurement is difficult in some cases such as transparency, wire meshes and fences; we choose to manually label the occlusion status of each object. Due to the lack of diversity of the content, our dataset based on CARLA is not suitable for training purposes, while it is suitable for cross-dataset testing. Qualitative results of our approach on this dataset are shown in Figure 4.10. Following KITTI conventions, we filtered out vehicles less than 25 pixels in height from our detection results.

Table 4.1 shows the mAP using an IoU of 0.5 across variations of our algorithm on 8,000 images. Overall, the projection transformation during training impairs the results by about 10% points. Our best results come from the combined style-transferred training dataset consisting of both Mapillary and CARLA (4% points increased compared to original) whilst training on the CARLA-adapted dataset alone impairs the performance by 2% points. This is due to the simplistic rendering and lack of variety of the synthetic dataset which impairs the style transfer. As a result, the CARLA-adapted dataset significantly boosts the accuracy for very low recall; however, it also reduces the recall ability of the network (Figure 4.12a). The model trained on the CARLA-adapted dataset achieves a mAP of 0.82 on our evaluation set of adapted images but only 0.35 on the actual CARLA dataset which shows that the style transfer is somewhat limited. The Figure 4.12b shows that the 3D IoU performance is similar with and without style transfer. This highlights that the style transfer primarily increases the quality of the detections but does not improve the 3D localisation of vehicles.

The monocular depth estimation results are shown in Table 4.1 for 200 images (for distances $<$

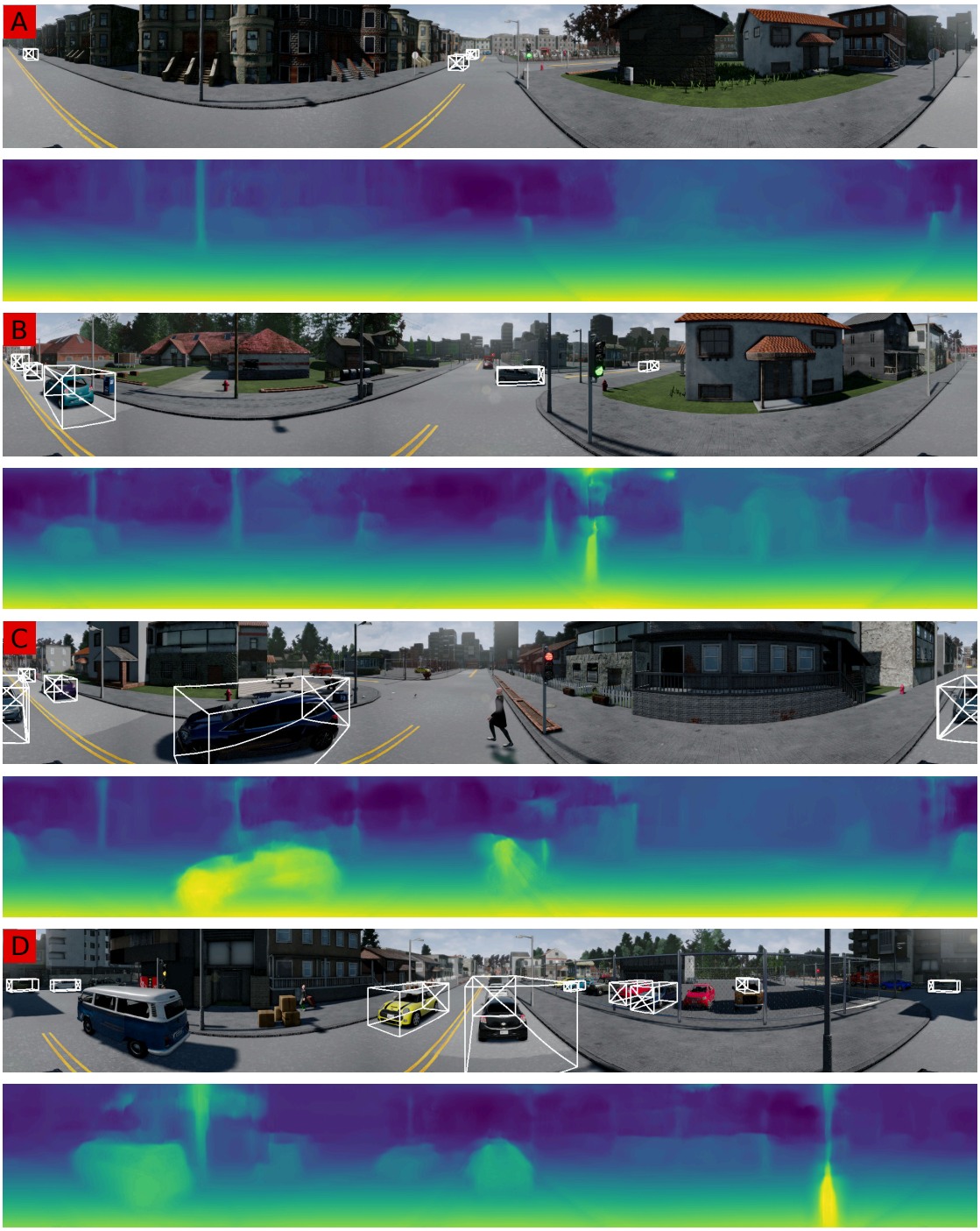


Figure 4.10: Monocular depth recovery and 3D object detection with our approach on our synthetic dataset based on CARLA.

$50m$). Similar to our detection result, using CARLA-adapted imagery impairs the performance. Using projection transformation, we see an increase of about 2.5% points in accuracy. Overall, those differences are smaller than those on object detection across the different transformations (Table 4.1).

Despite the lack of ground truth annotations in the Mapillary dataset, we can see that qualitatively the results are of higher quality on the Mapillary dataset than on the CARLA dataset. While the dataset adaptation described in Section 4.2.2 is suitable for real-world imagery such as the Mapillary dataset [171], the style-transfer with CycleGAN [2] struggles to find correspondences between the real-world KITTI dataset [8] and our synthetic virtual dataset as shown in Figure 4.11. Consequently, the KITTI images transferred to the synthetic dataset domain are still visually quite different from the target domain and have numerous defects as shown in Figure 4.11. In particular, it mismatches the road with grass (Figure 4.11A & B), the road with red rooftop shingles (Figure 4.11D), and introduces black artefacts (Figure 4.11A & C). Using a more robust style transfer approach or increasing the visual diversity and quality of the synthetic dataset would improve the domain adaptation quality and the overall results of our method. A consequence of this issue is that the training dataset combining images in the style of the Mapillary and CARLA dataset is able to outperform the CARLA training dataset on both detection precision and recall metrics (Figure 4.12a).

From our results, we can clearly see that we have identified a new and challenging problem within the automotive visual sensing space (Table 4.1) when compared to the rectilinear performance of contemporary benchmarks [1, 8].

4.4 Summary

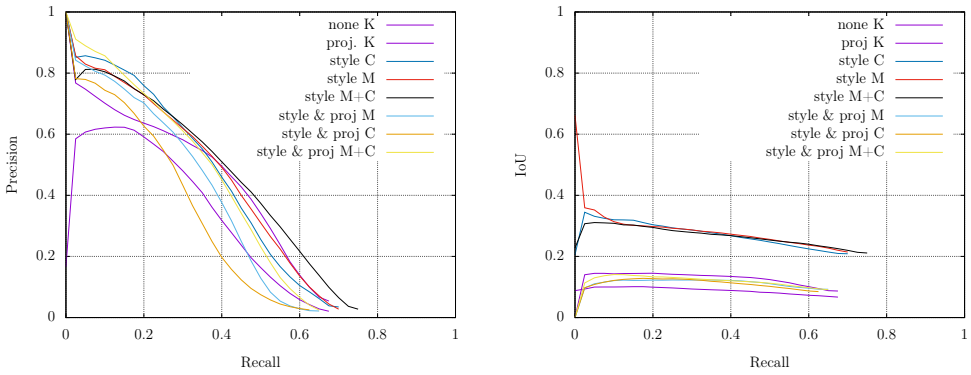
We have adapted our existing object detection approach described in Chapter 3, the depth estimation method of Mousavian *et al.* [4] and the KITTI dataset [1, 8], which are architectures and

Transformation	Dataset	Detection†	Depth Error Metrics‡				Depth Acc.†
		mAP	Abs. rel.	Sq. rel.	RMSE	RMSE log	$\delta < 1.25$
none	K	0.336	0.247	7.652	3.484	0.465	0.697
proj.	K	0.244	0.251	7.381	3.451	0.445	0.732
style	C	0.355	0.262	7.668	3.601	0.480	0.686
style	M	0.359	0.257	7.937	3.634	0.474	0.682
style	M+C	0.378	0.230	6.338	3.619	0.474	0.679
style & proj.	C	0.259	0.292	9.649	3.660	0.469	0.723
style & proj.	M	0.308	0.300	10.467	3.798	0.473	0.719
style & proj.	M+C	0.344	0.231	6.377	3.598	0.463	0.716

† Higher, better ‡ Lower, better

Table 4.1: Object detection (mAP) results; and depth recovery results using metrics defined by [9]. Training dataset: C: CARLA, M: Mapillary, K: KITTI

Figure 4.11: Example of failure cases of the style-transfer of KITTI images to the synthetic domain.



(a) Precision & Recall curve

(b) 3D Intersection over Union

Figure 4.12: Object detection results

training datasets proven on forward-facing rectilinear camera imagery, to perform on panoramic images. The approach is based on domain adaptation [2] using geometrical and style transforms, and novel updates to training loss to accommodate panoramic imagery. Our approach is able to recover the monocular depth and the full 3D pose of vehicles. We have also introduce a new padding technique to the convolutions in existing CNN to hide the discontinuity between the left and the right boundary of an equirectangular image.

Consequently, we have identified panoramic imagery has a new set of challenging problems in automotive visual sensing and provide the first performance benchmark for the use of these techniques on 360° panoramic imagery, with a supporting evaluation dataset of synthetic imagery generated using the CARLA automotive environment simulator [39], hence acting as a key driver for future research on this topic.

Chapter 5

Dense Object Detection and Tracking in Panoramic Imagery

5.1 Introduction

In Chapter 3, we introduced a frame-to-frame approach to object tracking using a siamese neural network which process simultaneously two frames together. In this chapter, we revisit object tracking in two ways: we use an end-to-end tracking CNN using recurrent connections and develop an object tracker for 360° panoramic video sequences in light of our work on 360° imagery presented in Chapter 4.

Object tracking is an essential component of the perception subsystem of any self-driving vehicle. Until the recent release of the nuScenes tracking benchmark [7, 102], automotive tracking datasets [1, 8] focused on 2D object tracking in forward-facing vehicle mounted cameras or LiDAR. Our approach presented in Chapter 4 being one of the first to extend object detection to

360° reasoning. In contrast, the nuScenes tracking benchmark [6] emphasises 360° surround tracking and 3D perception.

On the nuScenes benchmark, approaches for 360° tracking have focused exclusively on LIDAR data [201] with the exception of one camera-based method [202]. Following our approach presented in Chapter 4 on 360° monocular object estimation, we propose a new approach for 3D tracking within 360° equirectangular video sequences.

Object tracking is a broad field encompassing both single-object tracking-by-template and multi-object tracking-by-detection (MOTD) [101, 104, 105]. The later is predominantly used in self-driving applications. MOTD is often split into two parts, both conceptually and in the design of modern approaches [104] as discussed in Section 2.1.2. The first part, the detector, aims at detecting relevant objects in a single frame, which are typically identified by their bounding box [34]. For instance, such a detector might be configured and trained to detect vehicles and pedestrians. The second part, the tracker aims to form tracklets by matching across frames those detections, which belong to each individual objects [104].

The tracker matches each detection based on attributes such as detection position, size and appearance [104]. Unfortunately, it would be very difficult — if not impossible — to create an exhaustive list of all the desirable features for tracking purposes as explained in Section 2.1.2. The richer the set of attributes used, the more informed the tracker will be. As such, the recent literature has moved away from handcrafted features to instead use large feature maps as input to the tracker [124, 125]. Joint detection and tracking would simplify this problem because the information would no longer be explicitly passed between the two components.

End-to-end joint detection and tracking using neural networks is difficult for two main reasons. Firstly, in contrast to single-image processing, video processing is expensive at training time because the time complexity and memory required to process a batch of video sequences is linear

with respect to both width, height and the number of video frames. Secondly, the information in modern detection CNNs is spatially dense whereas the information in temporal trackers is sparse. This is because the tracker works with a handful of detections per frame instead of whole images or feature maps. Note that the video processing time is only a training issue; online inference is performed frame-by-frame and the frame processing time remains constant regardless of the number of frames. Leal-Taixé *et al.* [133], Feichtenhofer *et al.* [134] and our method proposed in Chapter 3 are end-to-end detection and tracking CNNs; however, they only process pairs of frames and lack temporal information — apart from the last frame. Voigtlaender [98] proposes a CNN based on 3D convolutions to integrate temporal information over 8 frames. In contrast, our proposed method uses recurrent connections with a hidden state.

We propose a novel approach with integrates tracking into a 3D detection CNN. As such, the tracking is performed directly on dense information rather than sparsely. This enables the tracker to reason holistically about the scene using attributes such as overall scene geometry, inter-object relations and occlusions, which are difficult to convey within the traditional framework. This tracker is able to detect, estimate the pose, and track objects in 360° panoramic video sequences. We evaluate our method on the recently released nuScenes Tracking Benchmark [7].

5.1.1 Proposed Contributions

Our key contributions are as follow:

- a new approach for joint detection and tracking which propagates feature maps through time at multiple scales. This approach is the first using 360° imagery and one of the first approaches [102, 202] based on camera only using the nuScenes Tracking Benchmark [6];
- a method to scale the training of such networks to video sequences of arbitrary length based on mixed precision training [203, 204] and reducing the memory requirements of back-propagation for video sequences;

- a neural network trained and tested on 360° panoramic video sequences generated from the nuScenes dataset [7] by stitching the imagery from the six cameras available in the dataset.

5.2 Method

Tracking an object is usually based on an appearance model of each object which is used to link detections from frame to frame into tracklets [107]. This appearance model can be learned as an embedding into an appearance space. In the prior literature [42], detections and associated embeddings are generated for each frame individually, then subsequently the embedding are used to link the frames together. Therefore, the embeddings learned by the network must somehow be a function of the appearance of each object, consistent from one frame to another. In autonomous driving scenarios, this is difficult because objects are often small (*e.g.* far away vehicles), poorly visible (*e.g.* backlit and dark vehicles) or ambiguous (*e.g.* very similar car model). Therefore some of the vehicles may have very similar appearances and thus very similar embeddings.

In our method, we do not explicitly rely on such embeddings. Instead we create a CNN which propagates a hidden state through time. It is the responsibility of the CNN to propagate whatever information is relevant for tracking through this hidden state. This information might be relevant to tracking but might also include relevant information to improve the precision of subsequent detections.

Our approach is based on the Single Stage Detector (SSD) defined by Liu *et al.* [40] and the Rolling Recurrent Convolution (RRC) defined by Ren *et al.* [44]. For each frame, our network produces a fixed number of detections at different scales at predefined prior locations [40]. For each of those detections, the network generates three group of outputs: classification (background, bicycle, bus, car, motorcycle, pedestrian, trailer, and truck), 2D & 3D location (bounding box) and the relative 3D position of the detection between the last and current frame (velocity). 3D

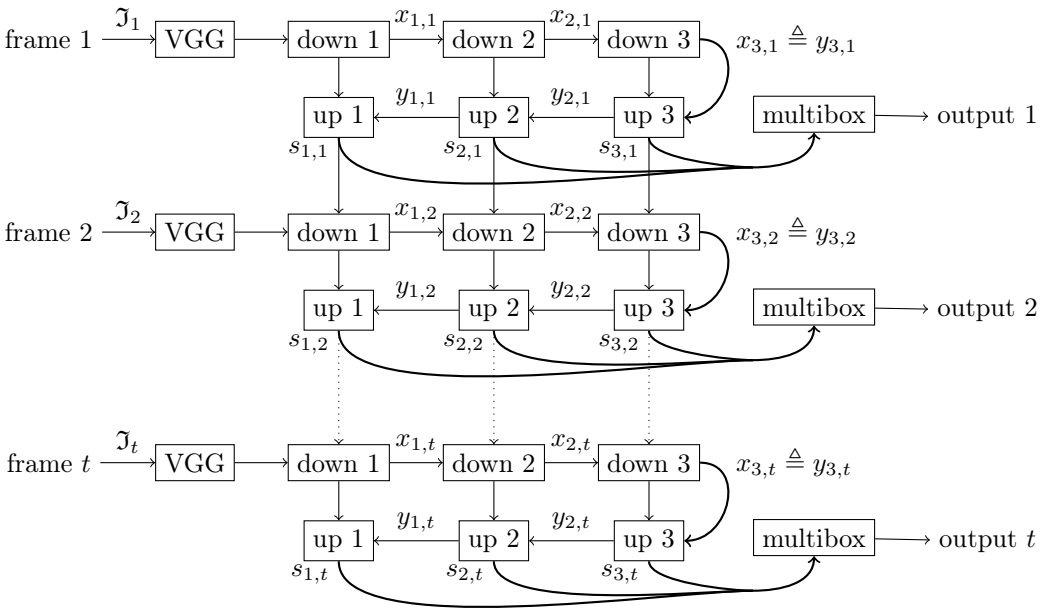


Figure 5.1: Overall architecture for a network with 3 scales.

tracking presents the advantage over 2D tracking that objects can always be separated by their position in 3D even when highly overlapping in the 2D image space.

We first introduce the overall network architecture, including the multi-scale and recurrent aspects of our approach (Section 5.2.1). Subsequently, we describe the details of the upscaling and downscaling blocks constituting the network (Section 5.2.2); followed by the description of the network outputs and loss function (Section 5.2.3). Finally, we conclude by presenting the inference process (Section 5.2.4) and training process of our network (Section 5.2.5).

5.2.1 Overall Network Architecture

As shown in Figure 5.1, a given input image $\mathfrak{I}_t$ captured at time t is fed through a VGG16 trunk up to the last layer of the fourth convolution block *conv4_3* [3], the resulting feature map tensor

denoted VGG($\mathfrak{I}_t$) is used as input to the first scale of our network. Our approach following the principles of Ren *et al.* [44] and Liu *et al.* [40] is multiscale and aggregates features from different scales into a single stage network, in contrast to two-stages networks such as Faster R-CNN [34]. This is done using an hourglass subnetwork, where at each scale, the feature maps are either downscaled or upscaled by a factor of 2. In downscaling blocks, if the spatial dimensions of the input tensors are not a multiple of two, we automatically pad the input tensors to round up the dimensions to a multiple of 2.

The input feature map VGG($\mathfrak{I}_t$) is fed through a cascade of downscaling blocks creating feature maps ($x_{i,t}$) using both the feature map from the previous scale $x_{i-1,t}$ as well as temporal hidden state $s_{i,t-1}$ from the previous frame at $t-1$. Subsequently the information is propagated upward through a cascade of upscaling blocks generating feature maps ($y_{i,t}$). Those upscaling blocks are also connected to their counterpart downscaling block at a given scale using skip connections. Those skip connections are implemented in the same way as the temporal connections across frames. The output of the downscaling network is a tuple of tensors ($s_{i,t}$), which are used as input to the next frame at $t+1$ as well as input to the multibox heads which produce the final neural network outputs. For the first frame of the video sequence, the hidden state ($s_{i,0}$) is zero-initialised.

The neural network outputs are generated using a multibox head as defined by Liu *et al.* in their SSD network [40] for each scale as shown in Figure 5.1. Each multibox head is fed from the corresponding hidden state feature map $s_{i,t}$ and produce the network outputs as defined in Section 5.2.3: classification, pose estimation and tracking outputs. For each output, the multibox head is constituted of a 3×3 convolutional layer producing the required number of channels for the output kind.

The number of channels and output boxes at each scale is shown in Table 5.1. The downsampling ratio shown in the table indicates the size of the feature map at this scale compared to the original

Scale i	№ channels	№ boxes	Downsampling ratio
1	4096	12	$1/8$
2	1024	24	$1/16$
3	256	24	$1/32$
4	256	24	$1/64$
5	256	6	$1/128$

Table 5.1: The number of channels for $x_{i,t}$ and $s_{i,t}$ and the number of boxes for each scale i.

image size. The first scale as a ratio of $1/8$ which corresponds to the ratio of the *conv4_3* layer in the VGG16 trunk [3].

5.2.2 Downscaling and Upscaling Blocks

We have shown in Figure 5.1 that the network is a succession of hourglass subnetworks constituted of downscaling and upscaling blocks. The details of a pair of upscaling and downscaling blocks at scale i and time t is shown in Figure 5.2. The first step of the upscaling block is to normalise its inputs $x_{i-1,t}$ and $s_{i,t-1}$ using Group Normalisation (GN) [205] then the inputs are added together to merge the information of the previous and current frame. This feature map is fed through a 3×3 convolution layer called the *merge block*. Subsequently, the network is forked in two parts: the downscale block is a 2×2 convolution layer of stride 2 used to downscale the feature map by a factor of two to produce the next scale $x_{i,t}$; the adapt block is a 3×3 convolution to produce the input of the upscaling block. The upscaling block is essentially constituted of the same operations as the downscaling in reverse order. The upscale blocks use a 2×2 transposed convolution of stride 2 to upscale $y_{i,t}$. Subsequently, the two inputs are normalised using GN and fed through a 3×3 convolution layer.

To reduce the footprint of the network, we use separable filters in all convolutions and transposed convolutions within the downscaling and upscaling blocks. This separable filter is constituted of a 1×1 convolution followed by a $k \times k$ convolution or transposed convolution on each individual channel. Both convolutions are followed by a Rectified Linear Unit (ReLU) activation function.

Each downscaling block is constituted of a merging layer (as shown in Figure 5.2) which merges together the input information $x_{i-1,t}$ with the temporal information $s_{i,t-1}$. This is followed by a downsampling layer which halves the size of the feature maps to produce the next scale $x_{i,t}$. Another adaptation block produces the tensor for the skip connection to the upsampling block. Therefore, the upsampling block is essentially an inverted downscaling block.

For feature map normalisation, we use GN [205] with 32 groups instead of the more common Batch Normalisation (BN) [206] because BN operates on a batch of inputs. In particular, BN does not work on batches degenerated to a single element. As described in Section 5.2.5, although, our training procedure accumulates gradients over a batch size of 8 video sequences, we process each sequence separately to decrease the memory footprint. Therefore, BN is not suitable to our training procedure. In contrast, GN does not normalise across the batch and is able to accommodate our training procedure without any changes. We use GN to ensure that the inputs to each upscaling and downscaling blocks have similar magnitudes. If inputs of widely different magnitudes are added together, the training will not converge; hence, GN solves that problem.

We pad all the convolutions in the neural network using the ring padding method described in Section 4.2.5 to process equirectangular panoramas without any left/right border effect. For downscaling convolutions and upscaling transposed convolutions, since kernel size is equal to the convolution stride (of 2), padding is not necessary.

5.2.3 Network Outputs and Training Loss Function

Joint tracking and detection is a multi-task problem composed of three tasks: object classification, object location regression, and tracking. In our approach, object classification is solved using the focal loss defined by Lin *et al.* [76] and 2D location regression is solved using the loss function defined by Liu *et al.* [40]. Pose estimation is solved using an approach similar to Chapter 3. Tracklets are formed by learning the relative 3D position in the previous frame and

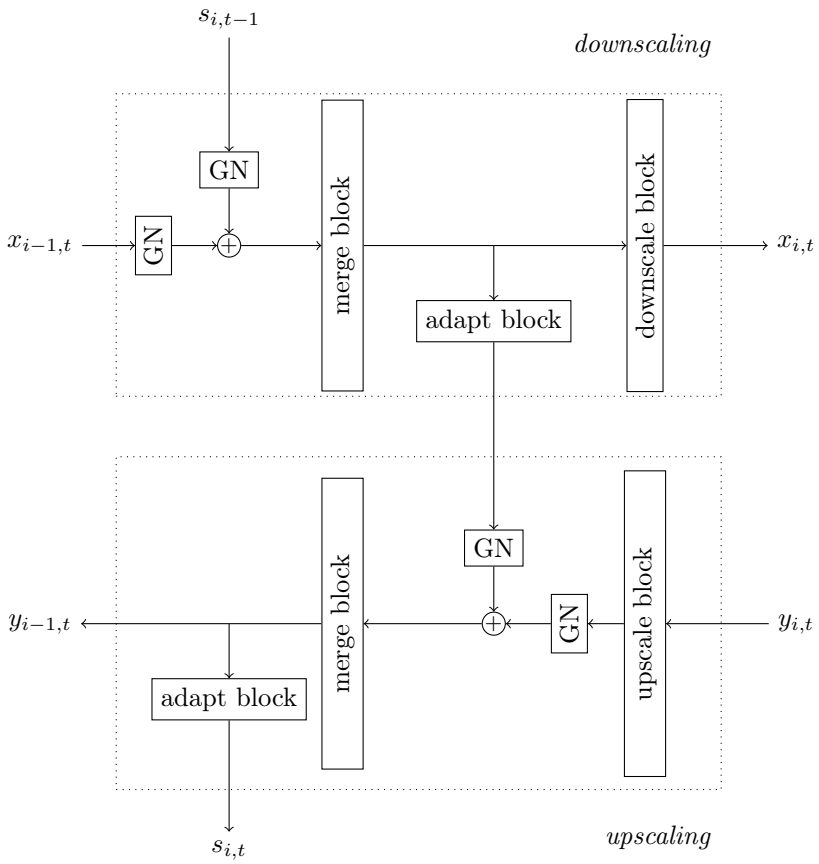


Figure 5.2: A downscaling module and upscaling module receiving inputs from the previous scale $x_{i-1,t}$ and the next scale $y_{i-1,t}$ as well as temporal information from the previous frame $s_{i,t-1}$.

combining the detections frame-to-frame. As in Chapters 3 and 4, we use the methodology of Kendall *et al.* [155] to dynamically adjust the multi-task weights during training.

Classification and 2D Localisation

We use the weighted cross-entropy loss called focal loss $l_{classification}$ defined by Lin *et al.* [76] to classify the object category. The likelihood of each category is defined using a softmax function and the loss function for a given object is defined as:

$$L_{classification} = -(1 - s_t)^{\gamma} \log(s_t) \tag{5.1}$$

where $\gamma = 2$ and t is the index of the ground truth category for the object and s_t is the t-th output of the softmax function.

Given the ground truth bounding box $\hat{\mathbf{b}} \in \mathbb{R}^4$ and the box predicted by the neural network $\mathbf{b}_o \in \mathbb{R}^4$, we use the smooth L_1 loss to regress the 2D bounding box relative to the prior anchors $\mathbf{b}_b$ as defined by Liu *et al.* [40]:

$$L_{loc} = \text{SmoothL1Loss}\left[\mathbf{b}_o, \frac{\hat{\mathbf{b}} - \mathbf{b}_b}{\mu}\right] \tag{5.2}$$

where μ is the relative position variance as defined by Liu *et al.* [40].

The loss functions are summed across all boxes, however unlike Liu *et al.* [40], we do not normalise the loss by the number of boxes. Such normalisation penalises the gradients on image with many detections. The aim of such normalisation is to keep the magnitude of the gradients similar across different images and consequently decreases the importance of detections in images containing many objects. Those images are often the hardest and most interesting examples. In contrast, examples with a single vehicle are not so interesting from a tracking perspective. Since, the focal

loss [76] already takes hard examples into account, we do not use Online Hard Example Mining (OHEM) [40, 67].

3D Pose Estimation

We extend the neural network to 3D object pose estimation using a similar approach to Chapter 3. For each detection, we regress the centre position, orientation and size of the object 3D bounding box. Given the ground truth centre position $\hat{\mathbf{p}} = (\hat{u}, \hat{v}) \in \mathbb{R}^2$, orientation $\hat{\mathbf{q}} \in \mathbb{H}$ and size $\hat{\mathbf{s}} = (\hat{h}, \hat{l}, \hat{w}) \in \mathbb{R}^3$. We aim to learn to detect their counterpart centre $\mathbf{p} = (u, v) \in \mathbb{R}^2$, orientation $\mathbf{q} \in \mathbb{H}$ and size $\mathbf{s} = (h, l, w) \in \mathbb{R}^3$. For this, we add another set of outputs to the network $\mathbf{p}_o \in \mathbb{R}^2$, $d_o \in \mathbb{R}$, $\mathbf{s}_o \in \mathbb{R}^3$, $\mathbf{q}_o \in \mathbb{H}$.

The centre of each bounding box is projected to image space using the camera matrix P, then the 2D centre coordinates in pixels and the distance in meters are regressed separately. We define the image space as using relative coordinates in the space $[0, 1]^2$ rather than the pixel space $[0, w] \times [0, h]$ and the camera matrix is scaled accordingly. This matches the definition employed for the 2D bounding box location regression [40]. The 2D centre of the 3D bounding box is regressed using the same approach as the 2D bounding box centre (as per [40]) using a smooth L_1 loss [65]:

$$L_{centre} = \text{SmoothL1Loss}\left[\mathbf{p}_o, \frac{\hat{\mathbf{p}} - \mathbf{p}_b}{\mu}\right] \tag{5.3}$$

where μ is the relative position variance as defined by Liu *et al.* [40] and $\mathbf{b}_b$ is the position of the anchor.

We regress the distance using a smooth L_1 loss [65]. The distance is multiplied by a constant defined as the ratio of the width of the prior anchor box w_{prior} and the focal length f to ensure

that the average magnitude of the learned representation remains the same at different scales and is independent of the image resolution. Being independent of the image resolution allows the network to be trained and used across different datasets and different cameras with different focal lengths. The distance loss function is therefore defined as:

$$L_{distance} = \text{SmoothL1Loss}\left[d, \log\left(\hat{d} \cdot \frac{w_{prior}}{f}\right)\right] \tag{5.4}$$

Unlike Chapter 3, we propose two approaches to regress the orientation. The first approach classifies the angle into two bins: $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and $[-\pi, -\frac{\pi}{2}] \cup [\frac{\pi}{2}, \pi]$ and we estimate the angle using an L_2 loss function within each bin. We train the estimator of each of the two bins for angles which fall within the bin or within $\frac{\pi}{4}$ radians of the bin. Training the estimator for angles falling slightly outside the boundary of a bin increases the accuracy for angles close to the boundary.

The second approach is based on quaternions. While the KITTI dataset only records the heading of the vehicles, the nuScenes dataset records the full 3D orientation of each object in the scene. Unlike an angle in radians, the space of rotation is continuous in quaternion space. Quaternions introduce an undesirable ambiguity for regression since the quaternion q and its opposite $-q$ represent the same orientation. In automotive, the axis of rotation is somewhat limited to the upward y-axis, hence we resolve this ambiguity by choosing quaternions such that $q_y > 0$. Therefore, we are able to directly regress orientation using a cosine similarity using the following loss function $L_{orientation}$:

$$S(\mathbf{q}, \hat{\mathbf{q}}) = \frac{\mathbf{q} \cdot \hat{\mathbf{q}}}{\max\{\|\mathbf{q}\| \|\hat{\mathbf{q}}\|, \epsilon\}} \qquad \text{where } \cdot \text{ is a dot product} \tag{5.5}$$

$$L_{orientation} = \arctan\left(\epsilon_1 \cdot |S(\mathbf{q}_o, \hat{\mathbf{q}})| + \epsilon_2\right) \tag{5.6}$$

where S is the cosine similarity, $\mathbf{q}_o$ is the network output and $\hat{\mathbf{q}}$ is the ground truth quaternion, ϵ is a small constant ($\epsilon = 10^{-8}$). The cosine similarity is essentially the cosine of the angle between the two quaternions (in quaternion space, not 3D space) however the gradient converges to 0 as the angle tends to 0 or $\pm\pi$. Those small gradients reduce the effectiveness of the loss function. Therefore, we use the arctangent function to linearise the loss function with respect to the angle and improve convergence. The two small constants ϵ_1 and ϵ_2 prevents the gradient of the arctangent from reaching $\pm\infty$ for colinear quaternions.

During inference, we can recover the detection orientation $\mathbf{q}$ by normalising the output quaternion:

$$\mathbf{q} = \frac{\mathbf{q}_o}{\|\mathbf{q}_o\|} \tag{5.7}$$

This loss function does not constraint the magnitude of the quaternion $\mathbf{q}$ which is not as important as its orientation. To prevent the quaternion magnitude from diverging to $+\infty$ during training, we add a small regularisation term using the L_2 norm $\|\mathbf{q}_o\|$.

Finally, the size is directly regressed in meters using a smooth L_1 loss:

$$L_{size} = \text{SmoothL1Loss}(\mathbf{s}, \hat{\mathbf{s}}) \tag{5.8}$$

By combining those four outputs (centre, distance, orientation and size), we are able to estimate the 3D pose and size of each object in the scene.

3D Tracking

Object tracking is performed frame-to-frame by estimating the motion of each object between each frame. Given the spherical coordinates of a ground truth object $\mathbf{p}_t^s = (\hat{\lambda}_t, \hat{\phi}_t, \hat{d}_t)$ at a time t. We define the motion $\hat{\mathbf{v}}_t$ between the two frames as:

$$\hat{\mathbf{v}}_t = (\Delta\hat{\lambda}_t, \Delta\hat{\phi}_t, \Delta\hat{d}_t) \tag{5.9}$$

where $(\Delta\hat{\lambda}_t,\ \Delta\hat{\phi}_t,\ \Delta\hat{d}_t)$ is the spherical coordinates difference between time $t-1$ and t.

For each location, the neural network estimate the motion $\hat{v}_t$ using a smooth L_1 loss:

$$L_{tracking} = \text{SmoothL1Loss}\left(\Delta\mathbf{v}_{t,o}, \Delta\hat{\mathbf{v}}_t\right) \tag{5.10}$$

This motion vector $\hat{v}_{t,o}$ can be used to link the detections at inference time together into tracklets.

5.2.4 Inference

Our network architecture can be used for online video stream processing as the inputs at a given time does not depend on subsequent frames. We feed each image through the hourglass network generating the hidden state s_t for the next iteration as well as the detections for the current frame and repeat the process with the next iteration.

For each frame, the detections are grouped into objects using Non Maximum Suppression (NMS) [34, 40] on the 2D bounding box in the spherical coordinate space (latitude and longitude). For each detection in a frame, by combining the object position in spherical coordinates $\mathbf{p}_t^s$ with the motion vector $\mathbf{v}_t$, we can compute its estimated position in the previous frame $\bar{\mathbf{p}}_{t-1}^s = \mathbf{p}_t^s - \mathbf{v}_t$. The estimated position in the previous frame can be used to search for the nearest neighbour

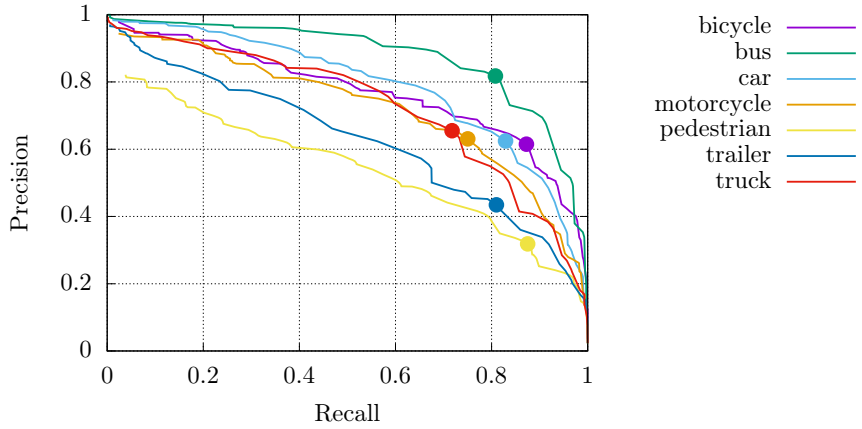


Figure 5.3: Frame-to-frame detection matching precision/recall upper bound achieved while varying the matching scaling factor for each object category. The circle on each curve indicates the scaling factor selected during inference.

object in the previous frame. We define a scaling factor ρ to take into account the scale difference between the longitude and latitude expressed in radians and the distance expressed in meters. Besides, depth estimation is a much more difficult problem than latitude and longitude estimation therefore it is desirable to allow for a greater margin of error for the distance. We define the distance between two points $\mathbf{p}_1$ and $\mathbf{p}_2$ as:

$$d(\mathbf{p}_1, \mathbf{p}_2) = \left\| \frac{\mathbf{p}_1 - \mathbf{p}_2}{\rho} \right\| \tag{5.11}$$

Using greedy matching, we find the assignment σ which minimises the distance between detections at time $t-1$ and the estimation at time t using the metric d. We reject any association if the metric d between the object position and its estimation is greater than 1.

To determine the constant scaling factor ρ, we compute the distance metric d for positive and

negative matches on the validation set and use grid search on the scaling factor to find the upper bound of the precision/recall curve as shown in Figure 5.3. We pick the scaling factor which maximise the sum of the precision and recall for each category (shown by a cross on Figure 5.3).

While training the neural network over multiple frames is a slow process, inference works a frame at a time, therefore our approach runs in real-time during inference at a frame rate of 10Hz on a GeForce 2080 Ti on large 360° panoramic imagery from the nuScenes dataset [6].

5.2.5 Training at Scale

The network is trained using the AdamW optimiser [207] on a minibatch of 8 video sequences using a learning rate of 1×10^{-4}. The network is pretrained first on pair of images from the KITTI detection and tracking datasets [1] for about 30,000 iterations and subsequently on pair of 360° equirectangular images from the nuScenes dataset for 30,000 iterations and finally fine-tuned on 360° equirectangular video sequences of 10 frames from the nuScenes dataset [7] for 40,000 iterations. We use pre-trained VGG weights [3] from ImageNet [84] and initialised the remaining weights of the network using a Xavier initialisation with a uniform distribution [208] while the multi-task loss weights [155] are initialised to 1.

Unfortunately, it is not possible to hold more than two or three frames of a single sequence in memory on modern GPU hardware. For those experiments, we use a GeForce 2080 Ti with 11Gb of memory. Therefore, we solve two problems: how to process a single sequence if it does not fit into memory and how to process a whole minibatch of sequences. Instead of processing whole minibatches, we process one video clip at a time and accumulate the gradients from each video clip. Subsequently, we update the network weights using the AdamW optimiser. This is mathematically equivalent to processing whole batches but requires 8 times less memory for the intermediary results.

We use mixed-precision training using the O1 optimiser provided by Nvidia Apex [203, 204] to reduce the memory requirements by half. In order to prevent gradient overflow, we use loss scaling [204] and normalise the input image intensity range to the range $[-1, 1]$. The input image normalisation is necessary in order to have CNN weights and biases of similar magnitudes, thus reducing 16-bits gradient overflows.

Despite the reduced memory footprint of mixed-precision training, it does not scale to arbitrary video sequence length. Therefore, we split our network into subcomponents using the methodology of Wang *et al.* [174] described in Section 2.2.2. For each video clip, we attempt to process the frames individually. To do so, we process the video clip fully in a first forward pass and store the temporal information ($s^{t,i}$) while throwing away all other information. This allows us to recompute the outputs of a given frame without having to recompute all the previous frames. In a second pass, we re-compute each frame in a reverse order, backpropagating and accumulating gradients along the way. The memory required to store the tensors ($s^{t,i}$) is negligible compared to the amount of memory required to backpropagate through a single frame. Besides for very large memory clips (> 100 frames), we can stream the tensors to and from the main memory or a hard disk with little performance overhead as it is possible to hide the latency using memory prefetching. In that respect, the algorithm has a near constant GPU memory complexity regardless of the number of processed frames.

5.3 Evaluation

We evaluate our approach on the recently released nuScenes tracking benchmark [7]. The nuScenes dataset is much larger than the KITTI dataset [1, 8] used in Chapter 3 and 4 and include a greater number of modalities (6 cameras, 5 radars, 1 LiDAR) however our approach relies only on the camera imagery.

The multi-view setup of the nuScenes dataset is comprised of six cameras: front, front left, front right, left, right and back. The cameras FoV have little overlap as shown in Figure 5.4. We stitch the imagery from the six cameras into a single equirectangular image of size 2048×175 by projecting each image using its corresponding calibration. The equirectangular image is centered around the front camera. In regions where two images overlap each other, we blend the two images together.

For each frame of the front camera, we generate a corresponding equirectangular image. Since the cameras are not synchronised between each other, we use the frames from each cameras which have the closest timestamp to the front camera frame. Only a small subset of the nuScenes dataset frames are labelled (on average, one every 5^{th} frame at 5Hz) therefore we interpolate those ground truth labels between the keyframes. The ground truth labels are interpolated in world space then projected to the front camera space. Subsequently, we train the network using the procedure outlined in Section 5.2.5.

5.3.1 Qualitative Analysis

Qualitative examples of the results of our approach on the nuScenes Tracking Benchmark are shown in Figures 5.5–5.10. Overall, the network performs substantially better than our method proposed in Chapter 4 because it has been trained directly on 360° imagery. However, to some extent, the approach struggles to generalise well on objects close to the camera ($< 2\,\text{m}$) as shown in the first five frames of Figure 5.6; which can be attributed to the relatively small number of ground truth objects close to the camera. Our approach works well on vehicles but struggles on pedestrians as the dataset features many large groups of pedestrians or individuals close to each other as shown in Figures 5.6 and 5.9. This increases detection inaccuracy (false positives and false negatives) and increases tracking target switches. Objects further from the egovehicle are also difficult to detect because they appear closely grouped, highly overlapping each other, while the distance estimation error increases.

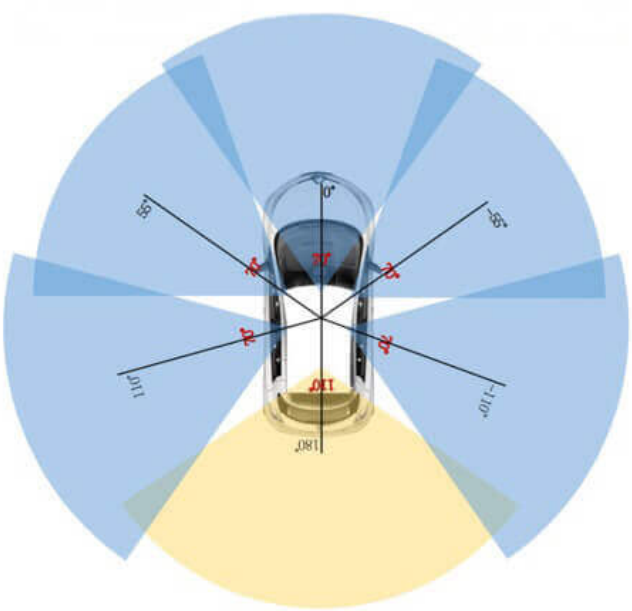


Figure 5.4: Field of view of the cameras of the nuScenes setup (reproduced from nuScenes [6]).

5.3.2 3D Pose Estimation

The nuScenes tracking benchmark [7] relies on 3D pose to match detections and ground truths; thus, the 3D pose quality is directly influencing the tracking MOTA and MOTP. The benchmark imposes a hard threshold of 2 m and any ground truth matching beyond the threshold is ignored. To evaluate pose estimation, we choose to match detections and ground truths based on 2D IoU > 0.5 (as in Chapter 3) on the nuScenes validation set.

The detection mAP and AOS are shown in Table 5.2 for different ground truth matching criteria: 2D IoU > 0.5, 3D IoU > 0.5 and 3D distance thresholds (as per nuScenes [7]). Overall, detection is much more challenging in 3D than 2D. The IoU-based metrics of the KITTI benchmark heavily penalises detections of small objects such as bicycles and pedestrians, compared to distance-based metrics; because the IoU is dependent on the size of the object.

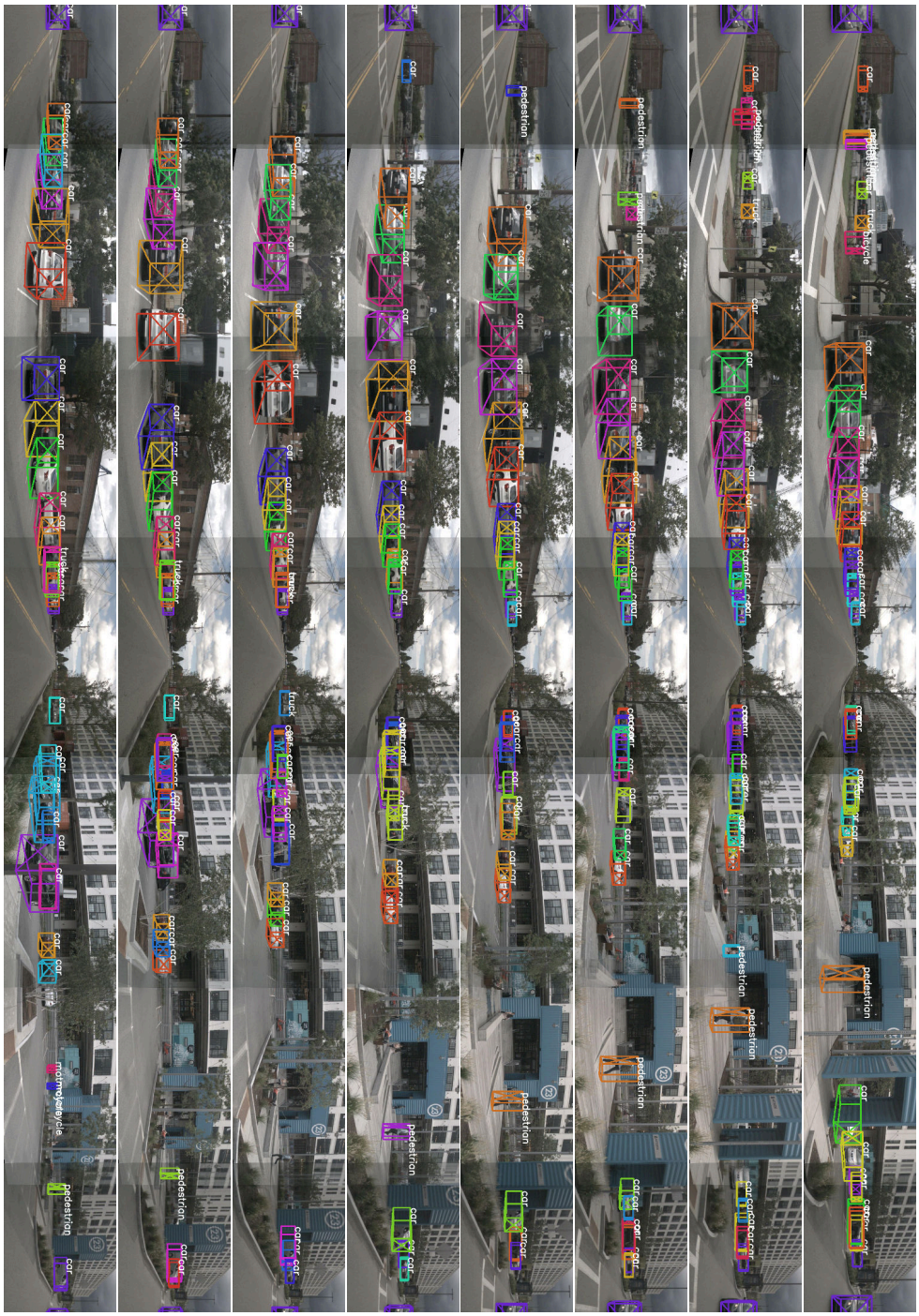


Figure 5.5: Quantitative results on the nuScenes dataset.

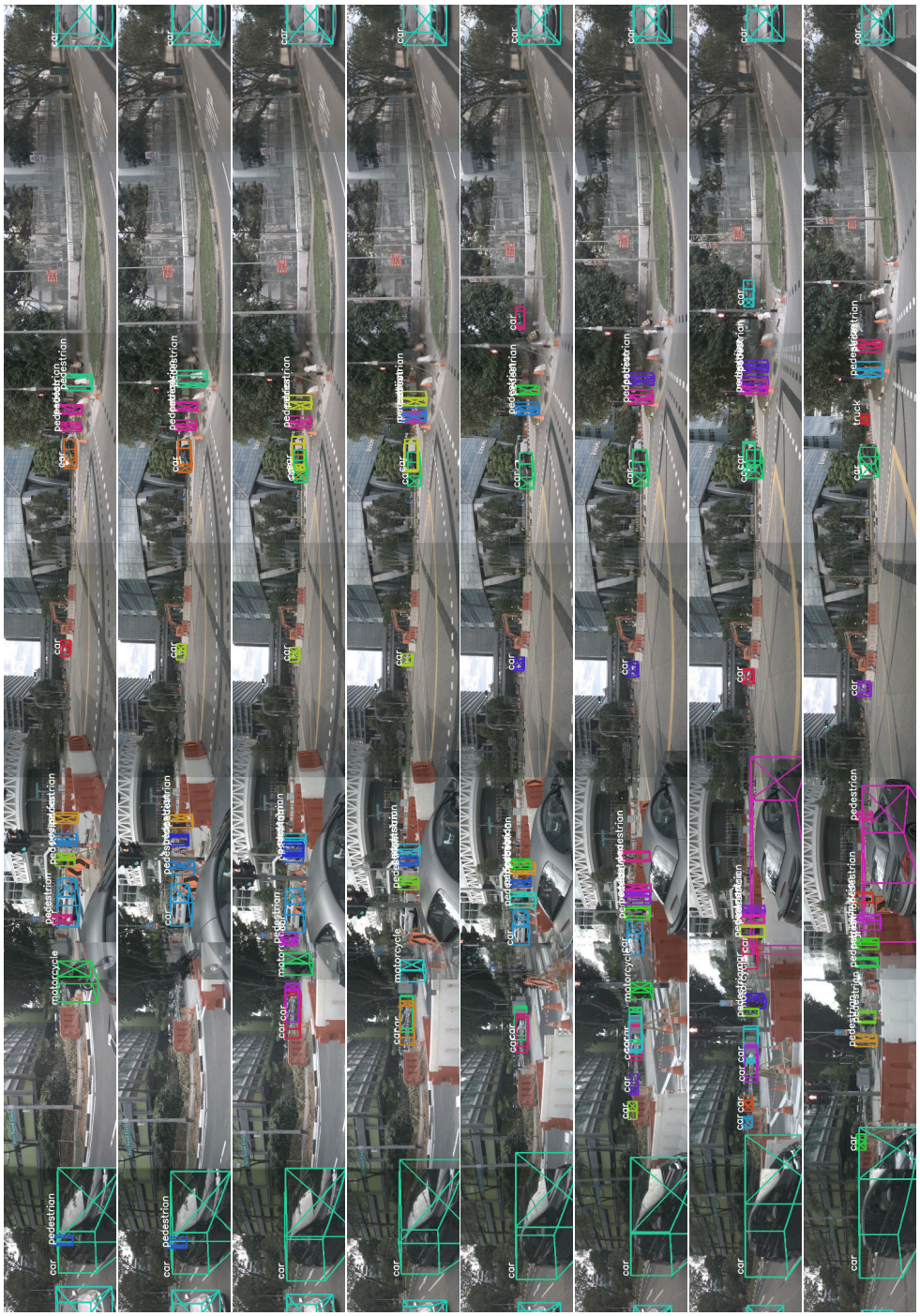


Figure 5.6: Quantitative results on the nuScenes dataset.

Figure 5.7: Quantitative results on the nuScenes dataset.

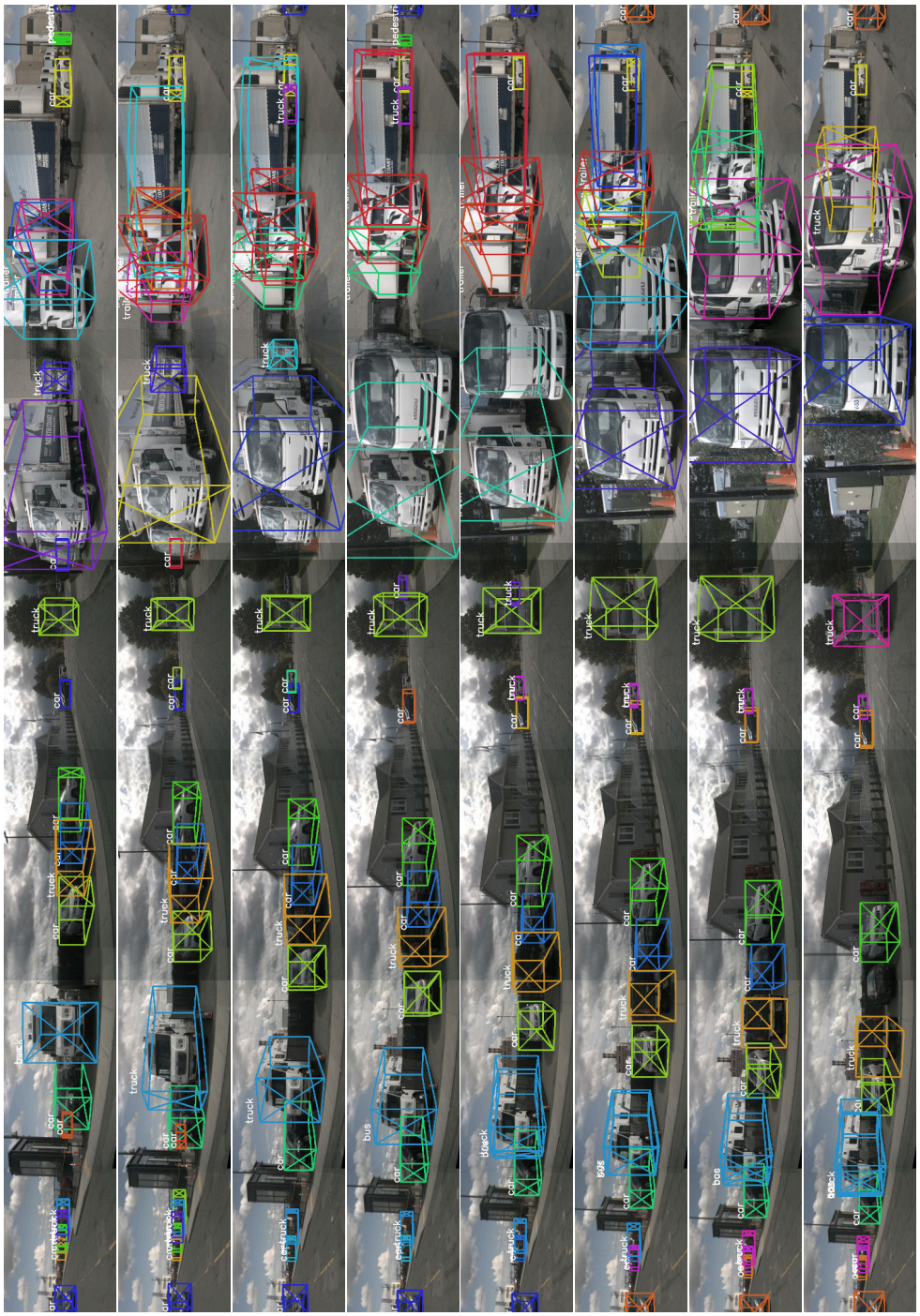


Figure 5.8: Quantitative results on the nuScenes dataset.

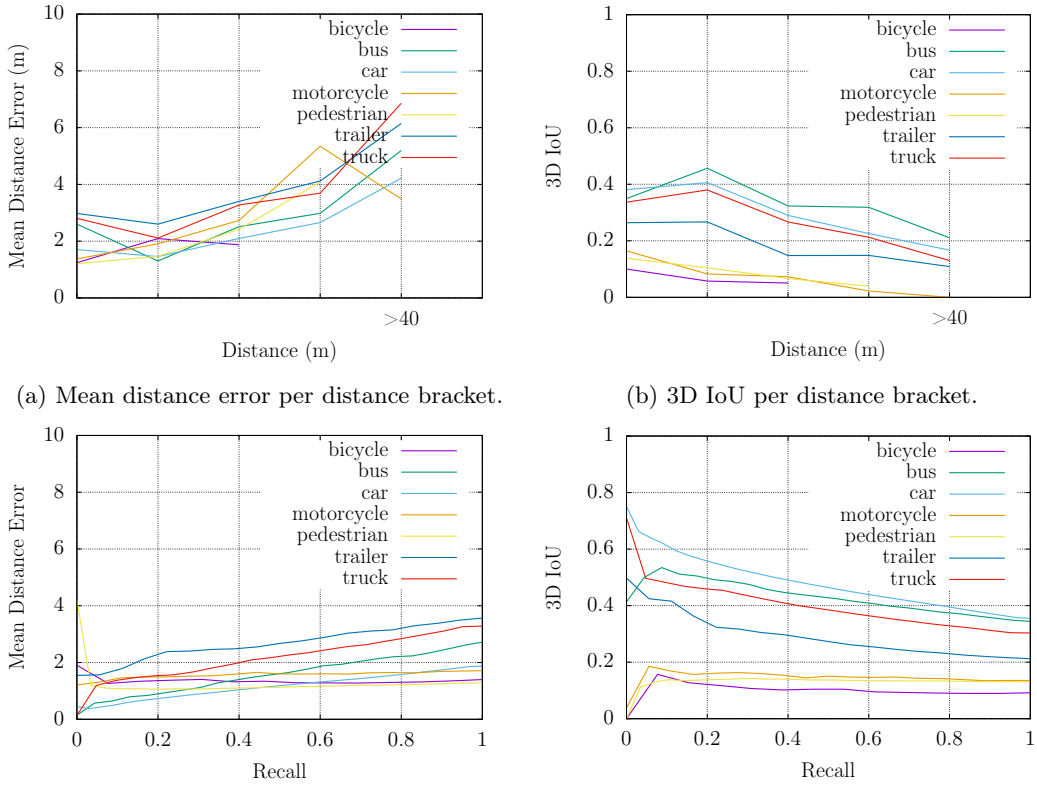
Figure 5.9: Quantitative results on the nuScenes dataset.

Figure 5.10: Quantitative results on the nuScenes dataset.

Category	mAP† 2D IoU	mAP† 3D IoU	mAP† 3D dist	Average Orientation Similarity (AOS)† 2D IoU	Average Orientation Similarity (AOS)† 3D IoU	Average Orientation Similarity (AOS)† 3D dist
bicycle	0.203	0.006	0.130	0.452	0.125	0.237
bus	0.490	0.082	0.217	0.766	0.417	0.562
car	0.679	0.194	0.395	0.790	0.565	0.656
motorcycle	0.247	0.001	0.165	0.417	0.020	0.310
pedestrian	0.423	0.003	0.339	0.464	0.025	0.379
trailer	0.253	0.018	0.090	0.455	0.018	0.226
truck	0.349	0.068	0.148	0.623	0.393	0.457

† Higher, better

Table 5.2: Detection results on the nuScenes [7] validation set: mAP and AOS for different evaluation methods



(a) Mean distance error per distance bracket.

(b) 3D IoU per distance bracket.

(c) Mean distance error as a function of recall.

(d) 3D IoU as a function of recall.

Figure 5.11: Statistics for all categories of the nuScenes [7] validation set.

Figures 5.11a–5.11d show the pose estimation performance for different distance brackets (Figs. 5.11a and 5.11b) and different recall thresholds (Figs. 5.11c and 5.11d). Performance decreases as distance or recall increases. This implies that the higher scoring detections have better quality 3D pose and bounding boxes than low scoring ones. Since the nuScenes tracking benchmark uses a ground truth matching threshold of 2 m, any of the detections with a distance error greater than 2 m will be ignored.

5.3.3 Tracking Benchmark

Tracking is evaluated using the metrics provided by the nuScenes Tracking Benchmark [7], which were originally introduced by Weng *et al.* [102]. The results of our method compared to the public leaderboard are shown in Table 5.3 while the per-object category results are shown in Table 5.4. The AMOTA and AMOTP of our method is significantly behind the top methods on the leaderboard (StanfordIPRL-TRI [201], Megvii-AB3DMOT [102, 209]); however those methods are LiDAR-based. In contrast, our method performance are similar to CenterTrack_Vision [202] which is the only other camera-based method. Overall, as shown in Table 5.3, our approach is much more successful at detecting and tracking cars than other types of vehicles and pedestrians. This can also be attributed to the lack of training data. The number of ground truth examples for each category is as follows: 58317 car, 25423 pedestrians, 9650 truck, 2425 trailer, 2112 bus, 1977 motorcycle, 1993 bicycle. Apart from the pedestrian category, all the other categories have significantly less examples than the car category.

5.4 Summary

In this chapter, we have demonstrated a novel approach to multi-object tracking-by-detection (MOTD) extending our work on 3D pose estimation and tracking presented in Chapter 3 and panoramic imagery in Chapter 4 using a single end-to-end CNN which operates directly on

Approach	Modalities	AMOTA†	AMOTP (m)‡	MOTAR†	MOTA†	MOTP (m)‡	Recall†
StanfordIPRL-TRI [201]	Lidar	0.550	0.798	0.768	0.459	0.353	0.600
Megvii-AB3DMOT [102, 209]	Lidar	0.151	1.501	0.552	0.154	0.402	0.276
CenterTrack_Open [202]	Camera, Lidar	0.108	0.989	0.267	0.085	0.349	0.412
Our method	**Camera**	**0.045**	**1.819**	**0.242**	**0.050**	**0.970**	**0.260**
CenterTrack_Vision [202]	Camera	0.046	1.543	0.231	0.043	0.753	0.233
PointPillars-AB3DMOT [102, 210]	Lidar	0.029	1.703	0.243	0.045	0.824	0.297
Mapillary-AB3DMOT [102, 211]	Camera	0.018	1.790	0.091	0.020	0.903	0.353

† Higher, better ‡ Lower, better

Table 5.3: Qualitative results on the nuScenes dataset

Category	AMOTA†	AMOTP (m)‡	Recall†	MOTAR†	MOTA†	MOTP (m)‡
bicycle	0.000	2.000	0.000	0.000	0.000	2.000
bus	0.047	1.718	0.126	0.701	0.081	0.840
car	0.217	1.551	0.360	0.695	0.216	0.638
motorcycle	0.000	1.936	0.409	0.000	0.000	0.701
pedestrian	0.000	1.913	0.380	0.000	0.000	0.826
trailer	0.000	1.926	0.340	0.000	0.000	1.034
truck	0.048	1.692	0.207	0.300	0.056	0.756

† Higher, better ‡ Lower, better

Table 5.4: Per-category results of the proposed approach on the test dataset of the nuScenes tracking benchmark [6]

360° panoramic video sequences. Our approach extends single-stage detectors such as SSD [40] and RRC [44] to video processing using recurrent connections. We have also shown how to scale the training process of such recurrent CNN to sequences of arbitrary length within the memory requirements of current GPU technology using mixed-precision training [203, 204] and by recomputing the required state during gradient backpropagation [174]. Our method is one of the first [102, 202] to be able to estimate and track the 3D pose of a broad range of object categories on the nuScenes tracking benchmark using camera technology alone. It is the only approach which uses 360° panoramic imagery rather than processing each camera imagery individually and achieves state-of-the-art results compared to existing camera-only approaches [102, 202] on the nuScenes Tracking Benchmark.

Chapter 6

Conclusion

In the introduction of this thesis, we identified two key challenges of autonomous driving. The first challenge is to reduce the number of sensors required for self-driving, particularly active sensors, which are usually expensive, lower resolution than cameras, and break in challenging environmental conditions such as rain, heavy reflections, as well as for materials such as black paint. We have proposed research to provide the same functionalities using cameras which are more readily accessible. Our work on this aspect focuses on the extension of monocular imagery to 3D object pose estimation, dense depth estimation and 360° panoramic imagery (Chapter 3 and 4). The second challenge is the processing of video sequences in order to leverage temporal continuity. To this extent, we looked at online end-to-end tracking in 360° panoramic video sequences (Chapter 5).

In Chapter 3, we introduced a novel approach based on Faster R-CNN [34] to detect objects and estimate the 3D pose, size and velocity of those objects in a scene using a single monocular

camera without specifying any geometrical and semantic constraints. We achieve state-of-the-art 3D pose estimation results compared to the contemporary prior work [4, 5]. In addition, we propose an architecture built on a siamese neural network [37] which is able to compute object correspondences inside a pair of past and present images. We have shown that those correspondences can be used to track objects frame-to-frame using a simple matching algorithm or can be used to enrich the input of an existing tracker. We have shown that it is more accurate to regress the speed directly using such a network rather than by computing 3D positions in each frame and calculating the speed as the difference between past and present positions.

In Chapter 4, we extended our work on 3D pose estimation (Chapter 3) as well as prior work on depth estimation [38] to 360° panoramic imagery. Our work is the first of its kind bringing 360° panoramic imagery as an image-based solution to 360° surround perception in automotive. We compensate the lack of availability of contemporary annotated 360° automotive datasets using style transfer [2] to reuse existing automotive datasets [1, 8], crossing the bridge between two different domains and reducing dataset bias. We also propose ring-padding, an approach to seamlessly compute convolutions and pooling across the left/right edges of an equirectangular image. Our work is quantitatively evaluated on a new publicly-available synthetic testing dataset generated using the Carla driving simulator [39] as well as qualitatively on 360° imagery from the Mapillary platform [171].

In Chapter 5, we further extend our work on object detection (Chapter 3) and 360° panoramic imagery (Chapter 4) to track objects over time using a single end-to-end multi-scale recurrent tracking network. We evaluate our method on the recently released nuScenes Tracking Benchmark [6, 7] and our method is one of the first [102, 202] to rely solely on camera imagery which we stitch into a single large equirectangular video sequence, subsequently fed to the tracking neural network. Our method is the first to exploit 360° equirectangular imagery to adapt CNN to 360° sensing without the complexity of multi-view sensor fusion.

The approach presented in this thesis goes toward bringing *level 4* and *level 5* autonomy as described in Chapter 1 as well as providing a more affordable solution to autonomous driving.

6.1 Contributions

In this thesis, we have contributed the following contributions to the field of visual perception for autonomous vehicles:

- a novel approach to 3D pose estimation in monocular imagery based on two-stage detector architectures which achieves state-of-the-art results compared to the contemporary prior work [4, 5] (Chapters 3 and 4);
- an approach to velocity estimation in a pair of monocular images, as well as, an example of frame-to-frame tracking based on a siamese network [37] (Chapter 3);
- a new publicly-available[1] testing dataset of synthetic 360° panoramic imagery generated using the CARLA automotive environment simulator [39] which serves as a basis for future work on domain adaptation to 360° panoramic imagery (Chapter 4);
- a method to adapt existing datasets and neural networks using style transfer [2] to a new modality such as 360° panoramic imagery without any ground truth labels in the target domain. We apply this method to adapt imagery from the KITTI dataset [1,8] to real-world imagery gathered from Mapillary [171] and to our synthetic 360° dataset (Chapter 4);
- an extension of existing 2D convolutions, ROI pooling and ground truth prior box matching to seamlessly compute across the borders of an equirectangular image (Chapters 4 and 5);
- an approach to multi-view object tracking based on a dense end-to-end neural network for tracking objects in 360° equirectangular video sequences. This approach is the first

[1] https://gdlg.github.io/panoramic

approach using 360° imagery and one of the first approach [102,202] based solely on cameras on the nuScenes tracking benchmark [7] (Chapter 5).

6.2 Potential for Impact

This work contributes to raise the awareness that expensive sensor setups on autonomous vehicle could be replaced by more cost-effective solutions based on cameras. We have shown that monocular 360° camera rigs can be used to fulfill the same functions as a LiDAR. Unlike LiDAR, cameras are passive components with no moving parts, more affordable and more readily available. Given current technology, cameras are not quite as accurate (distance-wise) as active sensors, however we propose that the current LiDAR precision is not required to achieve *level 5* autonomy. Indeed, autonomous vehicles today only represent a small fraction of active drivers, while millions of humans drive everyday using a much simpler sensing capability. Human drivers compensate for the lack of active sensing with a more thorough visual perception and a deeper understanding of the interactions between entities and the environment than current technology. This is the level of visual perception that we aim for.

We also touch on the problem of dataset bias and propose a solution based on style transfer. As new camera technology becomes more prevalent in autonomous driving such as High Dynamic Range (HDR), global shutter, infrared night vision, and new multi-view layouts, the ability to reuse existing datasets alongside new technologies will become more critical as gathering and annotating large datasets is time-consuming and expensive. Our approach in Chapter 4 shows a path to adapt existing datasets to new types of camera and helps reduce the size of the dataset required for autonomous vehicle development.

6.3 Limitations

Chapter 4 clearly illustrates an example of dataset bias which prevents current machine learning approaches from generalising well to a new dataset. We have used domain adaptation to improve the generalisation to panoramic datasets, however domain adaptation particularly struggled with the lack of diversity in our synthetic dataset. Style transfer between the KITTI dataset and synthetic dataset introduces many artefacts. Style transfer is limited to style adaptation and as such, it cannot compensate for the lack content diversity. While we applied style transfer to synthetic images for a quantitative evaluation, the converse proposition is becoming more prevalent as simulated environments are used for training autonomous vehicles [147, 212, 213].

By design, our approach in Chapter 5 does not decouple the object detection and tracking tasks. As tracking information is represented using a spatially dense network rather than sparse interactions between objects, the network cannot efficiently learn complex tracklet behaviour and interactions. Learning such behaviours would require vast amount of training examples and the meaningful patterns themselves would be lost in the noise during training with Stochastic Gradient Descent (SGD). Therefore, due the complexity of the network compared to the task, the tracking task is prone to overfitting on small datasets such as KITTI. This overfitting prevents the network from learning more complex tracking behaviours such as occlusions between multiple vehicles or with static objects (*e.g.* buildings). A clear separation between the detection and tracking steps would introduce resilience inside the tracker against detection failures. In contrast, our approach is not resilient to detection failures as it is based on the assumption that the detection and tracking mutually benefit from being solved jointly. Since our approach is inherently frame-to-frame, it does not work with object Re-Identification (ReID). In 360° equirectangular imagery, objects cannot by definition leave the field of view and therefore can only leave the scene through occlusion behind a building or disappearance in the distance. In either cases, unlike multi-view tracking applications such as video surveillance [214], object ReID is not a

requirement of autonomous driving.

Object detection and tracking based on equirectangular imagery requires the stitching of mutiple views into a single image (Chapter 5) however if the cameras are not synchronised, this might induce a noticeable shift of the position of objects which has two consequences: uncertainty about the actual position of the object during inference, and unprecise ground truth position, especially at the seams between images, during training. In the nuScenes dataset [7], the cameras are positioned quite far apart from each other on the vehicle (up to 2 m away) which can introduce stitching artefacts on objects close to the camera. The assumption is that this uncertainty is within acceptable bounds for self-driving vehicles.

6.4 Further Work

We have shown that 360° monocular panoramic imagery is a promising area of research to replace expensive sensors with more cost-effective and readily-available cameras, however we have highlighted several limitations of our approach in Section 6.3. We comment on subsequent contemporary work following on from this work and present further areas of research stemming from our approach to improve the reliability of monocular panoramic sensing in three key domains: 3D pose estimation, panoramic imagery, and multi-object tracking.

6.4.1 3D Pose Estimation

Subsequent work on 3D object detection in monocular imagery includes Zhu *et al.* [215] on learning object distances, particularly for distances greater than 40 meters. In contrast to our work in Chapter 3, they learn directly the distance rather than the inverse of the distance and optimise the distance using a re-projection loss. Bao *et al.* [216] combine colour images with a dense monocular depth map as neural network input. Recently, Ding *et al.* [217] adapt pointcloud-based methods typically used with LiDAR to monocular 3D pose estimation.

Depth estimation in monocular imagery is a challenging task due to the lack of strong geometric cues such as the epipolar geometry used in stereo vision. A new avenue would be to explore temporal continuity to improve the consistency of the 3D trajectory of objects across time, similarly to the prior work on 2D trajectory smoothing [86, 94] presented in Section 2.1.1. Our CNN introduced in Chapter 5 has temporal connections in the form of skip connections, which are not designed to allow spatial manipulations of the data to follow the objects motion, unlike spatial transformer networks [218] or optical flow [96], instead the spatial transformations are realised through convolutions which are not as efficient at preserving regressed values such as position and distances through time due to vanishing gradients and non-linearities. An architecture using more efficient spatial manipulations [96, 218] would improve the smoothness of 3D trajectories under a trajectory consistency loss [86].

6.4.2 Panoramic Imagery

Over the last year, we have seen the release of many new automotive datasets which includes multi-view setups [7, 52–54]. As such, it is now possible to build a training dataset of real-world 360° panoramic images. They illustrate that our insight in Chapter 4 is correct and panoramic imagery is indeed critical for driving. Our publication [219] of parts of Chapters 3 and 4 has led to follow-up works [220–224] and a revised version of our synthetic dataset has been used by Plaut *et al.* [221]. This version[2] has been extended to include 3D object detection metrics.

While the domain adaptation approach, which we developed based on style transfer is not needed anymore with the recent release of 360° surround datasets [7, 52, 54], it might still be applied to other modalities which have not yet been explored such as HDR imagery or infrared imagery. The addition of both near- and far-infrared channels to automotive cameras would be particularly useful during night time as they offer a solution to reduce the visible light pollution. While

[2] https://gdlg.github.io/panoramic

it is always possible to record new datasets to accommodate additional modalities, acquiring a new dataset is a very complex, time-consuming and expensive work which could be avoided or considerably reduced with approaches which are able to reduce dataset bias, adapt existing dataset to newer datasets, and improve generalisation in machine learning.

The 360° panoramic imagery stitching uncertainty introduced by the lack of synchronisation could be solved by introducing subframes where, at each camera update, the relevant subregion of the panorama and the corresponding parts of the CNN are updated rather than the whole panorama to add awareness of the timeshift between the different views inside the neural network.

6.4.3 Multi-Object Tracking

We investigated object tracking using dense spatial neural networks rather than sparse networks. The dense approach allows the network to better exploit the geometric information of the image, however it also requires much more memory and computation. Future neural networks could reduce the scene into a much more compact sparse representation. Unlike existing sparse methods which only maintain a list of object, a future sparse network could hold an arbitrary amount of information about all important objects in the scene (*e.g.* vehicles, road signs, road layout) and the interactions between them. This information could be processed much more efficiently with recurrent structures such as a multiscale RNN [225,226] and differentiable neural computers [227].

A 3D object trajectory prediction benchmark has been recently released by nuTonomy in 2020 [6] to complement the existing nuScenes object detection and tracking benchmarks. The neural network architecture presented in Chapter 5 is particularly well suited to this problem, because it features recurrent connections and can use the information from the past frames to interpolate the vehicle trajectories in the future. Object tracking and prediction are very similar tasks. Object tracking under heavy occlusion can become a prediction task for objects which are fully occluded and in its simplest formulation, the output of the prediction task can be defined in

the same way as object tracking; however the information known to solve the prediction task is restricted to past frames and by definition not directly observable.

While the emphasise of our work has been to introduce a new kind of sensor in autonomous driving: 360° monocular imagery; the tracking neural network presented in Chapter 5 features a complex multi-task output including object classification, localisation, 3D pose estimation, and tracking for each object at each anchor for each time step. A naive integration of trajectory prediction would add another dimension to predict the time at future time steps for each present time step. Other extensions such as multi-object tracking and segmentation (MOTS) [98] further increase the dimensionality of this intermediate representation. The focus of those extensions is to build an extensive model of the environment. In contrast, Sauer *et al*. [228] focus on a restricted low-dimensional intermediate representation called affordances (next traffic sign, vehicle distance, distance to road centre, etc), however its limited set of affordances cannot encompass all the situations required for driving, despite most of the information in an extensive representation being certainly redundant. In contrast, a perception neural network could select the salient pieces of information required for autonomous driving into a sparse compact representation. This process could be guided by the feedback from the control system back to the perception system, akin to the problem of Visual Question Answering (VQA) [229], where the visual component is the sensor input, the question comes from the control system and the answer is provided by the perception system; thus, this intermediate representation would reduce the size of the final stages of the perception neural network by multiplexing the outputs, while providing an insight into which visual information is the most pertinent to the control system.

Bibliography

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013. x, 12, 13, 28, 60, 62, 63, 68, 81, 85, 100, 101, 115, 116

[2] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251. x, 7, 9, 28, 29, 32, 70, 81, 84, 115, 116

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, 2015. xi, xii, 15, 18, 36, 37, 38, 70, 89, 91, 100

[4] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D Bounding Box Estimation Using Deep Learning and Geometry," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2016, pp. 5632–5640. xi, 7, 26, 35, 39, 40, 43, 44, 51, 62, 81, 115, 116

[5] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware Convolutional Neural Networks for Object Proposals and Detection," in *2017 IEEE Winter Conference on Ap-*

plications of Computer Vision (WACV). IEEE, May 2017. xi, 7, 35, 43, 44, 51, 115, 116

[6] "nuScenes dataset," https://www.nuscenes.org/, 2020. xiii, xv, 86, 87, 100, 103, 112, 115, 121

[7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," *arXiv:1903.11027 [cs, stat]*, Sep. 2019. xiv, xv, 8, 12, 18, 20, 46, 85, 87, 88, 100, 101, 103, 110, 111, 115, 117, 119, 120

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361. xv, 9, 11, 14, 18, 28, 32, 41, 42, 43, 47, 50, 60, 62, 63, 68, 72, 81, 85, 101, 115, 116

[9] D. Eigen, C. Puhrsch, and R. Fergus, "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. MIT Press, 2014, pp. 2366–2374. xv, 25, 82

[10] KPMG, "Self-Driving Cars: Are We Ready?" Tech. Rep., Oct. 2013. 1

[11] J. Elias, "Alphabet exec says self-driving cars 'have gone through a lot of hype,' but Google helped drive that hype," https://www.cnbc.com/2019/10/23/alphabet-exec-admits-google-overhyped-self-driving-cars.html, Oct. 2019. 1

[12] AAA, "Advanced Driver Assistance Technology Names," American Automobile Association (AAA), Tech. Rep., Jan. 2019. 1

[13] Steve, "Cars with Autopilot in 2020," https://www.autopilotreview.com/cars-with-autopilot-self-driving/, Jan. 2020. 1

[14] S. International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Jun. 2018. 1, 2, 4

[15] J. Walker, "The Self-Driving Car Timeline – Predictions from the Top 11 Global Automakers," https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/. 1

[16] S. M. Casner, E. L. Hutchins, and D. Norman, "The Challenges of Partially Automated Driving," *Commun. ACM*, vol. 59, no. 5, pp. 70–77, Apr. 2016. 1, 2, 3

[17] M. Canellas and R. Haga, "Unsafe At Any Level," *Commun. ACM*, vol. 63, no. 3, pp. 31–34, Mar. 2020. 1

[18] NTSB, "Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian," National Transportation Safety Board, Tempe, Arizona, Tech. Rep. NTSB/HAR-19/03 PB2019-101402, Mar. 2018. 1

[19] ——, "Preliminary Report Highway HWY19FH008," National Transportation Safety Board, Delray Beach, Florida, Tech. Rep. NTSB/HWY19FH008, Mar. 2019. 1

[20] ——, "Preliminary Report Highway HWY18FH011," National Transportation Safety Board, Mountain View, California, Tech. Rep. NTSB/HWY18FH011, Mar. 2018. 1

[21] ——, "Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck Near Williston, Florida," National Transportation Safety Board, Tech. Rep. NTSB/HAR-17/02 PB2017-102600, May 2016. 1

[22] S. Wang and Z. Li, "Exploring causes and effects of automated vehicle disengagement using statistical modeling and classification tree based on field test data," *Accident Analysis & Prevention*, vol. 129, pp. 44–54, Aug. 2019. 2, 4, 5

[23] "Vehicle Technology Survey," American Automobile Association (AAA), Tech. Rep., 2016. 2

[24] "Vehicle Technology Survey — Phase II," American Automobile Association (AAA), Tech. Rep., Mar. 2017. 2

[25] "Vehicle Technology Survey — Phase IIIB," American Automobile Association (AAA), Tech. Rep., May 2018. 2, 4

[26] "Vehicle Technology Survey — Phase IV," American Automobile Association (AAA), Tech. Rep., Mar. 2019. 2

[27] E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen, "The seeing passenger car 'VaMoRs-P'," in *Intelligent Vehicles '94 Symposium, Proceedings of The*, Oct. 1994, pp. 68–73. 3

[28] E. D. Dickmanns, "Vehicles capable of dynamic vision: A new breed of technical beings?" *Artificial Intelligence*, vol. 103, no. 1, pp. 49–76, Aug. 1998. 3

[29] D. A. Pomerleau, "ALVINN: An Autonomous Land Vehicle in a Neural Network," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1989, pp. 305–313. 3

[30] C. Lv, D. Cao, Y. Zhao, D. J. Auger, M. Sullman, H. Wang, L. M. Dutka, L. Skrypchuk, and A. Mouzakitis, "Analysis of autopilot disengagements occurring during autonomous vehicle testing," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 58–68, Jan. 2018. 3

[31] "DARPA Urban Challenge," http://archive.darpa.mil/grandchallenge/, 2007. 3

[32] "Google Self-Driving Car Testing Report on Disengagements of Autonomous Mode," Google, Tech. Rep., Dec. 2015. 3

[33] P. Koopman and F. Fratrik, "How Many Operational Design Domains, Objects, and Events?" in *SafeAI@AAAI*, 2019. 4

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99. 5, 7, 14, 15, 16, 17, 19, 35, 36, 37, 60, 74, 86, 90, 98, 114

[35] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Dec. 2018. 5

[36] R. Szeliski, *Computer Vision: Algorithms and Applications*, ser. Texts in Computer Science. Springer, 2011. 5, 27, 39

[37] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature Verification using a "Siamese" Time Delay Neural Network," in *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan-Kaufmann, 1994, pp. 737–744. 7, 9, 32, 36, 51, 115, 116

[38] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6602–6611. 8, 9, 25, 32, 60, 62, 63, 72, 73, 115

[39] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *1st Conference on Robot Learning*, 2017. 8, 62, 63, 75, 76, 84, 115, 116

[40] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *European Conference on Computer Vision*, vol. 9905, Sep. 2016, pp. 21–37. 8, 16, 88, 90, 92, 94, 95, 98, 113

[41] L. Leal-Taixé, A. Milan, K. Schindler, D. Cremers, I. Reid, and S. Roth, "Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking," *arXiv:1704.02781 [cs]*, Apr. 2017. 8, 19

[42] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep Learning in Video Multi-Object Tracking: A Survey," *Neurocomputing*, p. S0925231219315966, Nov. 2019. 8, 19, 88

[43] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 354–370. 9, 15, 16, 32, 36, 40, 42, 43, 51, 62, 63, 71, 72

[44] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu, "Accurate Single Stage Detector Using Recurrent Rolling Convolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 752–760. 9, 16, 32, 60, 88, 90, 113

[45] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-Shot Refinement Neural Network for Object Detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 4203–4212. 9, 16, 32

[46] J. Fritsch, T. Kuhnl, and A. Geiger, "A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms," in *2013 16th International IEEE Conference on Intelligent Transportation Systems - (ITSC)*, Oct. 2013, pp. 1693–1700. 12

[47] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070. 12, 23

[48] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset," *CVPR Workshop on The Future of Datasets in Vision*, 2015. 12

[49] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016. 12

[50] R. Guzmán, J.-B. Hayet, and R. Klette, "Towards Ubiquitous Autonomous Driving: The CCSAD Dataset," in *Computer Analysis of Images and Patterns*, ser. Lecture Notes in Computer Science, G. Azzopardi and N. Petkov, Eds. Springer International Publishing, Sep. 2015, no. 9256, pp. 582–593. 12

[51] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 5000–5009. 12, 60

[52] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinksy, W. Jiang, and V. Shet, "Lyft Level 5 AV Dataset 2019," 2019. 12, 120

[53] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape Open Dataset for Autonomous Driving and its Application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019. 12, 120

[54] "Waymo Open Dataset: An autonomous driving dataset," https://www.waymo.com/open, 2019. 12, 13, 120

[55] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," *arXiv:1912.04838 [cs, stat]*, Dec. 2019. 12

[56] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. 14, 22

[57] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3D Object Proposals for Accurate Object Class Detection," in *Advances in Neural Information Processing Systems*, 2015, pp. 424–432. 14, 26

[58] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3D Object Proposals using Stereo Imagery for Accurate Object Class Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. Volume 40, no. Issue 5, pp. 1259–1272, May 2018. 14

[59] X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for Generic Object Detection," in *2013 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013, pp. 17–24. 14

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012. 14, 15, 27

[61] K. van de Sande, J. Uijlings, T. Gevers, and A. Smeulders, "Segmentation as Selective Search for Object Recognition," in *2011 IEEE International Conference on Computer Vision (ICCV)*, Nov. 2011, pp. 1879–1886. 14, 15

[62] S. Lazebnik, C. Schmid, and J. Ponce, "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, Jun. 2006, pp. 2169–2178. 14

[63] S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. D. Bona, A. Binder, C. Gehl, and V. Franc, "The SHOGUN Machine Learning Toolbox," *J. Mach. Learn. Res.*, vol. 11, pp. 1799–1802, 2010. 14

[64] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587. 15, 16

[65] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1440–1448. 15, 16, 37, 95

[66] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2013. 15, 16

[67] A. Shrivastava, A. Gupta, and R. Girshick, "Training Region-based Object Detectors with Online Hard Example Mining," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016. 15, 95

[68] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifier," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 15

[69] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 936–944. 15, 16

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016. 15

[71] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1–9. 15

[72] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," in *NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems*, Dec. 2016, pp. 379–387. 15, 23

[73] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016. 16

[74] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jul. 2017. 16

[75] ——, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767 [cs]*, Apr. 2018. 16

[76] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018. 16, 92, 94, 95

[77] J. Hosang, R. Benenson, P. Dollar, and B. Schiele, "What Makes for Effective Detection Proposals?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 814–830, Apr. 2016. 16

[78] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017. 16, 19

[79] R. Hartley, *Multiple View Geometry in Computer Vision Second Edition*, 2nd ed. Cambridge, UK ; New York: Cambridge University Press, Mar. 2004. 16, 40, 64

[80] F. Deng, X. Zhu, and J. Ren, "Object detection on panoramic images based on deep learning," in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, Apr. 2017, pp. 375–380. 17, 73

[81] H.-N. Hu, Y.-C. Lin, M.-Y. Liu, H.-T. Cheng, Y.-J. Chang, and M. Sun, "Deep 360 Pilot: Learning a Deep Agent for Piloting through 360° Sports Video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, May 2017, pp. 1396–1405. 17, 62, 73

[82] W.-S. Lai, Y. Huang, N. Joshi, C. Buehler, M.-H. Yang, and S. B. Kang, "Semantic-driven Generation of Hyperlapse from 360° Video," *IEEE Transactions on Visualization and Computer Graphics*, 2018. 17, 62

[83] Y.-C. Su and K. Grauman, "Flat2Sphere: Learning Spherical Convolution for Fast Features from 360° Imagery," *arXiv:1708.00919 [cs]*, Aug. 2017. 17

[84] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Apr. 2015. 18, 100

[85] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jun. 2014. 18

[86] S. Tripathi, Z. C. Lipton, S. Belongie, and T. Nguyen, "Context Matters: Refining Object Detection in Video with Recurrent Neural Networks," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, Sep. 2016, pp. 44.1–44.12. 18, 19, 120

[87] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014. 18

[88] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-NMS for Video Object Detection," *arXiv:1602.08465 [cs]*, Aug. 2016. 18

[89] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object Detection from Video Tubelets with Convolutional Neural Networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 817–825, Jun. 2016. 18, 19

[90] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang, "Object Detection in Videos with Tubelet Proposal Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 889–897, Jul. 2017. 18, 19

[91] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang, "T-CNN: Tubelets With Convolutional Neural Networks for Object Detection From Videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2896–2907, Oct. 2018. 18, 19

[92] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. 18, 21

[93] L. Galteri, L. Seidenari, M. Bertini, and A. D. Bimbo, "Spatio-Temporal Closed-Loop Object Detection," *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1253–1263, Mar. 2017. 18, 19

[94] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2017. 18, 19, 120

[95] Y. Lu, C. Lu, and C.-K. Tang, "Online Video Object Detection Using Association LSTM," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2363–2371. 18, 19, 21, 28

[96] X. Chen, Z. Wu, and J. Yu, "TSSD: Temporal Single-Shot Detector Based on Attention and LSTM," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1–9. 18, 19, 120

[97] X. Chen, J. Yu, and Z. Wu, "Temporally Identity-Aware SSD With Attentional LSTM," *IEEE Transactions on Cybernetics*, pp. 1–13, 2019. 18, 19

[98] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "MOTS: Multi-Object Tracking and Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019. 19, 23, 87, 122

[99] C. Zhang and J. Kim, "Modeling Long- and Short-Term Temporal Context for Video Object Detection," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 71–75. 19

[100] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 2758–2766. 19

[101] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, p. 246309, May 2008. 19, 50, 86

[102] X. Weng and K. Kitani, "A Baseline for 3D Multi-Object Tracking," *arXiv:1907.03961 [cs]*, Dec. 2019. 20, 85, 87, 111, 112, 113, 115, 117

[103] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: HybridBoosted multi-target tracker for crowded scene," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 2953–2960. 20

[104] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, Jun. 2008, pp. 1–8. 21, 34, 86

[105] H. Pirsiavash, D. Ramanan, and C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 1201–1208. 21, 34, 86

[106] W. Choi, "Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor," in *ICCV '15: Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015, pp. 3029–3037. 21

[107] P. Lenz, A. Geiger, and R. Urtasun, "FollowMe: Efficient Online Min-Cost Flow Tracking with Bounded Memory and Computation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 4364–4372. 21, 34, 51, 88

[108] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele, "A Multi-cut Formulation for Joint Segmentation and Tracking of Multiple Objects," *arXiv:1607.06317 [cs]*, Jul. 2016. 21

[109] R. Kumar, G. Charpiat, and M. Thonnat, "Multiple Object Tracking by Efficient Graph Partitioning," in *Computer Vision – ACCV 2014*, ser. Lecture Notes in Computer Science, D. Cremers, I. Reid, H. Saito, and M.-H. Yang, Eds. Cham: Springer International Publishing, 2015, pp. 445–460. 21

[110] A. Milan, K. Schindler, and S. Roth, "Multi-Target Tracking by Discrete-Continuous Energy Minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2054–2068, Oct. 2016. 21

[111] S. Tang, B. Andres, M. Andriluka, and B. Schiele, "Subgraph decomposition for multi-target tracking," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 5033–5041. 21

[112] ——, "Multi-person Tracking by Multicut and Deep Matching," in *Computer Vision – ECCV 2016 Workshops*, ser. Lecture Notes in Computer Science, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 100–111. 21

[113] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple People Tracking by Lifted Multicut and Person Re-identification," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 3701–3710. 21, 22

[114] H. Karunasekera, H. Wang, and H. Zhang, "Multiple Object Tracking With Attention to Appearance, Structure, Motion and Size," *IEEE Access*, vol. 7, pp. 104 423–104 434, 2019. 21

[115] A. Ošep, W. Mehner, M. Mathias, and B. Leibe, "Combined Image- and World-Space Tracking in Traffic Scenes," in *IEEE Int. Conference on Robotics and Automation*, 2017. 21

[116] S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna, "Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018. 21

[117] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large Displacement Optical Flow with Deep Matching," in *2013 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013, pp. 1385–1392. 21

[118] G. Wang, Y. Wang, H. Zhang, and R. Gu, "Exploit the Connectivity: Multi-Object Tracking with TrackletNet," in *Proceedings of the 27th ACM International Conference on Multimedia*, Oct. 2019. 21

[119] Y. Xiang, A. Alahi, and S. Savarese, "Learning to Track: Online Multi-object Tracking by Decision Making," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 4705–4713. 21

[120] S. Hong, T. You, S. Kwak, and B. Han, "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network," *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning*, vol. Volume 37, pp. 597–606, Jul. 2015. 21

[121] H. Nam and B. Han, "Learning Multi-Domain Convolutional Neural Networks for Visual Tracking," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016. 21

[122] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual Tracking with Fully Convolutional Networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 3119–3127. 21

[123] A. Milan, S. H. Rezatofighi, A. Dick, K. Schindler, and I. Reid, "Online Multi-target Tracking using Recurrent Neural Networks," *arXiv:1604.03635 [cs]*, Apr. 2016. 21

[124] A. Gaidon and E. Vig, "Online Domain Adaptation for Multi-Object Tracking," in *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, Sep. 2015, pp. 3.1–3.13. 21, 86

[125] H. Zhou, W. Ouyang, J. Cheng, X. Wang, and H. Li, "Deep Continuous Conditional Random Fields with Asymmetric Inter-object Constraints for Online Multi-object Tracking,"

IEEE Transactions on Circuits and Systems for Video Technology, vol. Volume 29, no. Issue 4, pp. 1011–1022, 2018. 21, 86

[126] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep Reinforcement Learning for Visual Object Tracking in Videos," *arXiv:1701.08936 [cs]*, Jan. 2017. 22

[127] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 300–311. 22

[128] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie, "Trajectory Factory: Tracklet Cleaving and Re-connection by Deep Siamese Bi-GRU for Multiple Object Tracking," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, Jul. 2018. 22, 28

[129] C. Kim, F. Li, and J. M. Rehg, "Multi-object Tracking with Neural Gating Using Bilinear LSTM," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 200–215. 22, 28

[130] E. Ristani and C. Tomasi, "Features for Multi-target Multi-camera Tracking and Re-identification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 6036–6046. 22

[131] Z. Zhang, J. Wu, X. Zhang, and C. Zhang, "Multi-Target, Multi-Camera Tracking by Hierarchical Clustering: Recent Progress on DukeMTMC Project," *arXiv:1712.09531 [cs]*, Dec. 2017. 22

[132] J. Luiten, T. Fischer, and B. Leibe, "Track to Reconstruct and Reconstruct to Track," *IEEE Robotics and Automation Letters*, vol. Volume 5, no. Issue 2, pp. 1803–1810, Apr. 2020. 22

[133] L. Leal-Taixé, C. C. Ferrer, and K. Schindler, "Learning by tracking: Siamese CNN for robust target association," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Dec. 2016. 23, 87

[134] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to Track and Track to Detect," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 3057–3065. 23, 87

[135] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation," in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, Sep. 2014, no. 8693, pp. 756–771. 23

[136] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 7–42, 2002. 23, 24, 34

[137] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, Sep. 1999, pp. 1150–1157 vol.2. 24

[138] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin Heidelberg, 2006, pp. 404–417. 24

[139] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, Jun. 2005, pp. 886–893 vol. 1. 24

[140] P. Pinggera, T. Breckon, and B. Horst, "On Cross-Spectral Stereo Matching using Dense Gradient Features," in *Proc. British Machine Vision Conference*, 2012, pp. 526.1–526.12. 24

[141] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information." *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–41, Feb. 2008. 24

[142] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-End Learning of Geometry and Context for Deep Stereo Regression," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 66–75. 24, 60

[143] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning Depth from Single Monocular Images," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. MIT Press, 2006, pp. 1161–1168. 24

[144] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D Scene Structure from a Single Still Image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, May 2009. 24

[145] L. Ladický, J. Shi, and M. Pollefeys, "Pulling Things out of Perspective," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 89–96. 25, 60

[146] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2024–2039, Oct. 2016. 25

[147] A. Atapour-Abarghouei and T. P. Breckon, "Real-Time Monocular Depth Estimation using Synthetic Data with Domain Adaptation," in *Proc. Computer Vision and Pattern Recognition*. IEEE, 2018. 25, 28, 60, 118

[148] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised Learning of Depth and Ego-Motion from Video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6612–6619. 25

[149] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-Supervised Deep Learning for Monocular Depth Map Prediction," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2215–2223. 25

[150] Y. Zhang, S. Song, P. Tan, and J. Xiao, "PanoContext: A Whole-Room 3D Context Model for Panoramic Scene Understanding," in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science. Springer, Cham, Sep. 2014, pp. 668–686. 25, 62

[151] J. Xu, B. Stenger, T. Kerola, and T. Tung, "Pano2CAD: Room Layout from a Single Panorama Image," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2017, pp. 354–362. 25, 62

[152] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-View 3D Object Detection Network for Autonomous Driving," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6526–6534, 2017. 26, 43, 45, 47, 61

[153] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D Object Detection for Autonomous Driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2147–2156. 26

[154] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR*, 2017, pp. 1827–1836. 26

[155] A. Kendall, Y. Gal, and R. Cipolla, "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018. 26, 42, 94, 100

[156] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. 27

[157] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015. 27

[158] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. 27

[159] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, Dec. 2010. 27

[160] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *2011 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2011, pp. 1521–1528. 28, 68

[161] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate Shift by Kernel Mean Matching," 2008. 28

[162] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning Transferable Features with Deep Adaptation Networks," in *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, vol. Volume 37, Jul. 2015, pp. 97–105. 28

[163] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science. Springer, Cham, Oct. 2016, pp. 597–613. 28

[164] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial Feature Learning," in *International Conference on Learning Representations (ICLR)*, vol. abs/1605.09782, 2017. 28

[165] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial Discriminative Domain Adaptation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2962–2971. 28

[166] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 2414–2423. 28

[167] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *European Conference on Computer Vision (ECCV)*. Springer, Sep. 2016. 28

[168] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images," in *ICML'16: Proceedings of the 33rd International Conference on International Conference on Machine Learning*, vol. Volume 48, Jun. 2016, pp. 1349–1357. 28

[169] V. Dumoulin, J. Shlens, and M. Kudlur, "A Learned Representation For Artistic Style," in *International Conference on Learning Representations (ICLR)*, Oct. 2016. 28

[170] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," *BMVC*, 2017. 28

[171] Mapillary, "Mapillary Research," https://research.mapillary.com/. 28, 62, 68, 75, 81, 115, 116

[172] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1725–1732. 28, 30

[173] Z. Wu, T. Yao, Y. Fu, and Y.-G. Jiang, "Deep Learning for Video Classification and Captioning," *Frontiers of Multimedia Research*, pp. 3–29, Dec. 2017. 28

[174] W. Wang, G. Chen, H. Chen, T. T. A. Dinh, J. Gao, B. C. Ooi, K.-L. Tan, S. Wang, and M. Zhang, "Deep Learning at Scale and at Ease," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 12, no. 4s, pp. 69:1–69:25, Nov. 2016. 30, 101, 113

[175] A. Gruslys, R. Munos, I. Danihelka, M. Lanctot, and A. Graves, "Memory-Efficient Backpropagation Through Time," in *NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems*, Dec. 2016, pp. 4132–4140. 30

[176] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training Deep Nets with Sublinear Memory Cost," *arXiv:1604.06174 [cs]*, Apr. 2016. 30

[177] I. Kokkinos, "UberNet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5454–5463. 30

[178] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 3309–3318. 30

[179] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, "Memory-Efficient Implementation of DenseNets," *arXiv:1707.06990 [cs]*, Jul. 2017. 30

[180] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "MegDet: A Large Mini-Batch Object Detector," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018. 30

[181] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-Driven 3D Voxel Patterns for Object Category Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1903–1911. 43

[182] L. Plotkin, "PyDriver: Entwicklung eines Frameworks für räumliche Detektion und Klassifikation von Objekten in Fahrzeugumgebung," Ph.D. dissertation, Karlsruhe Institute of Technology, Mar. 2015. 45, 47

[183] B. Li, T. Zhang, and T. Xia, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," *Robotics: Science and Systems*, Aug. 2016. 45, 47

[184] J. M. U. Vianney, S. Aich, and B. Liu, "Refinedmpl: Refined monocular pseudolidar for 3d object detection in autonomous driving," 2019. 47

[185] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 47

[186] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 47

[187] B. Li, "3D Fully Convolutional Network for Vehicle Detection in Point Cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep. 2017. 47

[188] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Nov. 2015. 60

[189] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan, "Cascade Residual Learning: A Two-stage Convolutional Neural Network for Stereo Matching," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 878–886. 60

[190] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, pp. 88–97, 2009. 60

[191] D. Kondermann, R. Nair, K. Honauer, K. Krispin, J. Andrulis, A. Brock, B. Güssefeld, M. Rahimimoghaddam, S. Hofmann, C. Brenner, and B. Jähne, "The HCI Benchmark Suite: Stereo and Flow Ground Truth with Uncertainties for Urban Autonomous Driving," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2016, pp. 19–28. 60

[192] Y. Fisher, "Berkeley Data Drive," http://data-bdd.berkeley.edu/, 2018. 60

[193] Austrian Institute of Technology, "WildDash Benchmark," http://www.wilddash.cc/, 2018. 60

[194] O. K. Hamilton and T. P. Breckon, "Generalized dynamic object removal for dense stereo vision based scene mapping using synthesised optical flow," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 3439–3443. 61

[195] A. González, D. Vázquez, A. M. López, and J. Amores, "On-Board Object Detection: Multicue, Multimodal, and Multiview Random Forest of Local Experts," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3980–3990, Nov. 2017. 61

[196] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, Jun. 2005, pp. 510–517 vol. 1. 61

[197] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art," *arXiv:1704.05519 [cs]*, Apr. 2017. 61

[198] Y.-C. Su, D. Jayaraman, and K. Grauman, "Pano2Vid: Automatic Cinematography for Watching 360° Videos," in *ACCV*, 2016. 62

[199] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys, "Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo," in *2014 2nd International Conference on 3D Vision*, vol. 1, Dec. 2014, pp. 57–64. 62

[200] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski, "Low-cost 360 Stereo Photography and Video Capture," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 148:1–148:12, Jul. 2017. 62

[201] H.-k. Chiu, A. Prioletti, J. Li, and J. Bohg, "Probabilistic 3D Multi-Object Tracking for Autonomous Driving," *arXiv:2001.05673 [cs]*, Jan. 2020. 86, 111, 112

[202] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as Points," *arXiv:1904.07850 [cs]*, Apr. 2019. 86, 87, 111, 112, 113, 115, 117

[203] "NVIDIA Apex," NVIDIA Corporation, Mar. 2020. 87, 101, 113

[204] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. García, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed Precision Training," *ICLR*, 2017. 87, 101, 113

[205] Y. Wu and K. He, "Group Normalization," *International Journal of Computer Vision*, Jul. 2019. 91, 92

[206] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *ICML'15: Proceedings of the 32nd International*

Conference on International Conference on Machine Learning, vol. Volume 37, Jul. 2015, pp. 448–456. 92

[207] I. Loshchilov and F. Hutter, "Fixing Weight Decay Regularization in Adam," *ArXiv*, vol. abs/1711.05101, 2018. 100

[208] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010. 100

[209] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection," *arXiv:1908.09492 [cs]*, Aug. 2019. 111, 112

[210] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast Encoders for Object Detection From Point Clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 12 689–12 697. 112

[211] A. Simonelli, S. R. R. Bulò, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling Monocular 3D Object Detection," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, May 2019. 112

[212] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving Policy Transfer via Modularity and Abstraction," in *CoRL*, 2018. 118

[213] A. Navarro, S. Genc, P. Rangarajan, R. Khalil, N. Goberville, J. F. Rojas, and Z. Asher, "Using Reinforcement Learning and Simulation to Develop Autonomous Vehicle Control Strategies," SAE International, Warrendale, PA, SAE Technical Paper 2020-01-0737, Apr. 2020. 118

[214] S. D. Khan and H. Ullah, "A survey of advances in vision-based vehicle re-identification," *Computer Vision and Image Understanding*, vol. 182, pp. 50–63, May 2019. 118

[215] J. Zhu, Y. Fang, H. Abu-Haimed, K.-C. Lien, D. Fu, and J. Gu, "Learning Object-specific Distance from a Monocular Image," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019. 119

[216] W. Bao, B. Xu, and Z. Chen, "MonoFENet: Monocular 3D Object Detection With Feature Enhancement Networks," *IEEE Transactions on Image Processing*, vol. 29, pp. 2753–2765, 2020. 119

[217] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo, "Learning Depth-Guided Convolutions for Monocular 3D Object Detection," *arXiv:1912.04799 [cs]*, Dec. 2019. 119

[218] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial Transformer Networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2017–2025. 120

[219] G. Payen de La Garanderie, A. Atapour Abarghouei, and T. P. Breckon, "Eliminating the Blind Spot: Adapting 3D Object Detection and Monocular Depth Estimation to 360° Panoramic Imagery," in *ECCV*, 2018. 120

[220] D. Wang, Y. He, Y. Liu, D. Li, S. Wu, Y. Qin, and Z. Xu, "3D Object Detection Algorithm for Panoramic Images With Multi-Scale Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 171 461–171 470, 2019. 120

[221] E. Plaut, E. B. Yaacov, and B. E. Shlomo, "Monocular 3D Object Detection in Cylindrical Images from Fisheye Cameras," *arXiv:2003.03759 [cs]*, Mar. 2020. 120

[222] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A Survey on 3D Object Detection Methods for Autonomous Driving Applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019. 120

[223] K. Yang, X. Hu, L. M. Bergasa, E. Romera, and K. Wang, "PASS: Panoramic Annular Semantic Segmentation," in *IEEE Transactions on Intelligent Transportation Systems*, 2019. 120

[224] K. Yang, X. Hu, L. M. Bergasa, E. Romera, X. Huang, D. Sun, and K. Wang, "Can we PASS beyond the Field of View? Panoramic Annular Semantic Segmentation for Real-World Surrounding Perception," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2019, pp. 446–453. 120

[225] J. Koutník, K. Greff, F. J. Gomez, and J. Schmidhuber, "A Clockwork RNN," in *ICML'14: Proceedings of the 31st International Conference on International Conference on Machine Learning*, vol. Volume 32. IEEE, Jun. 2014, pp. II–1863–II–1871. 121

[226] A. Mujika, F. Meier, and A. Steger, "Fast-Slow Recurrent Neural Networks," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5915–5924. 121

[227] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, A. P. Badia, K. M. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, Oct. 2016. 121

[228] A. Sauer, N. Savinov, and A. Geiger, "Conditional Affordance Learning for Driving in Urban Environments," in *CoRL*, 2018. 122

[229] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh, "VQA: Visual Question Answering," *arXiv:1505.00468 [cs]*, May 2015. 122