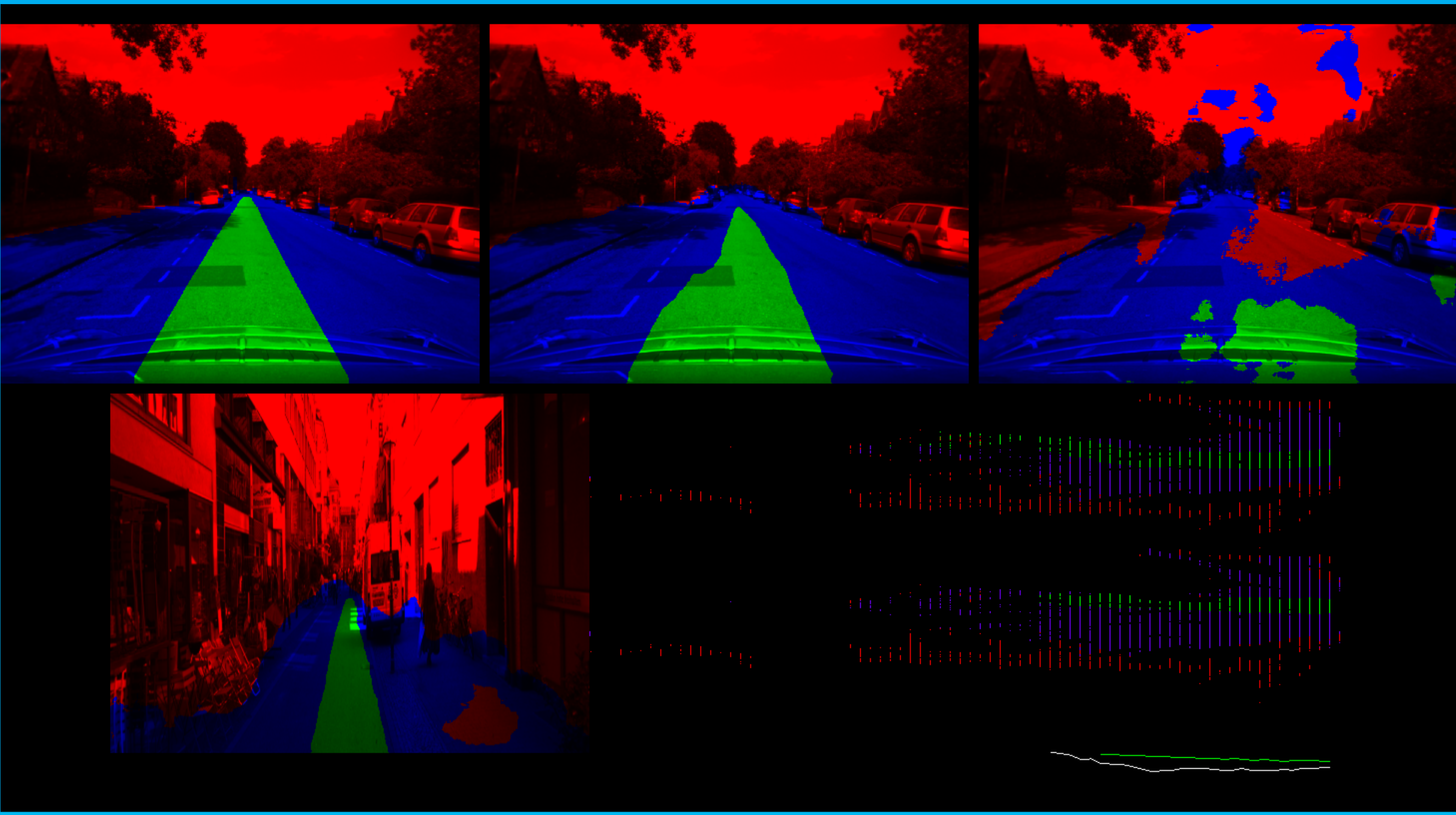
Deep segmentation of the drivable path of a self-driving vehicle using external data

Influence of domain shift factors and depth information

R.P.A. Bormans

Master thesis Geoscience & Remote Sensing



TUDelft

Deep segmentation of the drivable path of a self-driving vehicle using external data

Influence of domain shift factors and depth information

by

R.P.A. Bormans

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday November 26, 2018 at 2:00 PM.

Student number:	4384776	
Project duration:	October 2, 2017 – November 26, 2018	
Thesis committee:	Dr. R.C. Lindenbergh,	TU Delft, chairman, supervisor
	Prof. Dr. Ir. R.F. Hanssen,	TU Delft, supervisor
	Dr. J.F.P. Kooij,	TU Delft, external member
	Ir. F. Gaisser,	Robot Engineering Systems, daily supervisor

Abstract

Robot Care Systems (RCS) is involved in the development of the WEpod, an autonomous shuttle which can transfer up to six people. Based on a predefined map of the environment, the shuttle is able to navigate through mixed traffic its perception sensors such as camera, radar and lidar sensors. This study is acquired in collaboration with RCS and focuses on two parts: assessing the influence of different factors on the domain shift and assessing the importance of depth information in the transformation of scene understanding from image space to top view.

For the WEpod, or any self-driving vehicle to safely travel over the road and through traffic, it is important to understand road scenes that appear in our daily life. This scene understanding is the base for a successful and reliable future of autonomous vehicles. Deploying a Convolutional Neural Network (CNN) in order to execute the task of semantic segmentation is a typical approach to attain such understanding of the surroundings. However, when a CNN is trained on a certain source domain and then deployed on a different (target) domain, the network will often execute the task poorly. This is the result of differences between the source and target domain and is referred to as domain shift. Although it is a common problem, the factors that cause these differences are not yet fully explored. We filled this research gap with the investigation of ten different factors.

To explore these factors, a base network was generated by a two-step fine-tuning procedure on an existing convolutional neural network (SegNet) which is pretrained on the CityScapes dataset (dataset for semantic segmentation). Fine-tuning on part of the RobotCar dataset (road scenery dataset recorded in Oxford, UK) is followed by a second fine-tuning step. The latter is done on part of the KITTI dataset (road scenery dataset recorded throughout Germany). Experiments are conducted in order to obtain the influence of each factor on a successful domain adaptation (i.e. negligible domain shift). The influence of factors on the domain shift based on semantic segmentation is assessed by comparing the result of every factor to the result on the base network. Results consist of the F_1-measure and Jaccard index for drivable path segmentation and occupancy segmentation although the emphasis lies on the drivable path segmentation.

Significant positive influence on the estimation of drivable path for the WEpod domain was obtained when the ground truth labels only consisted of two labels (i.e. drivable path and non-drivable path) instead of three classes. This performance gain is signed by an increase of 8 percent points for both the IoU and the F_1 metric. Making all images intrinsically consistent, and thus removing all geometric differences between the camera sensors, resulted in a larger increase of performance metrics. Compared to the baseline, both the Jaccard index and F_1 metric increased with 10 percent points. The training order is a main contributor for domain adaptation with an increase of the IoU metric of 18 percent points and 20 percent points for the F_1 metric. This shows that the target domain (WEpod) is more closely related to RobotCar than to KITTI.

Although the investigation of different factors potentially can realise a better performance on the WEpod domain, understanding the environment in image space is not enough because path planning is utilised in top view. Hence, a transformation between image space and top view is needed to use the scene understanding based on semantic segmentation to the full extent. For this transformation, three setups are utilised with each a different form of depth information available: no depth information, ground truth depth information and estimated depth information. This approach gains insight into the relevance of depth information on the usability of scene understanding. The accuracy of top view transformations is measured in the form of four metrics: raw lateral error, scaled lateral error, overlap length and a count metric. The first metric is concerned with the raw difference between the estimated and ground truth trajectory. The second metric is a scaled form of the former. The overlap length measures to what extent the trajectory is estimated while the count metric takes into account if the estimated trajectory is present in top view.

From the experiments it is concluded that depth information plays an important role in the lateral sense of the trajectories while it is of less importance for the longitudinal length of the trajectory. It is also noticed that depth information does not necessarily needs to be dense. Moreover, the sparse but ground truth depth information leads to a better trajectory.

Acknowledgements

This report entails the MSc thesis for the graduation project of the master track Geoscience and Remote Sensing at the faculty of Civil Engineering and Geosciences of the Delft University of Technology. This study has been carried out in collaboration with Robot Care Systems, located in The Hague. I want to thank RCS for the great opportunity they offered me to broaden and deepening my knowledge and being involved in such a high-end project. Additionally, I want to thank all my (former) colleagues at RCS, helping me to process all the new matter via fruitful discussions. I also enjoyed the the times where we learned a lot on non-technical subjects.

I would like to express my appreciation to the members of my thesis committee starting with Ramon Hanssen for his critical view on my work which is summarised in this report. The second member I want to thank is Julian Kooij. Julian, thank you for your ideas and insights on the topic, both in the technical sense as in reporting style.

During the first part of my graduation, Fatemeh Karimi Nejadasl supported me on a daily basis and I want to thank her for the endless patience and ideas on the topic. For the last part of my graduation, Floris Gaisser was my daily supervisor. Floris, I want to thank you for your endless patience, new ideas on the topic and the tremendous help with writing this thesis; your feedback gave me helpful new insights in how to build a framework for this thesis.

I am very grateful to Roderik Lindenbergh who played an important role in the last year of my master. To begin with, you proposed RCS as possible company for a graduation project. Furthermore, as supervisor at the university, our meetings were always inspirational, keeping me focused on important issues. Your continuous enthusiasm is infectious and helped me throughout this thesis. I also want to thank you for the unbelievable opportunity you offered me, making it possible to visit the ISPRS Technical Commission II Symposium in Riva del Garda (Italy) and present my poster.

I want to thank my parents from the bottom of my heart for the unbelievable belief and pride you had in me. Thanks to your support, I had a stable base to build from. Additionally, I want to thank my friends for the numerous times of distraction, balancing the ratio between work and relaxation.

Finally, I would like to thank Jitske for her unconditional support and motivation. Jit, you were always there for me. You listened to my complaints and helped me through difficult situations related to both thesis and non-thesis obstacles. More than you probably realise, your words stimulated me and caused me to extend my boundaries which greatly contributed to this thesis.

Robbert Bormans
The Hague, September 2018

Contents

1 Introduction

In recent years, great progress has been shown in the fields of computer vision, machine learning and one of its applications, Autonomous Vehicles (AVs). This progress has led to a continuously growing interest in these fields, revealing new techniques and approaches, pushing its applicability increasingly further. In addition to these new advances, emerging interest in AVs is motivated by the potential of AVs itself.

The vast majority of traffic related fatalities is caused by human error [43], [48], [60]. Such errors can occur due to fatigue, distraction or simply a wrong interpretation of the traffic situation causing dangerous situations or collisions. Autonomous and intelligent vehicles have the potential to greatly reduce incidents induced by human error, simply by minimising the human inference or completely taking the human factor out of the process.

Another promising feature of autonomous vehicles is the ability of mobilising people who are unable to drive themselves. This could relate to elderly people who are not capable of driving anymore as well as people with disabilities or people who are not confident in the chaotic traffic situations that can occur. Another group that can relate to this benefit from autonomous vehicles are young people that are not allowed to drive a car. AVs will increase the action radius of these groups and therefore contribute to an increasing mobility in general.

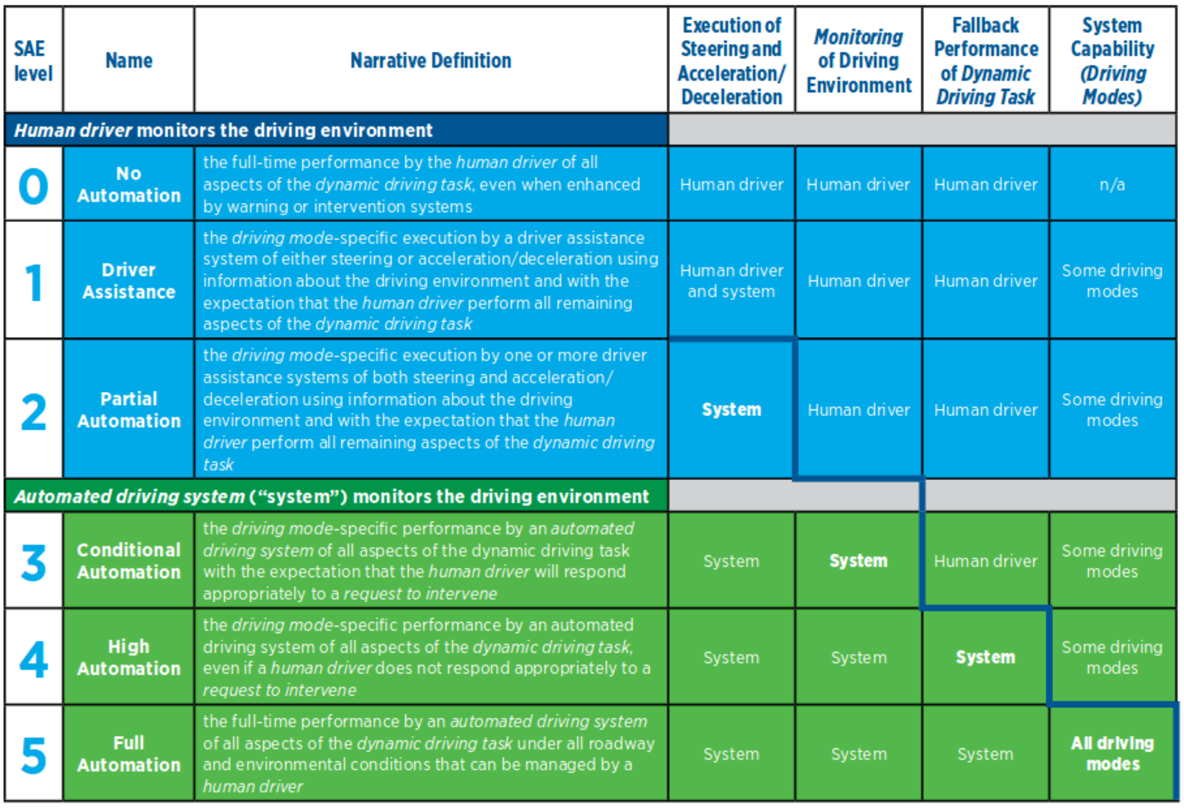
Furthermore, according to [51] the average round trip commuting time in the Netherlands is 47 minutes for the period 2005-2009 while in the same period it sums to 51 minutes in the United States. Currently, drivers have to be focused on the road and environment to avoid collisions. Autonomous vehicles can replace the need for human control and this will result in the possibility to use the commuting time in a productive or more relaxing manner which will result in a more efficient work-free time ratio.

The potential of AVs is also shown by two case studies in New York City and Singapore, conducted by [70]. Based on these case studies it is suggested that an Autonomous Mobility-on-Demand (AMoD) system would be much more affordable and convenient in order to access mobility compared to the traditional mobility models dependent on private vehicle ownership. To optimise performance, AMoD relies on highly automated vehicles. An overview of the levels of automation (ranging from 0 to 5) can be found in figure 1.1.

An intermediate step in the process towards fully or highly automated vehicles, are Advanced Driver-Assistance Systems (ADAS). These systems assist human drivers in the driving process, aiming to increase safety for both the driver as the other traffic participants. Examples of features of ADAS are collision avoidance system (preventing or reducing the severity of a collision) and adaptive cruise control (automatically adjusting vehicles speed to maintain a safe distance). Except the increase of general safety, ADAS are also used to increase the comfort of the driver (e.g. automatic parking).

In contrast to lower level automation features and vehicles, the WEpod is aiming for high automation (level 4). The WEpod is an autonomous shuttle, able to transfer up to six people. The shuttle is already deployed on trajectories in Wageningen and Delft. It has a maximum speed of 25 km/h and does not need a separate lane to drive in.

During autonomous driving the WEpod uses its sensors to sense its surroundings. The sensor suite of the WEpod consists of nine radar sensors, nine cameras and six four-plane lidars placed around the vehicle. The WEpod also has a GNSS receiver and IMU sensor. This sensor suite offers the ability to drive autonomously in a pre-built map of the area.



SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
Human driver **monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
Automated driving system **("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1.1: Levels of automation as defined by Society of Automotive Engineers (SAE) in J3016 [13].

Nevertheless, in order to utilise the potential of self-driving transportation in general, several barriers have to be overcome. These challenges are split up into different topics. Both on national and international level there are regulations that are not suited for (different ways of) autonomous driving. These need to be adjusted in order for utilise the potential of autonomous vehicles. Obviously, it is hard to change regulations and specifically to make new valid regulations which can cope with future situations. Therefore, it will take time for these changes take effect officially.

In addition to these legal challenges, there are a fair amount of practical challenges such as vehicle cost, adjustment of people's behaviour, insurance cases and privacy that need to be handled before full-autonomous transport can participate in our daily life.

There are still a lot of unsolved technical problems in order to obtain a full-autonomous system. These problems are all fields of research which need both funding and time to reach a practical solution. This thesis is part of the research challenge that has to be overcome.

Despite great developments, accidents with AVs (and vehicles with self-driving modes) made it clear that such systems are not satisfactory yet[1]. Therefore, research will continue in academia as well as in industry itself. One of the areas on which research is conducted is the understanding of road scenery which is one of the biggest challenges for AVs.

For a self-driving vehicle such as the WEpod to safely navigate on roads, it needs to *understand* road scenes that appear in our daily life. The environment that needs to be understood can be separated into several categories. The environment depends on the location which determines the infrastructure, but also traffic situations and traffic rules. Since the latter is part of different components in an AV such as behavioural planning (figure 1.2), these are not considered, whereas the infrastructure is an important feature which is included in this thesis.

This thesis focuses on the environmental understanding where it is safe to navigate to and where it is

[1]Examples:
`https://www.tesla.com/blog/tragic-loss?redirect=no`
`https://www.tesla.com/blog/update-last-week%E2%80%99s-accident`
`https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe`

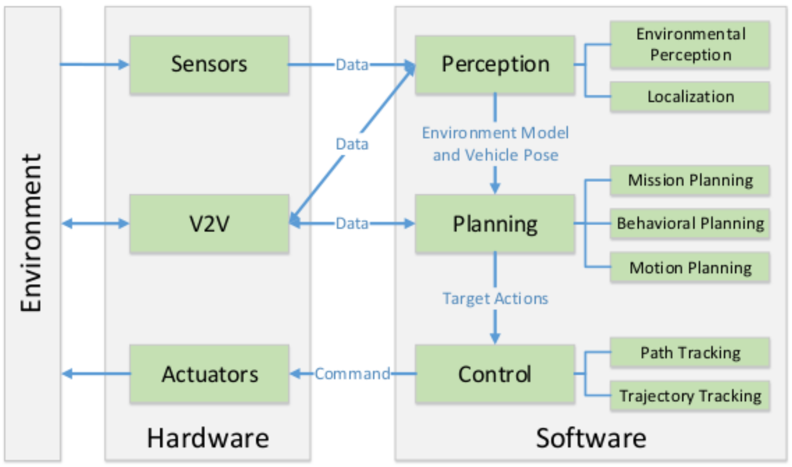


Figure 1.2: A general autonomous vehicle system overview. Obtained from [47].

not advisable (due to occupancies). Occupancies can occur due to a variety of reasons such as buildings, other cars and pedestrians. To the contrary, the path where a vehicle can safely navigate through traffic, is the trajectory where it does not collide with occupancies. But except for these obvious cases, it is also important a vehicle does not recognise the lane for the opposite traffic as "safe" by default.

One common way to achieve this level of understanding of the environment, is to use semantic segmentation. Semantic segmentation is the assignment of each pixel of an image to a semantically meaningful class. Large semantic datasets such as CityScapes [14], contain several groups where each group contains several classes. For CityScapes, this leads to 30 visual classes spread over eight groups. However, 19 classes from seven groups are used for creating the benchmark of this dataset. Examples of these groups are 'human', 'object' and 'construction' while examples of the classes are 'building', 'person', 'rider', 'pole', 'traffic sign'.

All these different classes and groups are not strictly necessary to make a relatively fair visualisation of the environment. For an AV it is important to know where it is possible to drive and where it is not allowed to drive. Therefore, a simple reconstruction of the environment can already be achieved by identifying three classes: occupancies, drivable path and unknown area.

The first goal of humans when driving around in traffic is to avoid any obstacle. The detection of obstacles is therefore an important aspect in order to realise autonomous vehicles. Two categories can be distinguished: static obstacles and dynamic obstacles. Static obstacles are ought not to move with buildings and trees being the most obvious examples. Dynamic obstacles can change direction and speed such as other traffic participants. Both static and dynamic obstacles should be detected with great accuracy to avoid dangerous situations or accidents.

As the name suggests, drivable path is a potential route which the vehicle may follow and is characterised by the outline of the total width the vehicle needs. Hence, a drivable path is more than only a trajectory line, enclosing an area with the width of the vehicle. The drivable path is not necessarily bounded to one "solution" because intersections can be thought of as an example of a combination of solutions. This path can be used for in-lane localisation.

Within every environment, there will be positions that cannot be classified as drivable path but is not perceived as obstacle, often due to the small size. Lanes (both in the same direction as opposite direction) are an example such a situation. Often curbstones, empty pavements and ditches also are included in this category, referred to as unknown area [5].

Traditionally, semantic segmentation of images were executed via semantic texton forests [55] or randomised decision trees [57]. The reason that these methods are outdated is due to the rise of Convolutional Neural Networks (CNNs) which have become a dominant player in the world of computer vision and semantic segmentation during recent years. CNNs showed to obtain high evaluation results concerning semantic segmentation of images, with increasing accuracy and decreasing computation time over the last years. However, it is commonly known that CNNs are *data hungry*. This means that a network needs a lot of (training) data in order make a reasonably good prediction.

Initially, no large amount of sensor data was available for the WEpod and stimulated by the fact that it is not desirable to mount (an expensive) lidar on every vehicle, already available (external) datasets are used to predict the drivable path, occupancies and unknown area for the WEpod.

Because only few training images containing full semantic ground truth are available weak labels are created for a subset of two large road scenery datasets, KITTI [20] and RobotCar [41]. These datasets not only contain the recorded image sequences but also laser, GPS and IMU data. The created labels are referred to as weak because they do not have clear boundaries for all three categories (occupancies, drivable path and unknown area). The quality of these labels depends to a certain extent on the sensor quality of the recording platform (both camera and lidar). By treating these labels as ground truth, it is possible to produce a vast amount of labels which will enable us to create a (large) set of training images within only a fraction of the manual annotation time. This comes at the cost of less precise labels. However, this decrease in quality is not necessarily a problem since the focus is put on the drivable path.

In general transfer learning can be very useful since it limits the required amount of training data and computational time needed to successfully train a CNN. However, one particular problem that occurs when a CNN is trained on a certain source domain (e.g. CityScapes) and then deployed on a different target domain (e.g. WEpod), the network will often execute the task (e.g. segmentation) poorly because of the differences between target and source domain (i.e. the domain shift). This limited ability of a CNN to adapt itself to new domains is a common problem of transfer learning.

The **first goal** of this thesis is to obtain an idea if and how several factors influence the domain shift from the KITTI and RobotCar domain towards the WEpod domain. Factors that influence the success of domain adaptation are identified and it is shown how they influence the result.

However, the WEpod is not able to process the segmented images directly because planning the trajectory generally occurs in a different space than image space. Hence, the segmented image should be converted towards the same space as where the path planning happens. This transformation between spaces is not straightforward since the segmentation misses one specific sort of information; depth. Depth is important in order to transform the predicted labels towards a top view, the space where path planning is executed. With depth information the segmentation can be used in the same space as path and therefore can act as a tool in order to understand the environment of the vehicle.

Hence, the **second goal** of this thesis focuses on achieving an usable format for semantic segmentation by highlighting differences between several setups.

1.1. Problem formulation and research objectives

The previous section briefly discussed the occasions and situations that lead to the the core of this research, which is motivated by two problems.

The first problem is based on the shift between datasets, obtained by different recording platforms. Due to the shift in domain for different datasets it is not possible to directly transfer knowledge, obtained from one dataset, to the other domain. Therefore, it is useful to obtain factors that are responsible for the shift between the domains.

The second problem is related to the transformation of scene understanding in image space towards a space that is useful for path planning: top view. As result of the project scope and research objectives, the following main research question is addressed in this study:

Is it possible to estimate and utilise drivable paths for the WEpod domain, solely based on data from different domains?

Several sub-questions assist to find a thorough answer to the main research question, stated above. These sub-questions are listed below:

- How can deep learning be used for estimating drivable paths and what is the role of domain adaptation?
  Drivable paths are an important aspect for autonomous vehicles as they play a role in localisation and path planning. Deep learning and moreover CNNs are an important tool in computer vision in order to predict these drivable paths. Insight in how these proposed paths are obtained and how domain adaptation can be utilised is a good base for this study.

- How to utilise the vast amount of data that is already available?
  A lot of data, often consisting of laser and camera data, is currently made available aiming to resolve different computer vision problems, including for the application of autonomous vehicles. However, there are different approaches in order to utilise this vast amount of data.

- What factors influence a successful domain adaptation for the task of semantic segmentation?
  Due to the domain shift resulting from the utilisation of existing datasets, results of semantic segmentation on the target domain are not satisfactory. It is beneficial to identify the factors that influence the performance by this domain shift.

- What is the importance of depth information in the transformation from image space to top view space?
  Autonomous vehicles are driving around in a 3D environment. However, part of scene understanding is done in image space, which is a 2D representation of this 3D environment. Since path planning is performed in a different 2D space the scene understanding needs to be transformed. Different setups are possible with varying depth information, resulting in different accuracies.

- How to evaluate drivable path proposals?
  In order to quantify the effect of certain parameters, it is important that well-defined evaluation metrics are used which will help to evaluate influences on the different classes.

1.2. Outline

Chapter 2 describes the theoretical background of this study which details various approaches and theories of semantic segmentation, transfer learning and the transformation of scene understanding to a usable format. The related work of these concepts will be helpful in understanding **chapter 3** which entails the methodology that is used during this research. This chapter consists of three parts: drivable path segmentation, domain adaptation and top view transformation. The first section will highlight the approach for drivable segmentation which are used in the second part of domain adaptation where the approach of different experiments is explained. The last section describes the method behind the transformation to top view in various experiments.

The setup used during the experiments is first discussed in **chapter 4**. This section is followed by a detailed description of the experiments and examination of the results. First this is done for the experiments of the domain shift factors while a third section handles the experiments of the top view transformation.

Discussion of the results, conclusions based on the results and recommendations for future research are reported in **chapter 5**.

2 Background

While chapter 1 sketched an overview of the project, this chapter reviews concepts which are used throughout this work more thoroughly. The chapter starts with a discussion about the observational data of the WEpod and different datasets are that used throughout this study in section 2.1. Further, this chapter entails (semantic) segmentation, obtaining an overview of different kinds of segmentation, discussing the pros and cons of various classes (section 2.2). Furthermore, different methods for segmentation are examined, making a clear distinction between hand-crafted methods and learned methods (section 2.3). Subsequently, transfer learning and more specifically domain adaptation is discussed (section 2.4). The relevance of the transformation from image space towards top view is explored (section 2.5). Finally, the chapter is summarised in section 2.6.

2.1. WEpod

The WEpod is a autonomous shuttle, capable of transferring up to six people. For the purpose of autonomous driving, the vehicle needs to observe and sense the surroundings with its sensors. Although the sensor suite differs per vehicle, these varieties are mostly based on the geometry of the sensors. Section 2.1.1 discusses the sensor suite which is utilised throughout this study. External datasets (i.e. not obtained by the WEpod) are an important part of this study and therefore explained in section 2.1.2.

2.1.1. Observational data

A number of sensors is used by an autonomous vehicle to safely drive through traffic. All sensors have their strengths and limitations. Although an autonomous vehicle has various sensors, only a part of the total sensor suite is used for this thesis and hence, only these sensors are briefly described below. Figure 2.1 gives a broader overview of the total sensor suite, mounted on autonomous vehicles.

Like the human eye, cameras are susceptible to adverse weather conditions and variations in lighting (incidence). However, it is the only technology that is able to capture texture, colour and contrast information. The high level of detail (which obviously varies with the cost of the camera) allow them to be the leading technology for classification. These features, combined with the increasing pixel resolution and decreasing costs, make camera sensors the "leading" sensors for Advanced Driver Assistance Systems (ADAS) and autonomous systems.

Light Detection And Ranging sensors, or lidar sensors in short, are capable of measuring the distance to any object nearby by simply calculating the time taken by a pulse of light to travel to that object and back to the sensor. These sensors therefore play an important role in creating a depth map which enables the vehicle to correctly estimate potential danger of objects.

Global Navigation Satellite System (GNSS) is the generic term for satellite navigation systems that provide autonomous geo-spatial positioning with global coverage while using different systems such as GPS, GLONASS, Galileo, Beidou and other regional systems are included. The advantage to having access to multiple navigation platforms is a higher accuracy and availability at different times and locations. The GNSS system will report the position of the receiver in the ellipsoid coordinate system (longitude, latitude and altitude). The quality of these measures can be affected by a range of elements such as urban canyoning.

An Inertial Measurement Unit (IMU) is an electronic device which detects linear acceleration using accelerometers and angular velocity using gyroscopes. Sometimes a magnetometer is included which measures the magnetic field intensity and is used as a heading reference. A generic configuration contains one accelerometer, gyroscope, and magnetometer per axis for each of the three vehicle axes: pitch, roll and yaw. Roll is referring to the rotation around the front-to-back axis, pitch is the rotation around the left-to-right axis and yaw (sometimes referred to as heading) is the rotation around the vertical (top-to-bottom) axis. The combination of GNSS data and IMU data allows one to reconstruct the driven path.

Inertial Navigation System (INS) is a system that processes the data collected by an (included) IMU in order to calculate relative position, orientation and velocity of the INS. Positions are retrieved through dead reckoning, which is the procedure that calculates the current position by adding the movement over elapsed time and course to the previously (known) position.

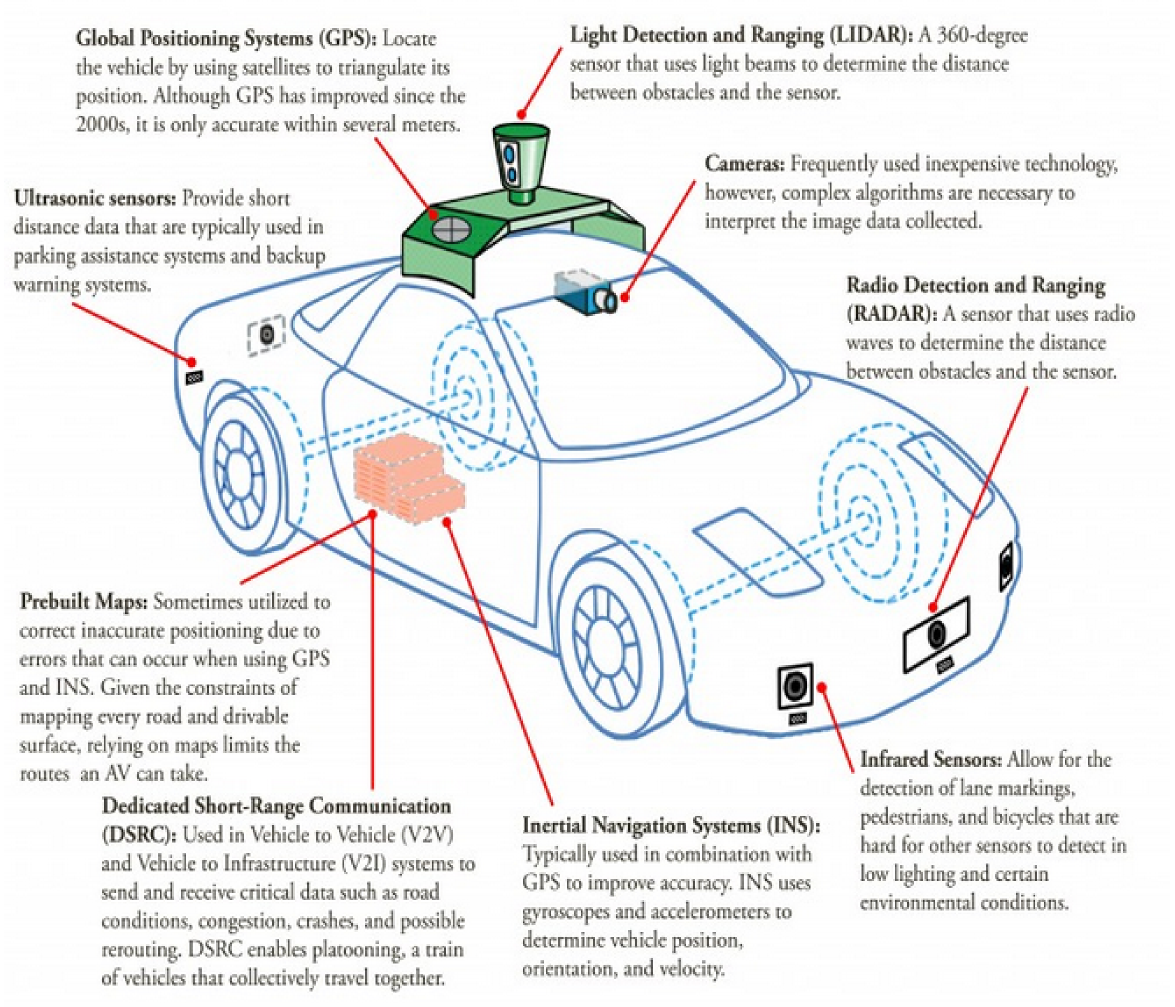

Global Positioning Systems (GPS): Locate the vehicle by using satellites to triangulate its position. Although GPS has improved since the 2000s, it is only accurate within several meters.
Light Detection and Ranging (LIDAR): A 360-degree sensor that uses light beams to determine the distance between obstacles and the sensor.
Cameras: Frequently used inexpensive technology, however, complex algorithms are necessary to interpret the image data collected.
Ultrasonic sensors: Provide short distance data that are typically used in parking assistance systems and backup warning systems.
Radio Detection and Ranging (RADAR): A sensor that uses radio waves to determine the distance between obstacles and the sensor.
Prebuilt Maps: Sometimes utilized to correct inaccurate positioning due to errors that can occur when using GPS and INS. Given the constraints of mapping every road and drivable surface, relying on maps limits the routes an AV can take.
Dedicated Short-Range Communication (DSRC): Used in Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) systems to send and receive critical data such as road conditions, congestion, crashes, and possible rerouting. DSRC enables platooning, a train of vehicles that collectively travel together.
Inertial Navigation Systems (INS): Typically used in combination with GPS to improve accuracy. INS uses gyroscopes and accelerometers to determine vehicle position, orientation, and velocity.
Infrared Sensors: Allow for the detection of lane markings, pedestrians, and bicycles that are hard for other sensors to detect in low lighting and certain environmental conditions.

Figure 2.1: Overview of the key sensors for an autonomous vehicle.[1]

2.1.2. Datasets

In order to create state-of-the-art algorithms and ensure they achieve good results, it is important to train on one particular set and evaluate on a different (test) set which ideally represents real-world challenges. This is the key role of datasets, providing a sample of real-world problems with ground truth to allow quantitative evaluation. On the basis of these evaluations, conclusions can be drawn about their constraints and abilities. In light of autonomous driving, the KITTI dataset (with tasks as depth prediction, object detection and tracking) and the CityScapes dataset (instance and pixel-wise segmentation) are challenging benchmarks that play a key role in research. The size of datasets has evolved over time from smaller datasets of several hundred images to huge datasets with thousands of images.

[1]Center for Sustainable Systems, University of Michigan. 2017. "Autonomous Vehicles Factsheet." Pub. No. CSS16-18

Many datasets aim for a large variety in the encountered road scenery. This results in a large variation of traffic situations and a large dissimilarity in traffic participants and thus the algorithm which is trained on such a dataset learns a lot of different representations of obstacles. These datasets do not take the challenges of environmental changes over time (i.e. seasonal changes) into account. [41] recorded images along with lidar data, GNSS data and Inertial Navigation System (INS) data by driving more than 1000 km over a period of one year. During this period (May 2014 to December 2015) one route in central Oxford (England) is covered twice a week. This driving scheme led to large variations in scene appearance due to illumination, weather and seasonal changes. Due to these long-term changes captured in the dataset, it is possible to develop algorithms that can handle these challenges.

Very recently, three enormous datasets have been released: Mapillary (May 2017), ApolloScape (March 2018) and BDD100K (May 2018) containing 25 000, 143 906 and 120 000 000 images respectively. Where ApolloScape does not have any diversity concerning cities, weather, times of day and scene types, Mapillary and BDD100K do contain all these varieties. However, ApolloScape and BDD100K were released too recently to be integrated in this work. Mapillary was not used because it lacks GPS and IMU data which is necessary to project the drivable path.

The generation of a dataset with ground truth labels is a time consuming task, especially when the labels need to be pixel-wise annotated as for semantic segmentation. This problem initiated research on ways to solve this problem by automating the process of labelling and alternative ways of data acquisition to avoid this problem. One alternative approach is to utilise synthetic data where ground truth on pixel-level can easily be acquired. However, for synthetic data there is a trade-off between the realism of the dataset and the flexibility of the data. In this area there are three main simulators: Grand Theft Auto V (GTA V), CAR Learning to Act (CARLA) [16] and The Open Racing Car Simulator (TORCS) [68]. GTA V has a very high level of realistic settings however, it is the least flexible among the three. TORCS is the least realistic but has high flexibility and CARLA is the most flexible simulator resulting in a less realistic setting than GTA V. However, despite the increasing effort for creating synthetic data it is unclear whether the scenes are realistic enough to replace real-world datasets. Creation of virtual content itself is time consuming. Where generation of ground truth labels are relatively easy to obtain for virtual data, the creation of the content itself is not, which makes the trade-off between real and synthetic data unclear. Hence, this thesis has been limited to real-world datasets. The datasets that are used throughout this thesis, are briefly mentioned below.

CityScapes

[14] tries to capture the complexity of real-world urban scenes by introducing the CityScapes dataset. This dataset comprises a benchmark suite and large amount of labelled images to semantically understand urban street scenes. The stereo video sequences of CityScapes are recorded in 50 different cities in Germany. 25 000 images are labelled from which 5 000 have high quality pixel-level annotations. These fine annotated images are divided into three categories: 2 975 training, 500 validation and 1 525 test images. The remaining 20 000 images have coarse annotations which are not partitioned into separate training, validation and test images as they only serve as additional training data.

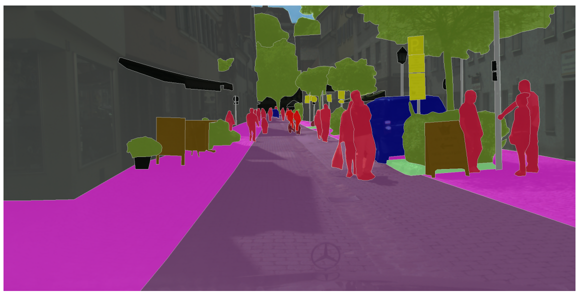


Figure 2.2: Example of a fine annotated image from CityScapes dataset, recorded in Tübingen.[2]

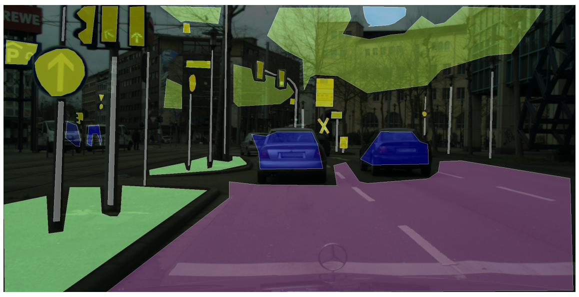


Figure 2.3: Example of a coarsely annotated image from CityScapes dataset, recorded in Saarbrücken.[3]

The images are recorded only during daytime and mostly with good or medium weather conditions. To achieve this recordings took place during spring, summer and fall. In total 19 semantic classes among seven groups are distinguished.

[3]https://www.cityscapes-dataset.com/examples/#fine-annotations
[3]https://www.cityscapes-dataset.com/examples/#coarse-annotations

KITTI

The raw dataset [20] consists of 6 hours recording throughout a variety of traffic environments. The recordings are obtained with high-resolution colour and stereo cameras, a Velodyne 3D laser scanner, GNSS data and IMU data. Although the weather conditions are similar across all recordings there is a large diversity in traffic situations that are encountered. The raw dataset is divided in four categories (city, road, residential and campus) based on the environment in which it is recorded. Image sequences that are calibrated, synchronised, timestamped and rectified are available.

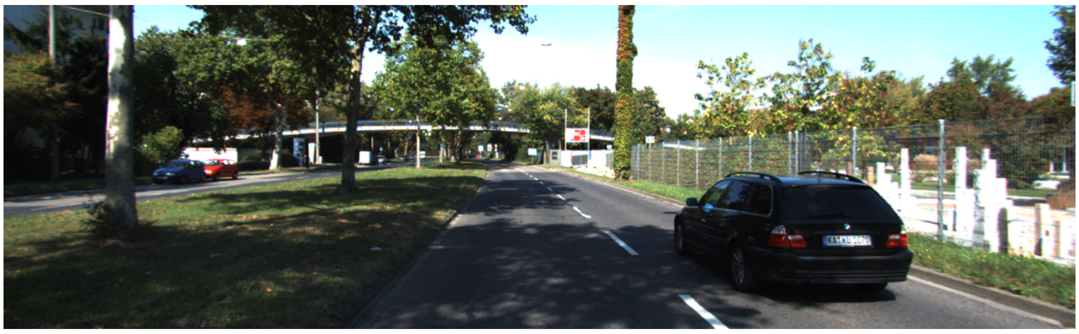


Figure 2.4: Image from the Raw KITTI dataset.

Oxford RobotCar

A different approach is followed by [41] who collected data over the period of a year leading to more diversity in appearance. During this period, one route in central Oxford is traversed twice a week collecting data for more than 1000 km. Although this is a notable distance, the novelty of this dataset is the variety in scene appearance since the sequences are recorded in different weather conditions during all four seasons.

The sensor suite of RobotCar consists of a stereo camera setup, two 2D lidars and one 3D lidar and GNSS/INS data. In contrast to the KITTI dataset, RobotCar consists only of raw data sequences without any ground truth labels.

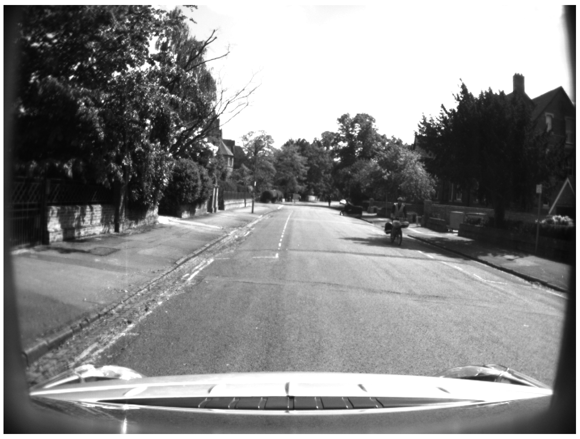


Figure 2.5: Raw image from the Oxford Robotcar dataset.

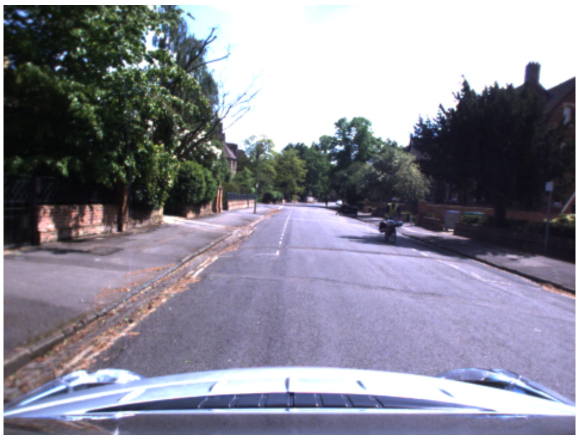


Figure 2.6: Undistorted and demosaiced image of Oxford Robotcar dataset.

WEpod

The WEpod is able to record its own dataset. For the goal of this project, two recordings are obtained. One dataset is recorded at Researchlab Automated Driving Delft (RADD) while the second dataset is recorded at the University in Wageningen.

The dataset at RADD is recorded in a private property setting which means that the image sequence does not resemble the general road scenery. The path is made out of concrete slabs and hence are not similar to road layouts. Furthermore, no pedestrians, cyclists or other traffic is present on the property during the recording. However, this dataset is used for testing the created algorithms. An example of this dataset is shown in figure 2.7.

The second dataset has parts which are recorded on public roads while other traffic participants are present. Despite these traffic participants, the recording does not show highly varying scenery. An example of this dataset is shown in figure 2.8.

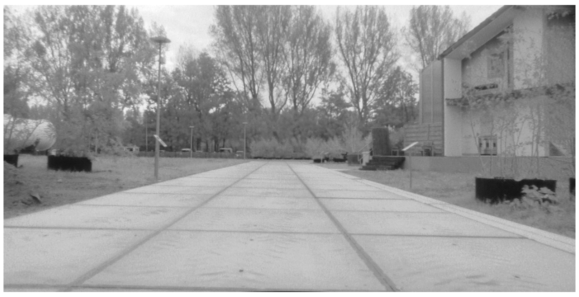


Figure 2.7: Raw image from WEpod dataset recorded at RADD (19-10-2017).

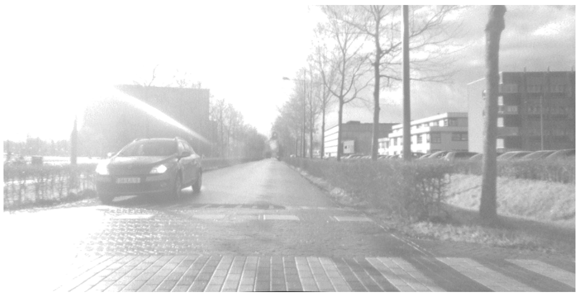


Figure 2.8: Raw image from WEpod dataset recorded at Wageningen.

2.1.3. Overview datasets

An overview of the aforementioned datasets is provided in table 2.1 based on number of training, validation and test images. Furthermore, the number of classes that are used during the training phase are noted.

The training set of KITTI contains 1060 images which are all recorded in a city-like environment and represents a variety of environmental scenes since different *drives* are used. The RobotCar training set exists of 2730 images from one recording

Test images from the KITTI domain are obtained from three different categories: city, road and residential. The road category and residential category are represented by 100 each while 50 images recorded in a city environment are present. The test set of RobotCar is composed of a subset of 250 images from one recording. Unfortunately, the WEpod is not as diverse as KITTI or RobotCar. Two options for the test set were the recording at RADD and the recording in Wageningen. Since the trajectory at RADD is far from representative of real road scenes, the trajectory at the University of Wageningen is chosen. From this set, a subset of 250 images is chosen to create the test set.

All test images are manually selected in order to achieve a good diversity throughout the total set and ensuring the correctness of the ground truth labels.

Dataset	# Training images	# Validation images	# Test images	# Classes
CityScapes	2975	500	n/a	11
KITTI	1060	114	250	3
RobotCar	2730	303	250	3
WEpod	493	54	250	3

Table 2.1: Overview of all utilised datasets throughout this study.

2.2. Image segmentation

Image segmentation is the process of partitioning an image into different regions based on the similarity of neighbouring pixels. This similarity can be based on a whole range of elements or parameters and can vary from colour to texture and shape [54]. The parameters on which segmentation is performed will differ per case since the objective of each segmentation may differ.

Image segmentation is used as a tool when an image is decomposed into separate parts for further analyses. This simplification is useful for different applications, such as medical imaging. Separating healthy tissues from abnormal tissues to correctly identify brain tumours is an example where image segmentation is helpful [2].

The sum of the segmented regions often covers the whole image. This is a requirement for semantic segmentation which is defined as the assignment of each pixel of an image to a semantically meaningful class. However, when not every pixel is classified, and hence the image is not completely covered by segments, it can be still referred to as segmentation; instance segmentation. An example is shown in figure 2.9.

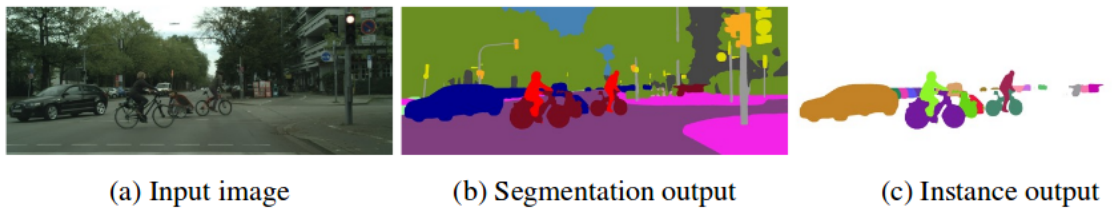


Figure 2.9: An example of semantic segmentation (b) and instance segmentation (c), clearly demonstrating the differences. Image modified from [32].

2.2.1. Semantic segmentation

As highlighted in section 2.2, image segmentation is the partition of an image into coherent parts without necessarily understanding what these parts represent. This is different from semantic segmentation which is the process of labelling each pixel in the image as one of the predefined classes and hence obtaining a certain understanding of the scene. Based on the predefined classes, AVs can obtain an understanding of its surroundings which is essential for enabling path planning and obstacle detection. Semantic segmentation can also help to play a role in visual odometry (determining position and orientation based on images) [1] by determining a higher uncertainty to dynamic obstacles than static obstacles by incorporating motion estimation.

[52] jointly uses semantic and geometric information in order to obtain robust visual localisation. The key of this attempt to apply localisation under extreme viewpoint and appearance changes is the method of learning robust 3D semantic descriptors. Hence, this approach shows that semantic segmentation can play an important and potentially essential role in solving the fundamental problem of visual localisation in a wide range of viewing conditions.

Since each pixel of the image needs to be assigned, it is obvious that manually creating a training set with semantically segmented images is a time consuming task. [14] mentions an annotation time (including quality control) of 1.5 hours for one image. High labour costs in combination with *data hungry* neural networks, result in an expensive combination.

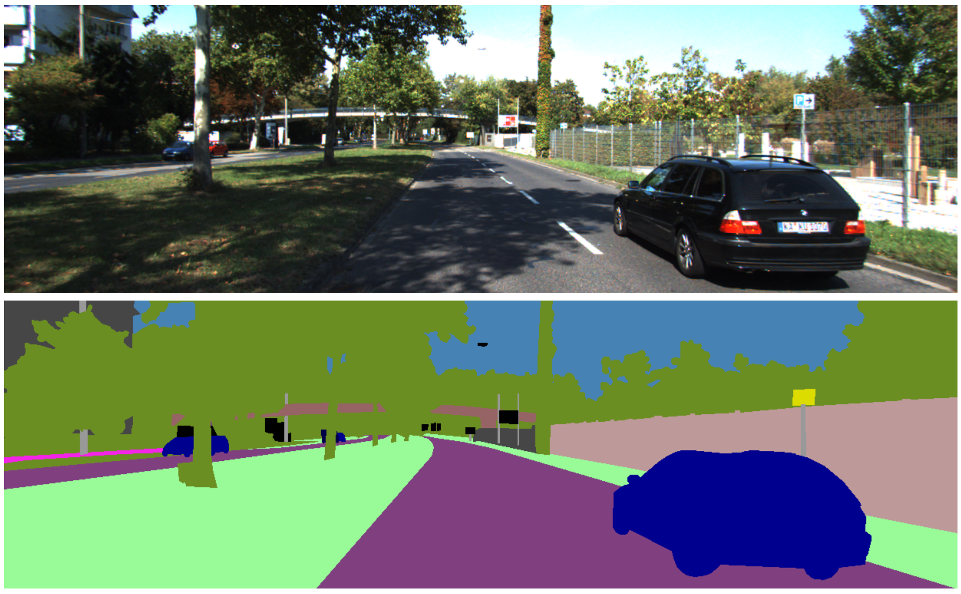


Figure 2.10: Image from the KITTI raw dataset with the pixel-wise semantic segmented ground truth.

2.2.2. Semantic classes

For semantic segmentation of images predefined classes are necessary. Obviously, these classes will relate to the application for which the images will be used for. An example of a fully semantic segmented image is shown in figure 2.10. The advantage of such a detailed segmented image is that it results in a high level of understanding of the scene. As mentioned in section 2.2.1, the disadvantage is the labour cost. It is possible to define less classes while still being able to give a useful scene understanding. Classes that are considered and differences with similar classes are highlighted in the following sections.

Road segmentation

Road segmentation is a vital part of scene understanding for pedestrian detection and autonomous driving [27]. It is important to know the position of the vehicle relative to the road markings or, when markings are absent, road layout. However, the challenge of road segmentation is to correctly segment in case of different occlusions such as cars and traffic participants. But also when the road type differs in some directions when roads have been broken open and repaired with a different material. It can also be challenging when weather conditions are bad for example when the road is still wet or when there are still puddles on the road. Except for rain, shadows can cause similar problems.

An approach based on publicly available OpenStreetMap (OSM) is exploited by [34]. Initial training labels are generated in an automated fashion by using noisy data from OSM. Hereafter, the labels are refined pixel-wise and used in a fully convolutional network in order to segment the road area. Due to this procedure annotation effort is reduced and hence supervised road segmentation algorithms are scalable.

Where [34] used OSM as initial training label, [30] use a deconvolutional network (DeconvNet) based on RGB and depth to provide initial labels. Both on pixel-level as on path-level, features are used to refine these initial labels. On patch-level, the road scene geometry feature is utilised while on pixel-level appearance and depth are used. The pixel-level features are obtained via a Random Forest classifier for each road scene.

A multi-task network is proposed by [63] who use a relatively straightforward encoder-decoder network to jointly deploy classification, detection and (road) segmentation. The approach is basic but efficient, performing with very low computation times. One encoder connects to three separate decoders which each perform their own task.

Lane segmentation

A lane segment is always a part of the road segmentation. For highways, lane segmentation is relatively easy since it is well defined by road markings but it becomes significantly harder to identify when travelling in urban areas since there are no predefined lanes and in some cases roads are not wide enough to contain two lanes.

[44] have divided the task of lane segmentation in two tasks/branches. The segmentation branch will create a binary lane mask while the embedding branch will produce an embedding per lane pixel in such a way that embeddings from the same lane are close together while different lane embeddings are located far from each other. After masking the background pixels using the binary mask from the segmentation branch, the lane embeddings are clustered together resulting in lanes separated by the marking splines.

Drivable path

Different from lane segmentation, where only the currently occupied lane is segmented, drivable path segmentation, highlights a proposed path where the vehicle is able to drive avoiding obstacles. Therefore, the drivable path will not occupy the whole lane and additionally is able to suggest a path around obstacles, changing lanes. When approaching intersections, it is important that the proposed path includes all possible options. This results in a manifold of solutions for multiple incoming roads [5].

In figure 2.11, the difference between the three previously mentioned segmentation classes is visualised. The upper element of the figure is the image as recorded. Below this image, the ground truth is representing three classes: road, non-road and unlabelled. In the bottom image, lane segmentation and drivable path are both shown to explicitly mention the difference between the two. In contrast to road segmentation, lane segmentation only labels the current lane where the recording platform is in. Drivable path ground truth contains the actual path the recording platform has followed. As can be seen in figure 2.11 (bottom), a lane change is ahead.

Free space estimation

The road is generally divided into two parts: occupied space and free space. Within the road segment, free space is defined as the space where a vehicle can freely move without colliding with other objects [39], [27]. Where the aim of road segmentation is to classify each pixel belonging to the road as such, the aim of free space segmentation is to classify each pixel where the vehicle can navigate without collisions as free space. Pixels beneath a car can be seen as a situation where the difference between the definition of these segments. For road segmentation the pixels beneath the car should be classified as road while for the free space segmentation, these pixels should be considered as obstacle (i.e. non-free).

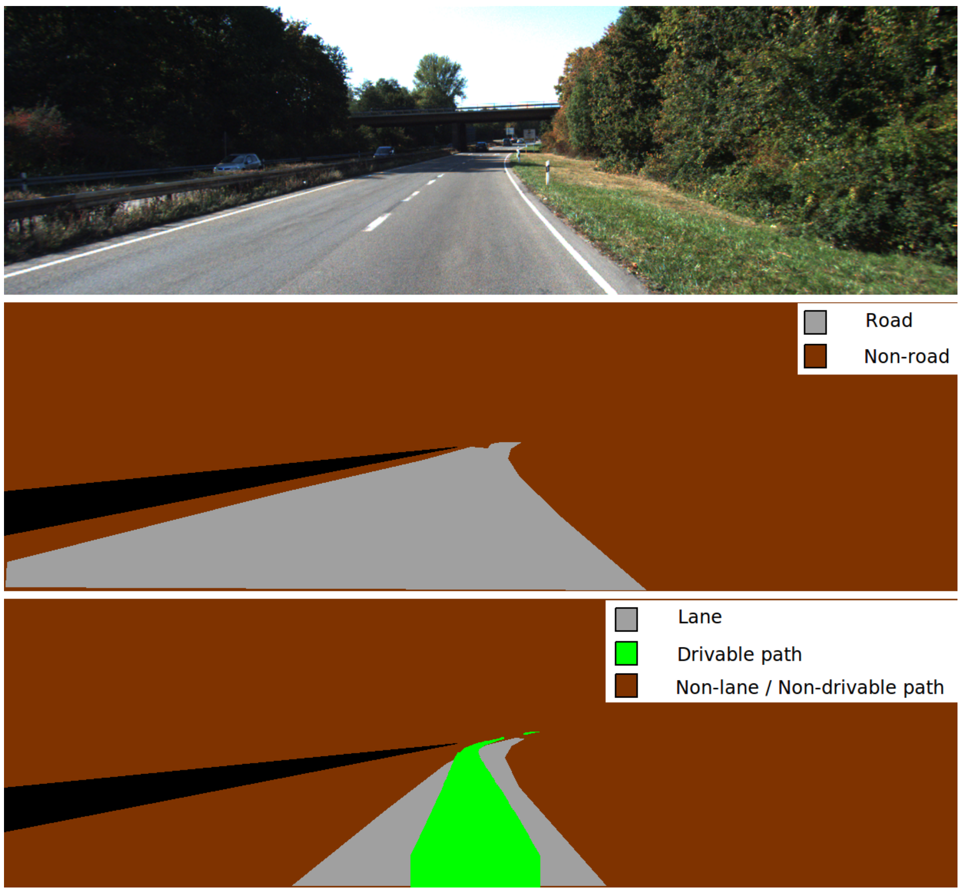


Figure 2.11: Visualisation of the difference between road segmentation (middle; grey area) and lane segmentation (lower; grey are) and drivable path (lower; green area). NOTE: *the lane change of the drivable path can clearly be seen in the lower image.*

Occupancies

Occupancies form a crucial class for autonomous vehicles. Because self-driving vehicles share the road with other traffic participants which are referred as occupancies from the view point of the autonomous vehicle. However, there are more occupancies for the self-driving car; this class is a collection of static obstacles such as parked cars and buildings and dynamic obstacles comprising cyclists, pedestrians and moving cars. Segmentation of the occupancy class is challenging in some setups due to spatially different lighting conditions (i.e. some areas are overexposed). Dependent on the location, obstacles can resemble the background which may lead to missing the obstacles [27]. Figure 2.12 shows an example image containing the occupancy label.

Unknown area

Unknown area specifies the space which is not included in the occupied space and drivable space. This definition results in no clear boundaries for this class because it is the remainder and has no specific meaning. In some cases, depending on the specifications of the lidar sensor, parts of occupancies can be included in the unknown area space. Furthermore, sidewalks are often classified as unknown area because of the complexity to extract them from the lidar data. Based on these examples of unknown area it is clear that there are some significant differences with free space.

2.3. Semantic segmentation methods

Different segmentation methods are categorised into two groups: hand-crafted methods and learned methods. Section 2.3.1 describes image segmentation techniques which are used before the rise of Deep Learning (DL), heavily depending on hand-crafted features combined with classifiers. Although these methods performed better over time, the performance of these approaches will always be bounded by the limited expressive power of the features. This restriction is resolved by deep learning since features are created by the system itself. The subsection of learned methods is related to this period of time (i.e. current time span),

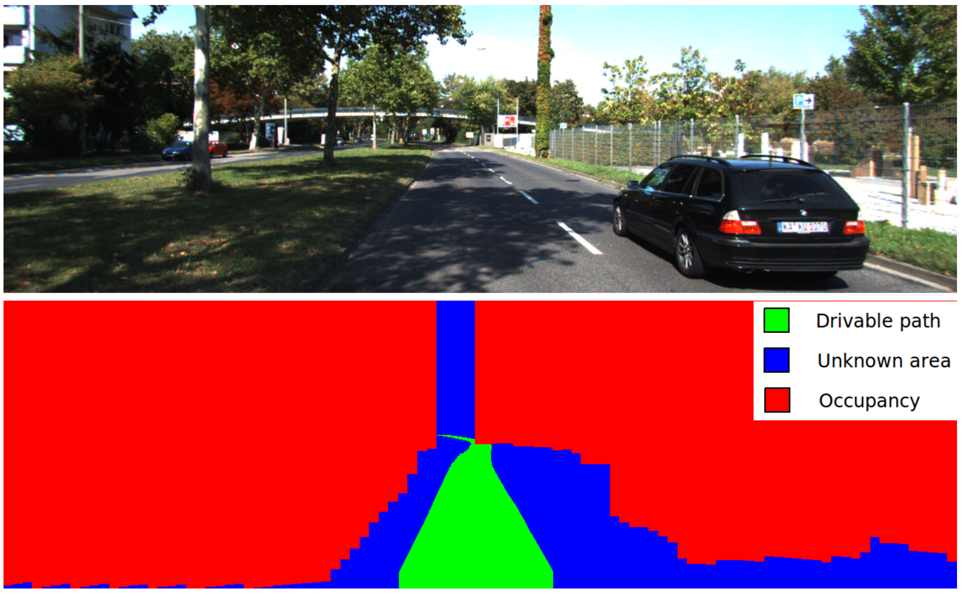


Figure 2.12: Example image of the KITTI raw dataset and the corresponding generated ground truth label. Red refers to occupancies, blue to unknown area and the drivable path is depicted as green.

mainly focusing on learned techniques to semantically segment road scenes.

2.3.1. Hand-crafted methods

Before the breakthrough of DL in 2012 [33], successful image segmentation was based on hand-crafted features combined with classifiers. Several machine learning approaches were used such as Support Vector Machines (SVM). Pixel-level colour and texture features are used as input for the SVM classifier by [66]. These features are manually extracted via a homogeneity model and a Gabor filter (a linear filter for texture analysis). Instead of pixel-level features, [19] use superpixel-level (local) features as the base for class segmentation. Aggregated histograms of the surrounding features are used as basis for the SVM classifier. Refinement of results is done by using Conditional Random Fields (CRFs) on the superpixel graph.

[42] defines image segmentation as the partitioning of an image into regions of coherent brightness and texture. Contours are treated via an existing framework while textons (clustering of responses of a linear filter bank) are introduced for analysing the texture. Hereafter, the cues of contour and texture differences are exploited simultaneously to find the correct partitions.

Textons are also exploited by [56] who use a texture-layout filter which is based on texture, layout and context information simultaneously. Additionally, boosting is used to achieve one-class classification and feature selection. Inclusion of this unary classification in a CRF is used to attain accurate object recognition and semantic segmentation.

In order to combine multiple classifiers [53] shows the capabilities of Random Forest, which is a collection of multiple decision trees. Furthermore, the combination of multiple features simultaneously as exploited via the random forest architecture leads to a further increase in performance when textons, colour, filter banks and Histogram of Oriented Gradients (HOG).

New low-level features are introduced in the form of semantic texton forests [55]. Semantic textons are ensembles of decision trees that act directly on image pixels. The bag of semantic textons is computed over local rectangular regions for segmentation. Textural and semantic context are included by semantic texton histograms and the region prior which is a class distribution computed as the average of leaf node class distributions of all trees in the forest.

A probabilistic framework proposed by [22] combines three individual models: a local classifier, regional label features and global label features, each encoding different information. By means of this framework contextual features are included via multi-scale CRFs with the aim of semantic segmentation of images.

2.3.2. Learned methods

Since interest in semantic segmentation increased due to the diverse applications such as autonomous driving, development on CNNs for semantic segmentation is a dynamic area of research resulting in impressive results. Modifying ILSVRC winning networks (e.g. VGG [59]) for image classification into a Fully Convolutional Network (FCN) for semantic segmentation resulted in state-of-the-art results on a variety of datasets [37]. SegNet [4] is a network consisting of two parts: an encoder and a decoder, where the encoder of SegNet is also based on VGG. In a few steps, this decoder will map the low resolution feature maps from the encoder to a final feature map by transferring pooling indices from the encoder. This helps to keep the spatial structure of the input (feature) map(s).

While both previously mentioned architectures are (partially) based on the VGG structure, [67] proposes a network based on a different idea. It is suggested to use a modified version of a Residual Network (ResNet [21]) as basis for semantic segmentation. Instead of constructing increasingly deeper networks, this modified ResNet is an ensemble of shallow networks. This version combines FCN and DeepLab ([10]) and outperforms very deep ResNets. Another extension of ResNets resulted in the densely connected convolutional networks (DenseNet [25]). [28] successfully extended DenseNet to be able to deploy the network for semantic segmentation instead of the original task of object recognition.

DeepLab uses Atrous (i.e. dilated [69]) Spatial Pyramid Pooling (ASPP). ASPP refers to the process of applying several parallel dilated convolutions with variable rates of dilation. These different rates are implemented in order to catch both *local* and *global* representations, classifying each pixel based on multi-scale features. PSPNet [72] performs several pooling operations on the feature map of the last convolutional layer with different pooling dimensions, aiming for the same goal, namely incorporating context information at different scales. PSPNet achieves higher evaluation metrics even without post-processing techniques such as CRFs which are used within DeepLab to boost performance.

To combine the advantages of an encoder-decoder architecture (identifying sharper boundaries) and spatial pyramid pooling module (multi-scale contextual information) [11] proposes a network which includes a spatial pyramid pooling module in the encoder while adding a simple decoder. This network, referred to as "DeepLabV3+" achieved top ranking on the CityScapes dataset.

2.4. Transfer learning

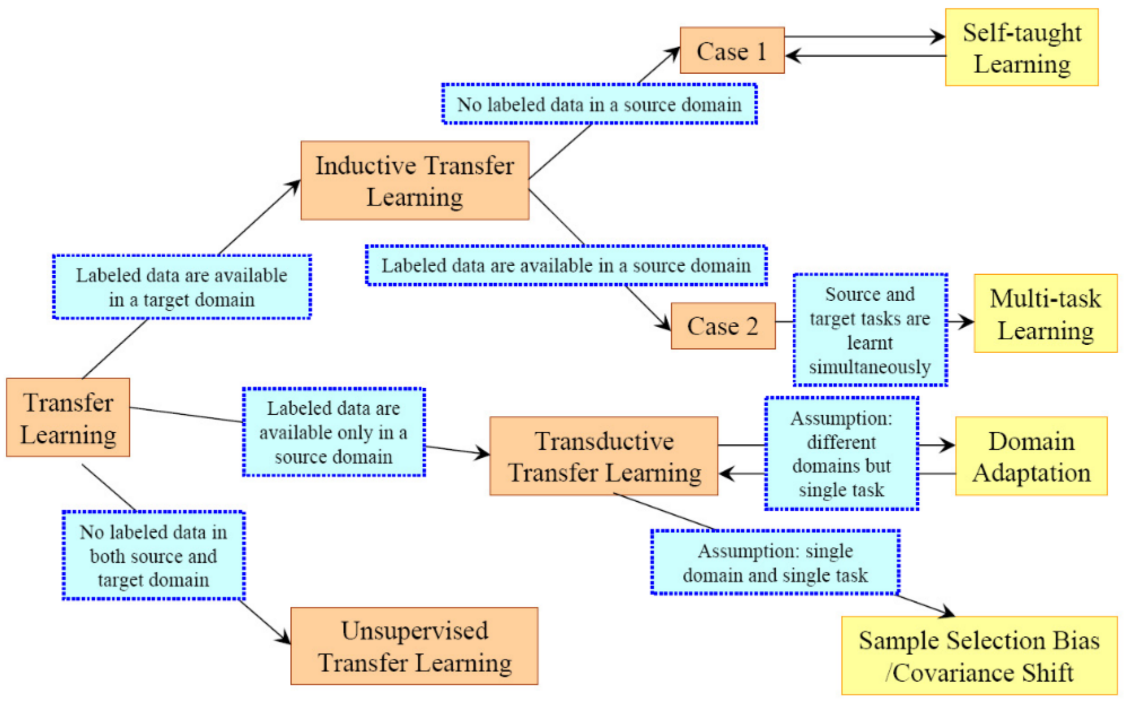

Case 1
Self-taught Learning
No labeled data in a source domain
Inductive Transfer Learning
Labeled data are available in a target domain
Labeled data are available in a source domain
Case 2
Source and target tasks are learnt simultaneously
Multi-task Learning
Transfer Learning
Labeled data are available only in a source domain
Transductive Transfer Learning
Assumption: different domains but single task
Domain Adaptation
No labeled data in both source and target domain
Assumption: single domain and single task
Unsupervised Transfer Learning
Sample Selection Bias /Covariance Shift

Figure 2.13: An overview of different settings for transfer learning. Image obtained from [46].

As described in the subsection 2.3.2, the results on several leading benchmarks for semantic segmentation are increasing to impressive accuracy. These state-of-the-art performances are achieved by the traditional approach of training a network on a certain source domain and subsequently testing on different data from the (same) source domain. For road scenes this implies that the test data is recorded with the same recording platform as the training set within the same environment. Additionally, all networks resulting in state-of-the-art benchmarks are trained for the task which eventually is deployed on the test data (i.e. the source task is the same as the target task). In short: state-of-the-art results are achieved under ideal circumstances. However, these ideal circumstances are often not available, implying the need for different approaches. One approach aiming to resolve this problem is referred to as Transfer Learning (TL) which is depicted in figure 2.13.

Domain Adaptation (DA) is a subset of TL and is concerned with the process of transferring knowledge obtained from a source domain to a different (target) domain while performing the same task. Based on two steps, [65] aims for simultaneous deep transfer across domains and tasks. First, the domain confusion is maximised which aims to align the domains and hence making the distributions of the domains as similar as possible. This is implemented by a domain confusion loss objective. Second, the proposed network transfers task information between domains by a cross entropy soft label loss.

The novelty of the approach of [38] lies in the implementation of classifier adaptation which is achieved by assuming that the source classifier differs from the target classifier by a residual function. Furthermore, feature adaptation is achieved by aligning the distributions across the different domains. While these approaches are successful, the implementation focuses on the image classification while this research is concerned with semantic segmentation.

Domain adaptation for semantic segmentation is initiated by [23] who considered the learning scenario of strong supervision in the source domain while no supervision was available in the target domain with the goal of semantically segmenting images. The implementation consists of global and category specific adaptation tested on three scenarios: synthetic to real adaptation, cross seasons adaptation, and cross city adaptation. While using a similar approach to tackle the problem [12] deploys the network only for cross city adaptation, however, in a more thorough way. Instead of cross city adaptation within the CityScapes dataset (as done by [23]) they use cross city adaptation for cities around the world.

Focused on unsupervised adaptation for the task of semantic segmentation [71] also exploits fully convolutional adversarial training. In more detail, the adaptation happens on both visual appearance and representation level. The former modifies both source and target images in such a way that they are more similar visually. The latter utilises the fully convolutional adversarial training method for a domain invariant representation.

Adversarial adaptation models applied in feature spaces discover domain invariant representations. Because adversarial adaptation models sometimes fail to capture pixel-level and low-level domain shift, [24] proposes a Cycle-consistent adversarial domain adaptation model (CyCADA). This model enforces cycle-consistency by taking five different losses into account: image-level GAN (Generative Adversarial Network) loss, feature level GAN loss, source cycle loss, semantic consistency loss and source task loss.

2.5. Top view transformation

It is essential for Autonomous Vehicles (AVs) to be able to relate measurements from different sensors for a variety of applications for the vehicle (e.g. tracking) [50]. Therefore, it is important to combine data from different sub spaces into one generic space. This generic space should contain all information and give an overview of the environment, hence the image space is not sufficient.

For autonomous driving applications, it is often sufficient to map the road surface in 2D (i.e. in top view) which allows for accurate localisation with respect to features on the road such as road markings or obstacles on the road surface [27]. The differences between top view space and image space are shown in figure 2.14.

Recent research has explored the field of road segmentation based on point clouds. [9] creates a network from scratch and thus perfectly matches the task. [8] uses a similar approach in order to generate the driving path. Both methods apply the network on the point cloud which is projected in top view. [40] even uses input maps projected in spherical coordinates because point clouds are least sparse in this projection.

Although these researches prove that (road) segmentation can be done in other views than image space, (semantic) segmentation still generally is deployed in the image space because image space contains a lot of details that vanish in top view and the image space is easily interpretable for humans.

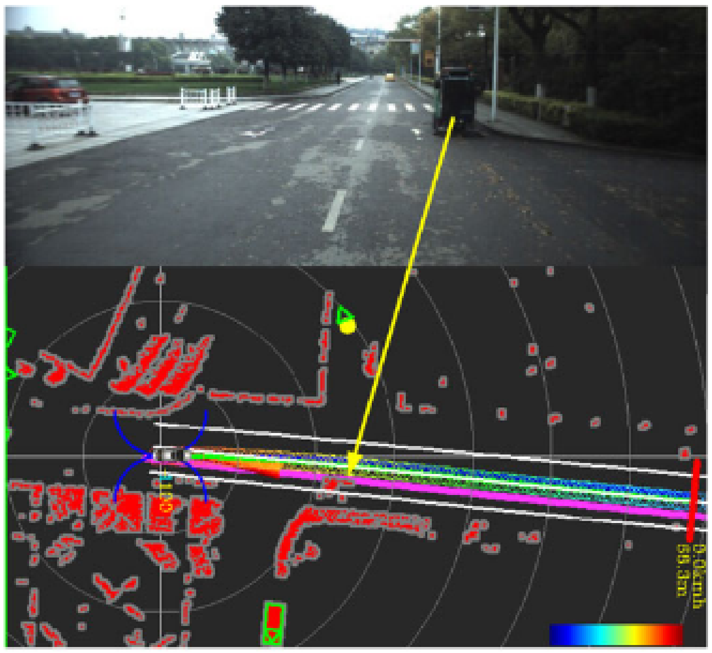


Figure 2.14: Image view (top) versus top view (bottom). Image obtained from [36].

2.5.1. Path planning

A detailed map of the environment is often necessary for path planning and navigation of an self driving vehicle [27]. The environment is not limited by the front view of the vehicle and occlusions are not allowed to play any role. It is necessary to have sight on the full 360°view around the vehicle. To obtain the whole environment of the vehicle in one map, a top view map can be created.

Many current methods for motion planning, path planning and route planning happen in top view. [45] gives a nice overview of motion planning and control techniques for autonomous vehicles.

The unified framework for manoeuvre classification and motion prediction created by [15] is projected in top view. In this way the framework is able to exploit multiple cues, such as the estimated motion of vehicles, an understanding of typical motion patterns of freeway traffic and inter-vehicle interaction.

[50] presents a framework for tracking multiple objects. The framework accepts object proposals from both camera and lidar to produce continuous object tracks. Throughout this work, two approaches for the projection towards top view are used. The effects of different camera setups, fusion schemes and 2D-to-3D projection schemes are examined.

Furthermore, [8] projects the driving path, generated by a fully convolutional neural network from the obtained point cloud, in a top view map. Although this resembles our goal of estimating drivable paths and projecting them into a usable space, there are some differences in the process. The mentioned research concerns with point clouds which can easily be transferred to top view while this work is solely dependent on the image segmentation making the transformation to top view different. Therefore, in this thesis, several possibilities are examined.

2.5.2. Relevance of semantic segmentation for autonomous vehicles

Semantic segmentation within the field of AVs mainly acts a tool for scene understanding but can also act as an helpful tool for localisation and visual odometry, as mentioned in the introduction of the chapter. Scene understanding viewed from the point of obstacles can be split up into two parts: dynamic obstacles (e.g. pedestrians and other traffic participants) and static obstacles (e.g. parked cars and buildings). Detecting dynamic obstacles is often done by obstacle detection, resulting in the type of obstacle and bounding boxes which indicate the extent of the obstacle. Static obstacles are mapped via semantic segmentation.

Semantic segmentation can aid in case of obstacle detection. Obstacles can falsely be detected by a variety of reasons. The silhouette of a person can be projected on the ground because of shading while cars can be reflected in the windows of buildings or buses (an example is shown in figure 2.15). In order to recognise these

[4]https://www.youtube.com/watch?v=LSX3qdy0dFg&t=2000s

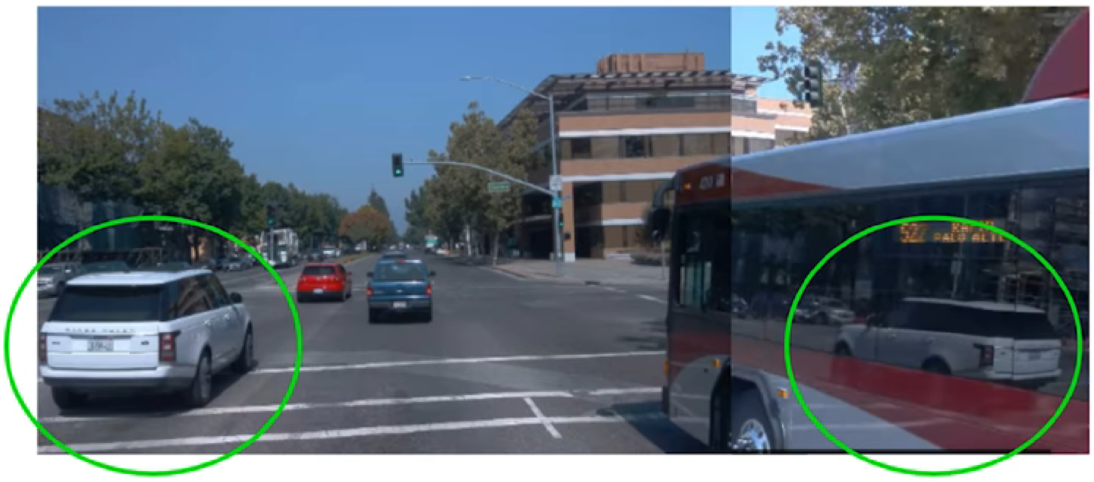


Figure 2.15: A situation where semantic segmentation could help object detection. Image obtained from a guest lecture of Sasha Arnoud (Director of Engineering, Waymo) for the MIT 6.S094 course[4].

false positives, a combination of systems should be applied. Semantic segmentation helps to understand the scene correctly and is able to prevent these false detections.

2.6. Summary

An autonomous vehicle, such as the WEpod, is equipped with a sensor suite. Except this sensor suite, and due to the increased attention for self-driving vehicles, datasets are made available to push the field to new levels. Both the sensor suite of the vehicle itself and datasets are necessary for the vehicle to understand its environment.

In addition to these elements, semantic segmentation of road scenes is a valuable approach to understand the surroundings of the vehicle. A broad variety of semantic classes can be useful for *understanding* the scene. However, a highly detailed understanding of the environment inherently leads to a large number of classes.

A distinction between two categories is made when evaluating the possible semantic segmentation methods. The first category contains methods that are based on hand-crafted features. These hand-crafted features are not required for the second category which is based on learned methods, mainly convolutional neural networks. The latter achieves better performance since the recent development of these methods.

One common disadvantage of these learned methods is the amount of training (both data and time) that is needed in order to achieve satisfactory results. Transfer learning is one approach to avoid this problem. Transfer learning is able to transfer "knowledge" from one problem consisting of a certain (source) task in a certain (source) domain to a different (target) task and/or a different (target) domain. This reduces the need for training data since advantage is taken from already existing knowledge. In this study, domain adaptation will be utilised which is the transfer of knowledge when source and target task are the same but deployed in a target domain that differs from the source domain.

To benefit from scene understanding in image space (i.e. semantic segmentation is deployed on image sequences), a conversion to another space should be applied. This transformation is necessary because path planning and higher level decisions are made in top view or a similar space.

3 Methodology

After discussing related work about semantic segmentation methods, different transfer learning approaches and the importance of the transformation to top view in chapter 2, the methodology used for three aspects is discussed in this chapter. The approach of weakly-supervised learning and the generation of labels is described in section 3.1. Then, the methodology behind the examination of the influence of several domain shift factors for semantic segmentation is discussed in 3.2. Different setups of the transformation to top view are described in section 3.3. The last section of this chapter contains a summary.

3.1. Drivable path segmentation

This section is divided into three parts. The first subsection will zoom from a broad view of learning methods towards the conducted learning method. The second subsection will examine this learning approach into detail while the last subsection briefly explores the implemented network for the learning approach.

3.1.1. Learning methods

Data is essential for learning. Except obtaining data, it is also key how to utilise and organise the data in such a way that it is optimally used to learn. In general data is subdivided into three sets, independent of learning method:

- **Training set:** A subset of the data which is used to fit the model or network. During training, model parameters are adjusted such that they fit this set of data.
- **Validation set:** A selection of the data which is used to estimate the prediction error for model selection or hyperparameter (i.e. parameters that cannot directly be learned during training) tuning.
- **Test set:** Sample of the data which is used for assessment of the generalisation error of the finally chosen model and/or (hyper) parameters.

It is important to create a separate test and validation set because the error estimates of the final model and hyperparameters on validation data will be biased. It will show better evaluation metrics than the true metrics since the validation set is containing data that is used to select the final model and tune the final hyperparameters. Because of this reason, it is not "allowed" to tune the model based on the test set.

Learning techniques are categorised in three major categories [6]: supervised learning, unsupervised learning and reinforcement learning. These categories are briefly explained in the sections below. Additionally, weakly-supervised learning is highlighted since this technique plays an important role in this work.

Supervised learning

Supervised learning is a method where the training set contains input data (e.g. images) together with their corresponding output vectors (in case of semantic segmentation this equals the the semantic label of the image). If the desired output consists of one or more continuous variables, the learning task is referred to as regression. However, when the task is to assign each input vector to one category, out of a finite number of categories, it is called classification.

Unsupervised learning

For unsupervised learning, in contrast to supervised learning, the training set only consists of input vectors. No corresponding output vectors are available. Unsupervised learning can be used for a range of applications: clustering (grouping similar data points), density estimation (determination of distribution of input data) and visualisation (reduce the dimensions of the input vector) [6].

Reinforcement learning

To complete the major learning categories, reinforcement learning is briefly discussed. The goal of reinforcement learning [61] is to learn a task consisting of a series of inputs with a delayed result. The learning technique learns to find suitable actions that will complete the task. It differs from supervised learning by the fact that it does not have the desired (optimal) output for very input, but only the desired end goal after a series of inputs. However, by iterating it can find a (better) solution which results in a higher reward. This iteration process can be seen as a sequence of states and actions which result in interaction with its environment [6]. The most well-known example of reinforcement learning is AlphaGo [58], the first program to defeat a world champion in the game of Go.

Weakly-supervised learning

Although current supervised learning techniques are able to obtain high accuracy for a whole range of tasks, it often is difficult to obtain or create strong labels to act as ground truth. Hence, weak supervision is a solution to this problem. There are different types of weak supervision possible [73]:

- **Incomplete supervision:** the training set only partially has ground truth labels
- **Inexact supervision:** the training set only has coarse labels
- **Inaccurate supervision:** the training set contains labels which are not always equal to the ground truth

Incomplete supervision is the learning environment where only a part of the total training set contains ground truth labels. This is often referred to as semi-supervised learning and will not be considered in this work. The other two types of weak supervision are very similar: inexact and inaccurate supervision. Inexact supervision is the approach where ground truth labels are available for the complete training set but the labels are only coarsely annotated (e.g. figure 2.3). For inaccurate supervision, the training set also consists of ground truth labels corresponding to every image of this training set. This ground truth, however, can contain small inaccuracies in the labels. In other words, pixels in the ground truth label classified as class X can in reality contain (small) parts of class Y. In this thesis, when referring to weakly-supervised learning, inexact supervision is intended. Therefore weakly-supervised learning is similar to supervised learning but instead of comparing the output vectors to ground truth data, it compares every output vector to weak labels that act as ground truth.

Self-supervised learning is the process of using the output of the model or network as a proxy for the desired output vector. Basically the structure of the data is used as a supervisory signal (which relates to unsupervised learning since no (manual) annotations are used).

3.1.2. Weakly-supervised segmentation

The weak labels are created in an automated fashion, adopted from [5]. This three-step process starts with the projection of the drivable path in the image. This is based on the assumption that the trajectory taken by the recording platform is the ground truth trajectory. The drivable path refers to the outermost points of the contact of the tires with the ground. These ground contact points are projected because the positions are known via the sensor setup (i.e. dimensions of the vehicle with sensor placement) of the recording platform and the GPS and IMU data. From the sensor setup, the dimensions from the contact point of the front wheels with the ground to the camera (with respect to the camera frame) is used.

The projection of points in the camera frame to the image plane is achieved by the pinhole camera model as visualised in figure 3.1. In this model, a scene view is formed by projecting 3D points into the image plane using a perspective transformation. This perspective transformation is specified as:

$$\begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = K \cdot \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix}, \tag{3.1}$$

where x_{cam}, y_{cam} and z_{cam} are the coordinates in camera frame while u and v are the corresponding image coordinates. In equation 3.1, s is a scale parameter. K represents the camera matrix and is expressed as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.2}$$

where f_x and f_y are the focal lengths in x and y direction measured in pixel units. In a true pinhole camera, they are equal to each other. c_x and c_y are referring to the principal point offset, also noted in pixel units. Because K is only concerned with the transformation between the camera coordinates and the image coordinates, the absolute camera dimensions are irrelevant. Using pixel units for focal length and principal point offset allows to represent the relative dimensions of the camera, namely, the film's position relative to its size in pixels.

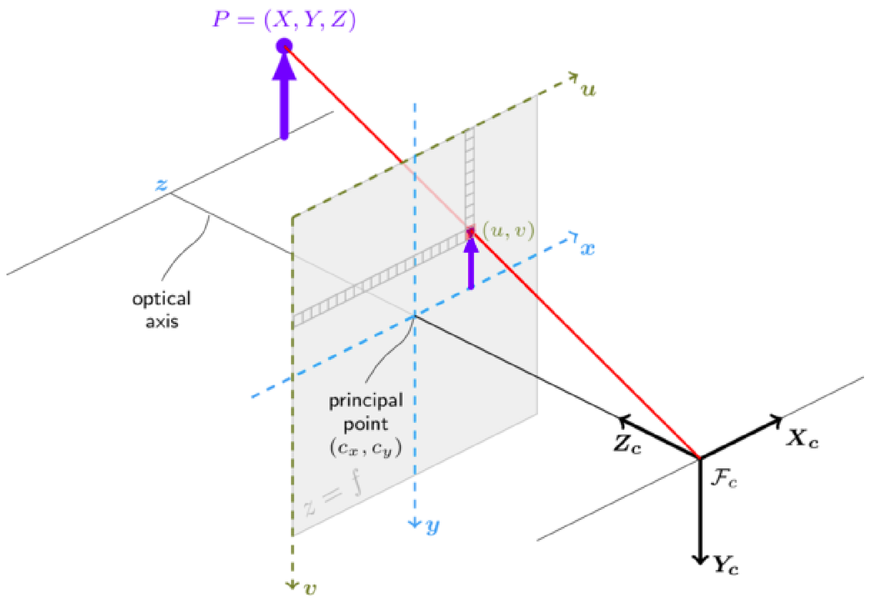


Figure 3.1: Visualisation of the pinhole camera model[1].

Ground scatter removal

The point cloud obtained by the laser scanner contains, besides the obstacles, also the ground plane of the road in front of the vehicle. Obviously, this ground scatter will not act as an obstacle for the vehicle and should thus be removed from the point cloud. Hence, after this removal the point cloud will only contain points which correspond with an obstacle for which it is important that the vehicle should avoid.

The ground points are removed from the point cloud based on GNSS data, IMU data and the external dimensions of the recording platform (i.e. the height of the laser scanner above the road). For this explanation, the time stamp which is currently considered is referred to as t_0 while the image during this time stamp is equal to the *base image* and denoted as i_0. The position of the IMU (the pose) at a certain time stamp is denoted analogous to this time stamp (i.e. IMU position, or pose, at time stamp t_0 is referred to as IMU_0). Time stamps ahead of this base, called future time stamps, are denoted as t_1, t_2,... while the notation of the pose follows the same trend (IMU_0, IMU_1,...). It is possible to visualise future poses in the base image via the GNSS data.

Basically, the point cloud belonging to time stamp t_0 is subdivided into different areas parallel to the side-to-side line of the vehicle. These areas are enclosed by pose IMU_k to IMU_{k+1} with k ranging from 0 to 50. For every area, points from the point cloud are removed when the point is less than 10 cm above the ground based on the height of the laser scanner and the position of the vehicle (i.e. IMU) relative to the time stamp

[1]Image obtained from: `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?#`

t_0. In this manner, the slope of the road ahead is accounted for.

After the ground scatter is filtered from the point cloud, the resulting point cloud is transformed to image coordinates. Then, the base image is virtually divided in 100 vertical bins. For each bin, all pixels above the lowest projected laser point are labelled as obstacle in that specific bin. When there is no laser point present in a bin, no pixels in that bin are labelled as obstacle. In case a pixel obtains labels both as drivable path and obstacle, the obstacle label is superimposed over the drivable path [7].

As third and last step, pixels without a label at this point will be labelled as 'unknown area'. These areas often consist of sidewalks, road and depending on the lidar (4-beam; RobotCar and WEpod) sometimes will include small areas of other vehicles and buildings. This characterises that we are using weak labels. Figure 3.2 shows example images from KITTI, RobotCar and WEpod with the corresponding weak labels.

Because this method is not applied through an extensive set of work (yet), there is no benchmark available.

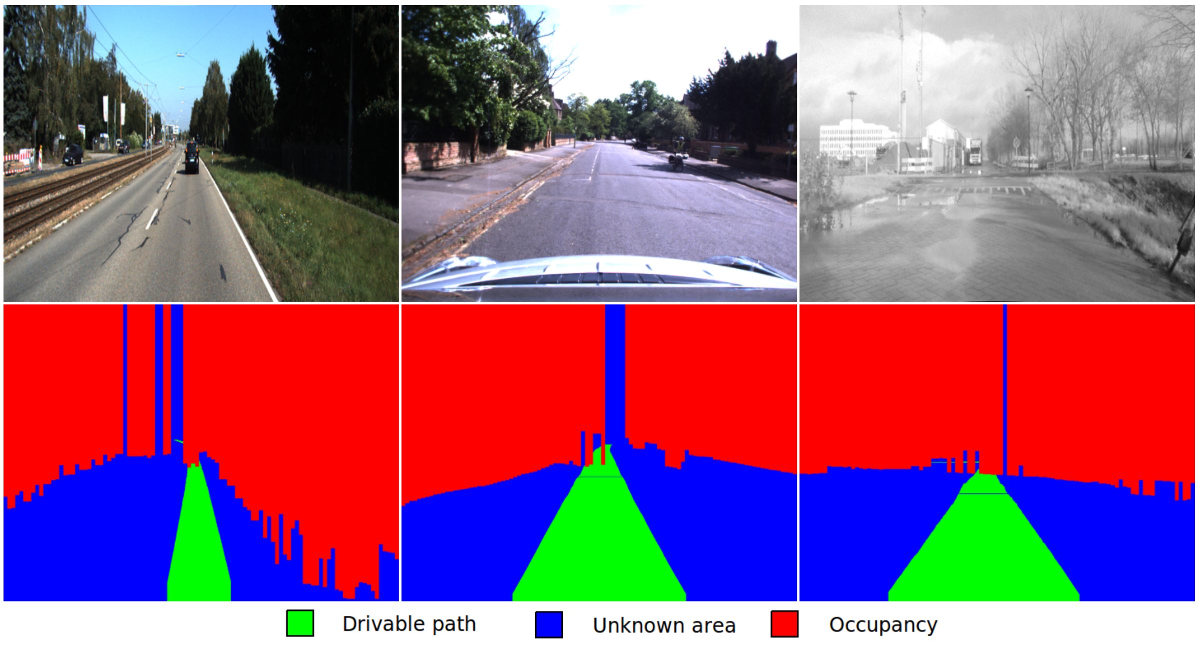


Figure 3.2: Example images and created labels of the KITTI raw dataset (left), Oxford RobotCar (middle) and the WEpod dataset (right). Red refers to occupancies, blue to unknown area and the drivable path is depicted as green. All images are a resized to fit.

3.1.3. SegNet

A neural network is deployed to segment images into three classes (occupancies, drivable path and unknown area). The network needs to be robust and the goal is not focused on achieving state-of-the-art results. [5] proves that SegNet is able to achieve good results with the same classes and therefore SegNet is chosen as network for semantic segmentation.

SegNet is an encoder-decoder network first introduced by [3] in 2015. Despite not obtaining state-of-the-art results for semantic segmentation anymore, it has proven to be a very robust network and often acts as a baseline for experiments with new architectures. The architecture of SegNet is shown in figure 3.3.

The architecture shows the encoder on the left side while the decoder follows on the right side. The encoder of SegNet is identical to the VGG16 network [59] except for the three fully connected layers of VGG16, these are left out in SegNet. Every convolutional layer (except the last layer in the decoder) is followed by a batch normalisation layer [26] and a Rectified Linear Unit (ReLU) activation function. The pipeline of these operations is presented as a blue layer in figure 3.3. In order to reduce the resolution of the input (feature) map a pooling layer is applied at the end of each block (the green layers in figure 3.3). During these downsampling steps, indices of the pooled (maximum) values are transferred to the corresponding upsampling layers (red layers in figure 3.3) in the decoder, aiming to keep the spatial structure of the original image. The last layer of the decoder consists of a softmax layer (yellow layer in figure 3.3) which results in a pixel-wise segmented image.

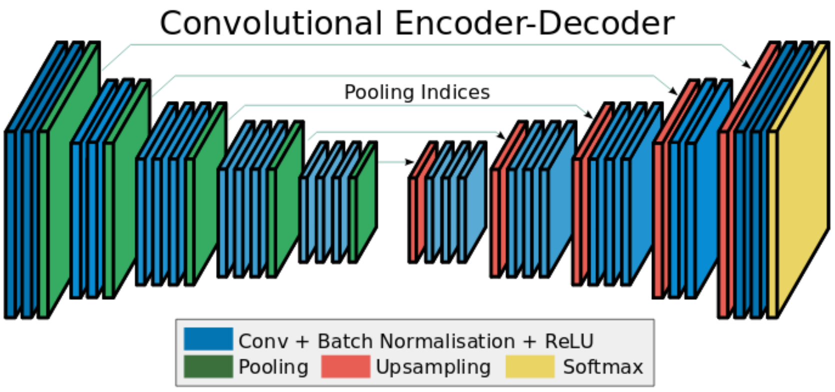

Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax

Figure 3.3: Encoder-decoder architecture of the implemented CNN. Image modified from [4].

Convolution layer

A convolutional layer consists of a filter (i.e. kernel). The size of the filter is adaptable but generally equals three by three pixels. The third dimension of the kernel will always cover the total number of input (feature) maps. The kernel is filled with values called weights and will slide over the image with a certain stride; when the stride equals one, the complete filter will shift with one pixel at the time. During this operation, it will multiply every element of the kernel with the corresponding element on the feature map(s) and finally sum all these multiplications, resulting in one value. This process is often referred to as neuron. The operation is simply represented by the following formula:

$$FM_{R,C} = \sum_{K} \left(W_{R,C,k} \cdot x_{R,C,k} \right) + \text{bias}, \tag{3.3}$$

where FM is the output feature map, W are the kernel weights and x is the input (feature) map. In this equation, R represents the row and C shows the column. K is equal to the number of feature maps while $k \in K$. The number of channels of the output (activation) map is equal to the number of filters.

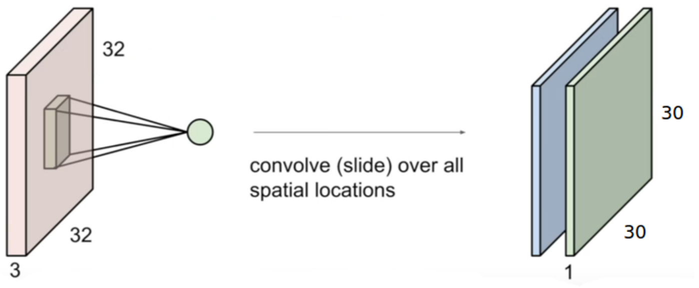


Figure 3.4: Low level example of convolution in a neural network. Image modified from CS231n course[2].

Batch normalisation layer

Introduced by [26], Batch Normalisation (BN) has three (main) goals:

- Batch normalisation improves gradient flow through the network
- Batch normalisation allows higher learning rates
- Batch normalisation reduces the dependency on initialisation

The BN layer often occurs in one pipeline with the convolutional layer and is followed by an activation layer. During the training stage, every mini-batch is normalised separately by computing the mean and variance of that specific mini-batch. Two parameters, γ and β are learned through training. These parameters represent a scaling and shifting value respectively, resulting in more flexible learning.

[2]Lecture 5 - Convolutional Neural Networks:
`https://www.youtube.com/watch?v=bNb2fEVKeEo&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk&index=5`

Activation layer

The activation function is a key component within neural networks. It brings non-linearity into the network which enlarges the potential capabilities to solve complex problems. There are a variety of activation functions which should be differentiable in order to train the network via back propagation.

A Rectified Linear Unit (ReLU) is used in SegNet which mathematically is presented by:

$$f(x) = \max(0, x), \tag{3.4}$$

where x represents the input to a neuron while f(x) equals the output of the activation layer. The ReLU layer is visualised in figure 3.5.

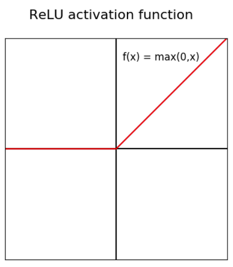


Figure 3.5: Visualisation of the ReLU activation function.

Utilising a ReLU function will lead to a sparse network since not every neuron will be activated by ReLU which is a positive attribute concerning computation. Examining figure 3.5, it is clear that all negative values will be converted to zero. The result of the behaviour of the ReLU function in the negative domain is a gradient which also equals zero. Because the gradient is zero, weights are not updated during back propagation and create *dead* neurons which never will be activated again.

Pooling layer

There are different variants of pooling layers such as average pooling but SegNet only uses maximum pooling layers with a dimension of two by two. Hence, the pooling layer is a window of two by two pixels that slides over every channel of the image or feature map. During this operation it only keeps the maximum value and hence reduces the output resolution by a half in the case of SegNet.

The indices of these maximum values are also stored which is specific for the SegNet architecture. These indices are used in the corresponding upsampling layer in the decoder to upsample the feature maps. This transfers the spatial structure of the original image to the output vector and is visualised in figure 3.6.

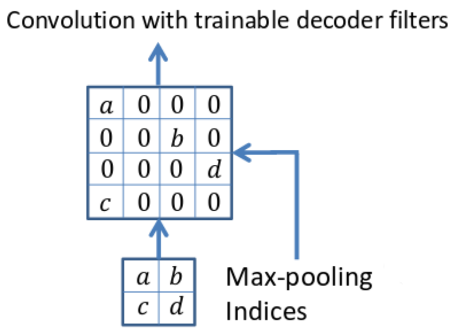


Figure 3.6: Visualisation of the upsampling process of SegNet. Image obtained from [5].

Softmax layer

The softmax layer in SegNet is the last layer of the network and acts as a classifier. This layer produces an probability for each class with the highest value for the most probable class. The distribution of these probabilities, however, depends on the regularisation strength. The softmax function is also known as the normalised exponential and equals the multi-class generalisation of the logistic sigmoid function [6]. The softmax function is represented by

$$S(y_i) = \frac{\mathrm{e}^(y_i)}{\sum\limits_j \mathrm{e}^(y_j)}, \tag{3.5}$$

where y is the input vector with length j over which the softmax function is computed. Hence, y_i represents the ith element of this input vector.

Loss function

SegNet is using a modified version of the cross-entropy loss. The original cross-entropy loss of one observation is expressed as:

$$\mathscr{L}(y, \hat{y}) = -\sum_{c=1}^{C} y_c \ln(\hat{y}_c), \tag{3.6}$$

where C represents the total number of classes. y_c is a binary indicator for class c and is equal to 1 when the observation is classified correctly and equals 0 otherwise. $\hat{y}_c$ is the predicted probability that the observation is classified as class *c*.

To tackle the problem of class imbalance (i.e. more pixels are classified as obstacles and unknown area than drivable path), the original cross-entropy loss is weighted in the SegNet architecture. These weights are class dependent and different for each dataset. The imbalance weights are not trainable which means that they are constant throughout training. Computation of the weights is according to the equation below:

$$W_c = \frac{M(F)}{f(c)}, \tag{3.7}$$

where *f(c)* stands for the frequency of *c* (i.e. the number of pixels of class *c* divided by the total amount of pixels in the image). *M(F)* is equal to the median of frequencies of all classes. This weighting procedure results in low weights for the larger classes while the smaller classes (drivable path in our case) has the highest value and hence, will cause the loss to increase. The weight values for occupancy, drivable path and unknown area are determined for a subset of the training data consisting of 256 images.

The objective or cost function of SegNet is a weighted cross-entropy loss, summed over all pixels of the mini-batch and expressed as:

$$\mathscr{L}_{Cross-Entropy}(y, \hat{y}) = -\sum_{p=1}^{P}\sum_{c=1}^{C} W_c \cdot y_c \ln(\hat{y}_c), \tag{3.8}$$

with P equal to the total amount of pixels in each mini-batch (containing four images in our case). **Note:** the regularisation term in this loss function is ignored.

Regularisation

Although, the regularisation term is not included in equation 3.8, it is present in SegNet. Regularisation is added to the cost function in order to prevent the model from over-fitting. Hence, the total loss function exists out of two parameters: the data loss and the regularisation loss. This is shown in the following equation where the first part equals the data loss and the second part is the regularisation loss:

$$\mathscr{L}_{SegNet}(y, \hat{y}) = -\sum_{p=1}^{P}\sum_{c=1}^{C} W_c \cdot y_c \ln(\hat{y}_c) + \lambda R(W), \tag{3.9}$$

where λ is the regularisation strength and also referred to as weight decay parameter. This parameter determines the ration between the data loss and the regularisation loss. $R(W)$ represents the regularisation method. SegNet utilises the commonly used L2 norm which is the sum of the element-wise square of weights and expressed as:

$$R(W) = \sum_{e=1}^{E} w_e^2, \tag{3.10}$$

where, E is total amount of elements in the weight matrix. This will penalise the larger weights more because of the presence of the square.

3.2. Domain adaptation

As referred to in section 2.4, Transfer Learning or TL is one approach for resolving the fact that often no ideal dataset is available. However, TL is a large collection of scenarios based on different situations of deploying a certain task on a certain domain and contains three subcategories according to [46]: inductive transfer learning, transductive transfer learning and unsupervised transfer learning. This is summarised in table 3.1. The case of transductive transfer learning refers to the setting where the target domain does not have labelled data available while the source domain does. Finally, the last category of TL is the method where labelled data is not available in both the source and target domain. This is referred to as unsupervised transfer learning. This research focused on a special form of transductive transfer learning, called Domain Adaptation (DA). Domain adaptation requires a setup where source task and target task are the same but the domains where these tasks are deployed are different.

Learning setting		**Source and target domain**	**Source and target task**
Traditional machine learning		the same	the same
Transfer Learning	Inductive Transfer Learning	the same	different but related
	Transductive Transfer Learning	different but related	the same
	Unsupervised Transfer Learning	different but related	different but related

Table 3.1: Relationship between traditional machine learning and transfer learning setups. Obtained from [46].

It is uncommon to train an entire CNN based on random initialisation, referring to the initial weights of the CNN being random values. Datasets are often not large enough to be able to train with random initialisation. In an exceptional case when the dataset is sufficiently large, the consequence of a very long training period cannot be avoided. Instead, it is common to use weights which are already available through the increasing popularity of neural networks.

These weights can be used as a fixed feature extractor. This relates to the case that the weights of the pre-trained network are directly used and hence are not influenced by the training phase of the network. Except this fixed feature extractor, another possibility is to use the weights as initialisation. This allows the network to adjust the weights slightly (i.e. fine-tune) during the training phase. This is useful when the pre-trained network had different number of classes which should be classified. How the weights are fine-tuned, can be tuned per layer.

The following section will introduce the phenomenon called domain shift. The approach of the research concerning the factors influencing this domain shift is explained in section 3.2.2. Finally, in subsection 3.2.3, the evaluation metrics are explained. These metrics are used to make conclusions about the influence of different factors on the domain shift for semantic segmentation.

3.2.1. Domain shift

Based on the setting of DA mentioned above, it is possible to use a network which is trained on a source domain for the task of semantic segmentation while deploying the network for the task of semantic segmentation on a different but related domain than the source domain. However, generally this setup show huge performance drops for the target domain while achieving good results for the source domain, this is shown in figure 3.7. This problem can be explained by the difference in distributions of the domains, also known as the domain shift. This limited ability of a neural network to adapt itself to new domains is a common problem in DA.

Mitigation of the domain shift is examined by a broad range of papers, however, only few resources target to track the domain shift down to factors that can be manipulated. [31] analysed the influence of four factors on object detection: spatial location accuracy, appearance diversity, image quality and aspect distribution. They managed to show that these four factors almost close the whole performance gap which resulted from the domain shift. However, this research focused on object detection and in our case, the focus will be on semantic segmentation.

In order to address the influence of factors on the success of the domain adaptation, a baseline is created. This baseline acts as an evaluation marker to monitor whether factors have a positive or negative influence on the ability of the network to adapt itself to a new domain. The factors that are examined are mentioned in section 3.2.2.

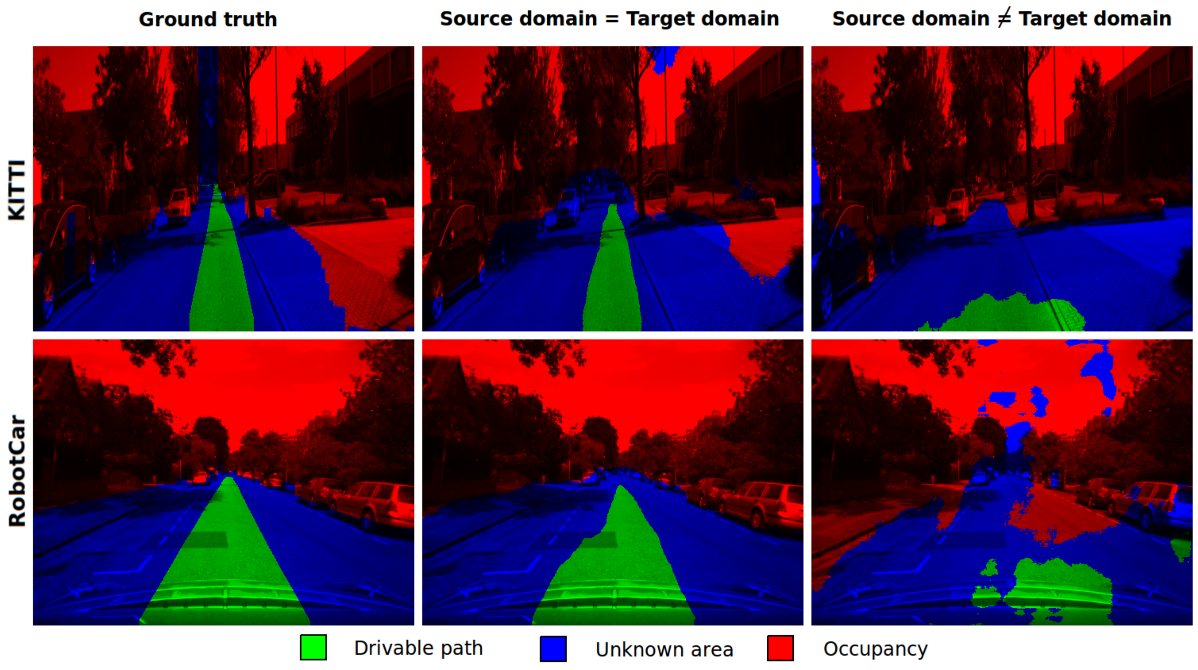


Figure 3.7: Visualisation of domain shift. Left is the ground truth label for two images. The middle column represents the estimated drivable path when trained on the target domain. The right column represents the estimated drivable path when segmentation is deployed on a target domain that differs from the source domain.

3.2.2. Factors

Each factor that is examined is listed and the methodology of each experiment is explained in the subsections below.

Default network and upper bound

In order to be able to determine the influence of different factors and to get a global view on the consequences of different modifications, it is important to set one default network. Every factor is evaluated compared to this default setting and thus no combinations of multiple factors is examined. For the default setting the modification elements within the workflow are not activated.

Data equalisation

The initial dataset combination of RobotCar and KITTI consisted of 2730 and 1060 training images respectively. In contrast to the imbalance in classes, the imbalance between these two domains is not resolved in the loss function. Therefore, equalising the number of training images from these datasets will potentially resolve a bias towards the larger dataset in the feature space.

Order of training

Another training setup is effecting the order in which the network will be fine-tuned. As a first approach, SegNet is fine-tuned on RobotCar first and later on it is fine-tuned on KITTI. To exclude the effect of tuning order, the setup is reversed; first train on KITTI and later on RobotCar. Exactly the same data is used to train with exactly the same settings. The only difference is the order of tuning and hence the default workflow does not match with this setting.

Greyscale

Originally, SegNet is intended for RGB images. However the target imagery (WEpod) is only available in greyscale. Initially this was handled by copying the greyscale channel three times such that the original filter weights could be implemented on the greyscale images of the WEpod. However, this implies that features which are learned primarily by colour are not obtained when tested on target imagery. Therefore, another setup was made by fine-tuning the neural network on the same images but converted to greyscale. The comparison of these setups will indicate to what extend colour is important for creating features, which is automatically done by the neural network.

In neural networks it is important to detect edges and the combination of the edges make an object. For tracking most edges, colour is not necessary. The transformation from RGB to greyscale is computed using the formula noted in Rec. 601 [17]. The conversion is formulated as:

$$Y = 0.30 \cdot R + 0.59 \cdot G + 0.11 \cdot B, \tag{3.11}$$

where R, G, B represent the red, green and blue channel of the original image respectively. Y is the output value describing the intensity of the pixel with the same range of the original RGB channels.

Left-hand traffic

The RobotCar dataset is recorded in Oxford and consequently trajectories of the vehicle are located on the left side of the road. The opposite is true for KITTI and the target domain of WEpod, thus an experiment where the training and validation images are flipped is executed. A flipped image can simply be created by the following expression:

$$\text{Flipped}_{r,c} = \text{Original}_{r,W-c}, \tag{3.12}$$

where *Flipped* is the flipped image and *Original* is the source image which is being flipped. r and c run through every pixel in row and column respectively. W is the width of the source (and hence destination) image. An example is shown in figure 3.8.

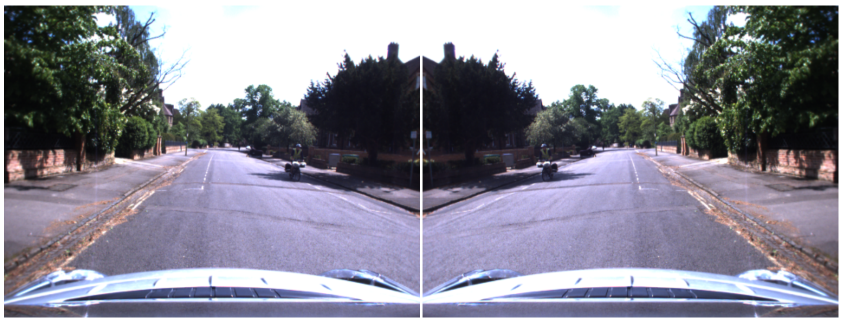


Figure 3.8: Original image of the RobotCar training set (left), after the flipping procedure (right).

Gamma correction

The human eye perceives colour and luminance differently than the camera sensors of the recording platform. When the sensor acquires double the amount of photons, the signal is doubled where the human eye only will see this doubled amount as a fraction brighter. In other words: the camera has a linear relationship between amount of photons and signal while the human eye has a non-linear relationship.

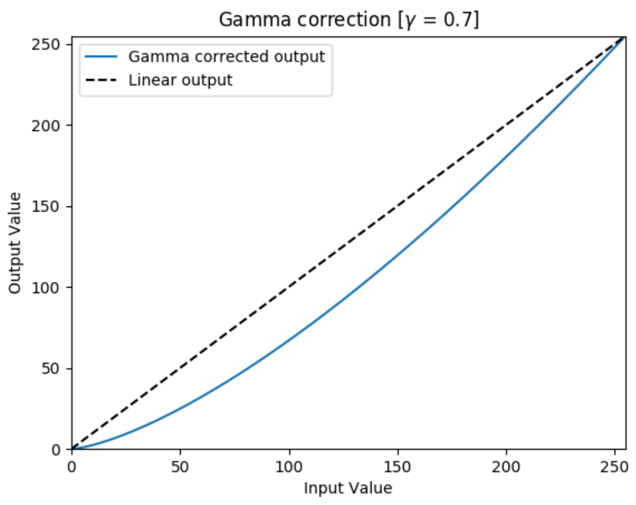


Figure 3.9: Gamma correction for $\gamma = 0.7$.

In order to account for this effect, gamma correction is applied which enhances the contrast of an image. The mathematical expression of the gamma correction is depicted in figure 3.9 and is formulated as:

$$V_{output} = \left(\frac{V_{input}}{255}\right)^{1/\gamma} \cdot 255, \tag{3.13}$$

where V_{output} is the gamma corrected output value and V_{input} equals the input pixel value. This input pixel value is the *Value* component in the HSV colour space. Hence the image is first converted from RGB colour space to HSV colour space before the power law transform is performed. The conversion of RGB colour space to HSV colour space is expressed as:

$$\begin{aligned} V &= \max(\mathrm{R,G,B}), \\ S &= \begin{cases} \frac{(\mathrm{V - min(R,G,B)}}{V} & \text{if V} \neq 0 \\ 0 & \text{otherwise} \end{cases}, \\ H &= \begin{cases} \frac{60\cdot(G-B)}{(V-min(R,G,B)} & \text{if V = R} \\ \frac{120+60\cdot(B-R)}{(V-min(R,G,B)} & \text{if V = G} \\ \frac{240+60\cdot(R-G)}{(V-min(R,G,B)} & \text{if V = B} \end{cases}. \end{aligned} \tag{3.14}$$

Histogram equalisation

A method for contrast enhancement of an image is to distribute the intensity values more uniform over the complete range. The dispersion of intensity values is translated as forcing the cumulative density function of the pixel intensities of the image into a linear trend. A quantification of this intensity distribution is the image histogram. An example of the intensity distribution of an original WEpod image is shown in figure 3.10. After applying histogram equalisation, the intensity values are indeed more spread out over the total range as can be seen in figure 3.11.

Histogram equalisation is a non-linear process on the intensity values of the image. Hence, to preserve the colour balance of the image, the RGB colour space is converted to a representation that separates the intensity values from the colour components. One of the possibilities is to convert the YUV colour space which follows the following equations:

$$\begin{aligned} Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B, \\ U &= -0.147 \cdot R - 0.289 \cdot G + 0.436 \cdot B, \\ V &= 0.615 \cdot R - 0.515 \cdot G - 0.100 \cdot B. \end{aligned} \tag{3.15}$$

After the conversion, the equalisation is deployed on the intensity values, presented as Y-value in YUV space. The updated YUV values are then converted back to RGB values by means of equation 3.16.

$$\begin{aligned} R &= Y + 1.140 \cdot V, \\ G &= Y - 0.395 \cdot U - 0.581 \cdot V, \\ B &= Y + 2.032 \cdot U. \end{aligned} \tag{3.16}$$

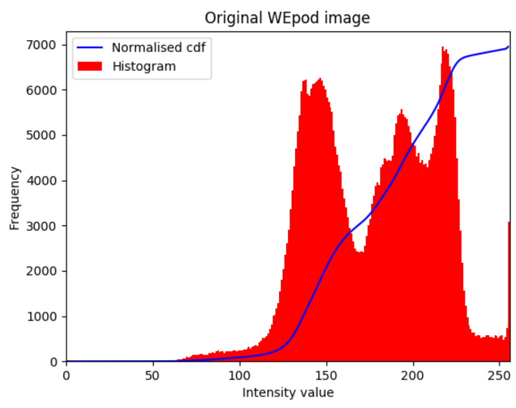


Figure 3.10: Histogram of a raw WEpod image.

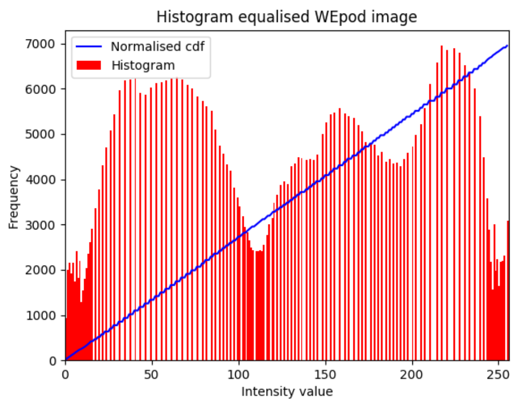


Figure 3.11: Histogram of the WEpod image after histogram equalisation.

Number of classes

The initial proposal included three classes. Because our target platform is equipped with a lidar, it would be possible to deploy it during test phase. Therefore, occupancies can be processed on the fly and only two classes need to be present in the segmentation: drivable path and *non-drivable path*. The idea behind the focus on path proposal estimation is that fusing two chaotic classes will increase the overall accuracy. Occupancies and unknown area are referred to as chaotic because they do not have clear distinguishable features.

Horizon

The height of the horizon line in the image for a certain camera is the result of the camera height, roll and pitch. KITTI and RobotCar data have a similar horizon height in the images (they only differ by a few pixels). However, WEpod images have a relatively low horizon line due to the significant lower placement of the camera (0.8m versus 1.65m for KITTI and 1.52m for RobotCar), resulting in a smaller part of the image containing road and potentially a crucial difference between the datasets. To examine this, the horizon height of the WEpod test images is changed such that it is similar to KITTI and RobotCar.

Cropping

The original aspect ratio differs for all three domains. Therefore, when resizing the images to the fixed input aspect ratio this will lead to strange deformations for KITTI and to a lesser extent for WEpod. RobotCar does not have these deformations since the input resolution has the same aspect ratio as the raw images. To correct for these deformations, images of the KITTI domain are directly cropped to the input size of the network and thus no resizing is applied after the cropping. In the test set, WEpod images are cropped to the input aspect ratio and afterwards resized to the fixed input size.

Intrinsic consistency

Datasets are created with the use of different recording platforms and often different camera sensors. These sensors have different characteristics which have influence on the output in the form of an image. To cancel these differences, the datasets are manipulated to be intrinsic consistent. Intrinsic consistency refers to consistent intrinsic parameters throughout the different camera sensors. In order to acquire consistency, three steps are taken.

First, the Horizontal and Vertical Field of View (HFOV and VFOV) are determined. The HFOV is obtained by projecting two points on the road, two metres apart in horizontal direction and seven metres away from the camera in longitudinal direction. The pixel coordinates of these two points are obtained and the difference between them in horizontal direction is calculated. A similar approach is followed for calculating VFOV, separating the points in vertical direction instead of horizontal direction. It turned out that HFOV is the same (pixel-wise) as VFOV for all platforms and hence, only one is noted in table 3.2. The actual HFOV and VFOV at seven metres of the camera, are calculated by taking the image size into account.

	D_{dataset} 2m @ 7m [px]	Image size [H x W]	HFOV @ 7m [m]	VFOV @ 7m [m]
KITTI	206	375 x 1242	12.06	**3.64**
RobotCar	276	960 x 1280	9.28	6.96
WEpod	239	512 x 1024	**8.57**	4.28

Table 3.2: Characteristics for the KITTI, RobotCar and WEpod camera sensors.

Second, the images are cropped to their new size. This new size is defined by the characteristics of table 3.2. All images are cropped relative to the smallest HFOV and VFOV, resulting in a decisive height of KITTI and decisive width of WEpod. The new height is calculated as:

$$H_{\text{new, dataset}} = \frac{VFOV_{\text{normative}}}{VFOV_{\text{dataset}}} \cdot H_{\text{original}}, \tag{3.17}$$

and the new width as,

$$W_{\text{new, dataset}} = \frac{HFOV_{\text{normative}}}{HFOV_{\text{dataset}}} \cdot W_{\text{original}}. \tag{3.18}$$

In these equations $H_{new,dataset}$ and $W_{new,dataset}$ are the new height and width of that particular dataset, respectively. $VFOV_{normative}$ represents the ratio of normative VFOV (i.e. smallest VFOV; bold in table 3.2) to dataset VFOV and both values are extracted from table 3.2. Similarly, $W_{normative}$ represents the ratio of normative HFOV (i.e. smallest HFOV; bold in table 3.2) to dataset HFOV. Both values are found in table 3.2.

These new size are calculated for every dataset except the normative sizes (i.e. width of WEpod images and height of KITTI images) and therefore these will be kept as the original size. The new size are aligned below:

$$\begin{aligned} H_{\text{WEpod}} &= \frac{3.64}{4.28} \cdot 512 = 435 \text{ pixels}, \\ W_{\text{RobotCar}} &= \frac{8.57}{9.28} \cdot 1280 = 1182 \text{ pixels}, \\ H_{\text{RobotCar}} &= \frac{3.64}{6.96} \cdot 960 = 502 \text{ pixels}, \\ W_{\text{KITTI}} &= \frac{8.57}{12.06} \cdot 1242 = 883 \text{ pixels}. \end{aligned}$$

As a third and last step, the images are resized to a smaller uniform size while maintaining the aspect ratio. All images in the dataset are resized to 240 x 565. Figure 3.12 shows an undistorted and demosaic image of RobotCar which is not intrinsic consistent with the other two domains. The result of intrinsic consistency is shown in figure 3.13, also depicting less information than the original because RobotCar images are cropped both in height and width.

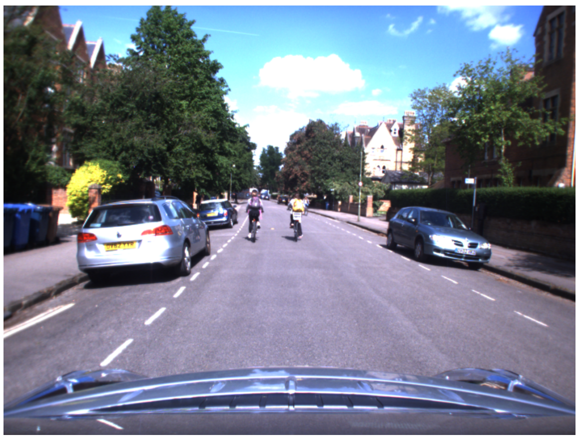


Figure 3.12: Undistorted and demosaiced image from the Oxford Robotcar dataset.

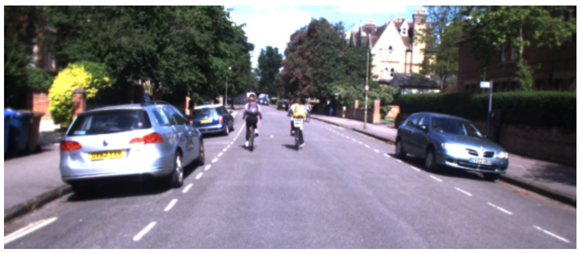


Figure 3.13: Intrinsic consistent image of Oxford Robotcar dataset.

3.2.3. Evaluation

Several classical metrics are available to evaluate the performance of semantic segmentation on pixel-level. However, since there is no benchmark suite for the semantic classes (drivable path, unknown area and occupancy) that are applied, no comparison with respect to the current state-of-the-art concerning semantic segmentation can be made. This comparison is of less importance for the experiments because the aim of the experiments is to depict the influence of each factor on the domain shift. This is achieved by comparing results to the results of the baseline instead of comparing it to the state-of-the-art. Although the state-of-the-art results are not needed, ground truth is required for the test images in order to evaluate the results both quantitatively and qualitatively. Ground truth is created by labelling test images similar to the labelling technique used on the training images.

When qualitatively evaluating a drivable path it is important that occupancies are not segmented as drivable path. This situation is potentially more catastrophic than drivable path segmented as occupancies or unknown area as occupancy. Hence, it is important to have as few false positives as possible and thus, precision is a more informative metric than recall in the case of drivable path estimates. Precision is indicated as P and calculated according to [18]:

$$P = \frac{TP}{TP + FP}, \tag{3.19}$$

where TP and FP denote True Positive and False Positive respectively.

For occupancies, the opposite is true. Qualitatively it is *better* to have classified too many pixels as occupancy than missing a lot of occupancies. Therefore, it is important to have as few false negatives as possible which means sensitivity is a more informative score than precision. Sensitivity or recall is indicated as R and expressed as [18]:

$$R = \frac{TP}{TP + FN}, \tag{3.20}$$

where FN denotes False Negatives. However, it has to be stressed that the performance cannot be summarized in one metric.

Quantitative evaluation of the semantic segmentation and hence, relating the effect of different factors to the domain shift is presented via two generic evaluation metrics: Intersection-over-Union (IoU) and F_1-measure (F_1). The IoU metric is also known as the Jaccard index and measures the similarity between two segments (in our case between the ground truth and the network output) and is stated as [18], [49]:

$$\mathrm{IoU(i)} = \frac{TP}{TP + FP + FN}, \tag{3.21}$$

where i representing image i. In order to summarise the performance in one value, the IoU metric is averaged over all images according to:

$$\mathrm{IoU(total)} = \frac{\sum_{i=1}^{n} IoU(i)}{n}, \tag{3.22}$$

with n being the total amount of images in the domain that is considered.

The F_1-measure considers both precision and recall being the harmonic mean of the two metrics. A perfect F_1-score (equal to one) is reached when both precision and recall are equal to 1. The F_1-measure or F_1-score is expressed as [18]:

$$\mathrm{F}_1\mathrm{(i)} = \frac{2 \cdot P \cdot R}{P + R}, \tag{3.23}$$

with P and R referring to the previously explained precision and recall metric. Similar to the Jaccard index, the F_1-score is summarised to one metric value according to:

$$\mathrm{F}_1\mathrm{(total)} = \frac{\sum_{i=1}^{n} F_1(i)}{n}. \tag{3.24}$$

Because the spatial resolution reduces as the road is further away, a pixel-wise metric is biased. This could be handled in different ways, such as weighting the metric values as they lie further from the vehicle. However, in the image space this effect is neglected.

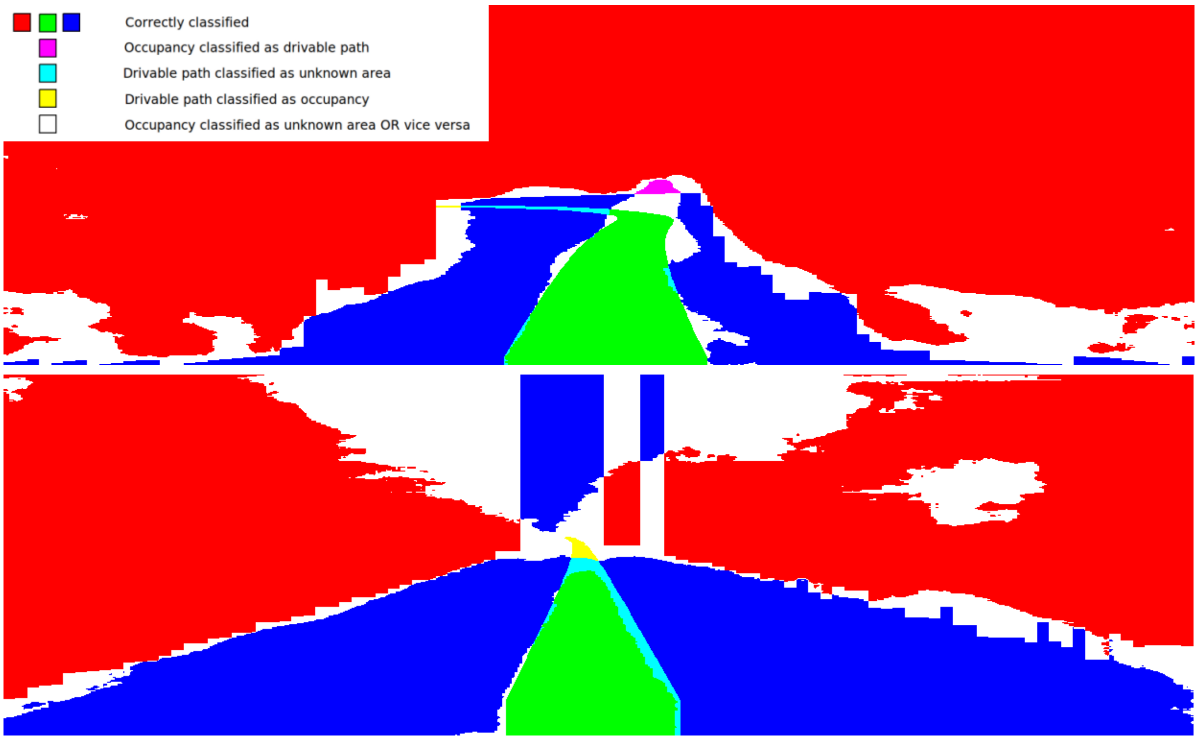

🟥 🟩 🟦 Correctly classified
🟪 Occupancy classified as drivable path
🟦 Drivable path classified as unknown area
🟨 Drivable path classified as occupancy
⬜ Occupancy classified as unknown area OR vice versa

Figure 3.14: Two visualisations of segmentation errors. Green, red and blue are correctly labelled as drivable path, occupancy and unknown area respectively. Yellow refers to drivable path classified as occupancy while cyan refers to drivable path classified as unknown area. Magenta corresponds to occupancies which are classified as drivable path. White pixels are classified as unknown area while being occupancy or vice versa or unknown area classified as drivable path.

3.3. Top view transformation

A path is a geometric trace that the vehicle should follow in order to reach its destination without colliding with obstacles. Path planning is finding a geometric path from an initial configuration to a given terminating configuration such that each configuration and state (if time is taken into account) on the path is a feasible one. A feasible configuration/state does not result in a collision and adheres to a set of motion constraints such as road and lane boundaries, as well as traffic rules. It should be noted that importance is given in finding the best and safest geometric trace, under the constraints described above which also have a logical argument regarding the rules of traffic.

For the application of autonomous driving, four different views are considered [40]: image view, top view (i.e. bird's eye view), spherical view and cylindrical view. Figure 3.15 shows the four representations. The former two view points are easily understandable for humans as they represent the world as humans see it. However, taking the sensor perception into account, it gives a deformed representation because of the way the sensors create data. The monocular camera system retrieves data as a flat plane, missing one dimension which is retrieved by the human eye (stereo system). High-level lidar (64-beam), rotates around its own axis which results in sparse data for image view and top view. Additionally, occlusions potentially occur in image space which automatically will result in problems for the autonomous vehicles. The segmentation results will be transformed to Cartesian coordinates because the current path planner of the WEpod is implemented in this space.

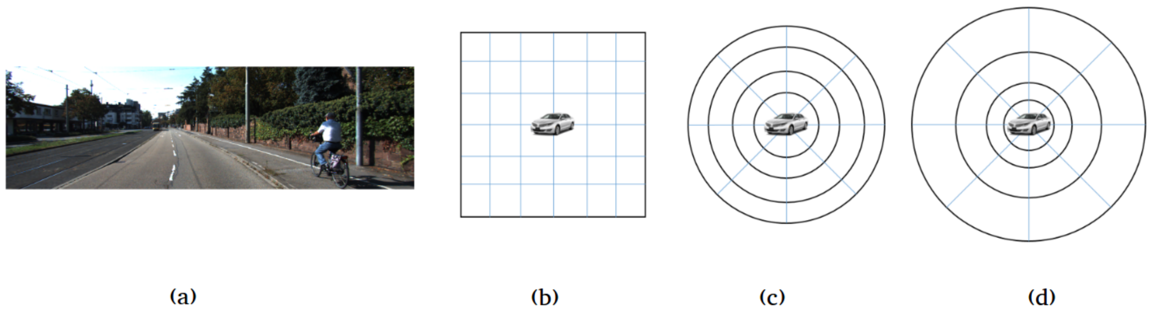


Figure 3.15: Four possible views. (a) Image space, (b) top view, (c) polar view and (d) spherical view. Image modified from [40].

3.3.1. Setups

Semantic segmentation of images retrieves an understanding of the environment in image space. To transform this understanding from image space towards top view there is one *factor* missing; depth. In order to acquire depth information, several options are possible. Two frequently used options are stereo imaging and lidar, both having their (dis)advantages. It is very challenging to obtain ground truth dense depth data which represent a road scene (or any scene) well. Scenarios for the transformation of image space to top view is only carried out for the KITTI dataset since ground truth depth is available from the 64-beam lidar. The RobotCar dataset and WEpod dataset are not considered because the 4-beam lidar is too sparse to represent depth well enough.

3.3.2. Without depth

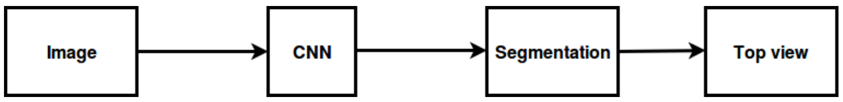


Figure 3.16: Setup for top view creation without ground truth depth.

Even without ground truth depth from lidar it is possible to create top view from the image space. This is executed in the first setup. This is done by obtaining a transformation matrix, based on four corner points of a rectangle which is chosen as the extent of the top view. The corner points define the rectangle on the road projected in the image (i.e. the projection is a trapezoid). In lateral direction this rectangle reaches up to 5 metres to the left and right of the origin of the camera frame. In longitudinal direction the rectangle spans from 7 to 46 metres measured from the origin of the camera frame, resulting in a rectangle with dimensions of 10 metre by 39 metre. The transformation matrix, M, is found by solving the following equation for all four corner points:

$$\begin{bmatrix} s_i \cdot u_i^* \\ s_i \cdot v_i^* \\ s_i \end{bmatrix} = M \cdot \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}, \tag{3.25}$$

In this equation, u_i and v_i describe the pixel values of the corner points while u_i^* and v_i^* are the corresponding quadrangle vertices in top view (i.e. the extent of the image). s_i is a scale factor. i is referring to a corner point and thus ranging from 0 to 3. The transformation matrix M is stated as

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}. \tag{3.26}$$

When the transformation matrix M is found, it is used to transform the source image (I) to the output (top view) image, O, via the following equation:

$$O_{u,v} = \mathrm{I}_{\frac{M_{11} \cdot u + M_{12} \cdot v + M_{13}}{M_{31} \cdot u + M_{32} \cdot v + M_{33}}, \frac{M_{21} \cdot u + M_{22} \cdot v + M_{23}}{M_{31} \cdot u + M_{32} \cdot v + M_{33}}}. \tag{3.27}$$

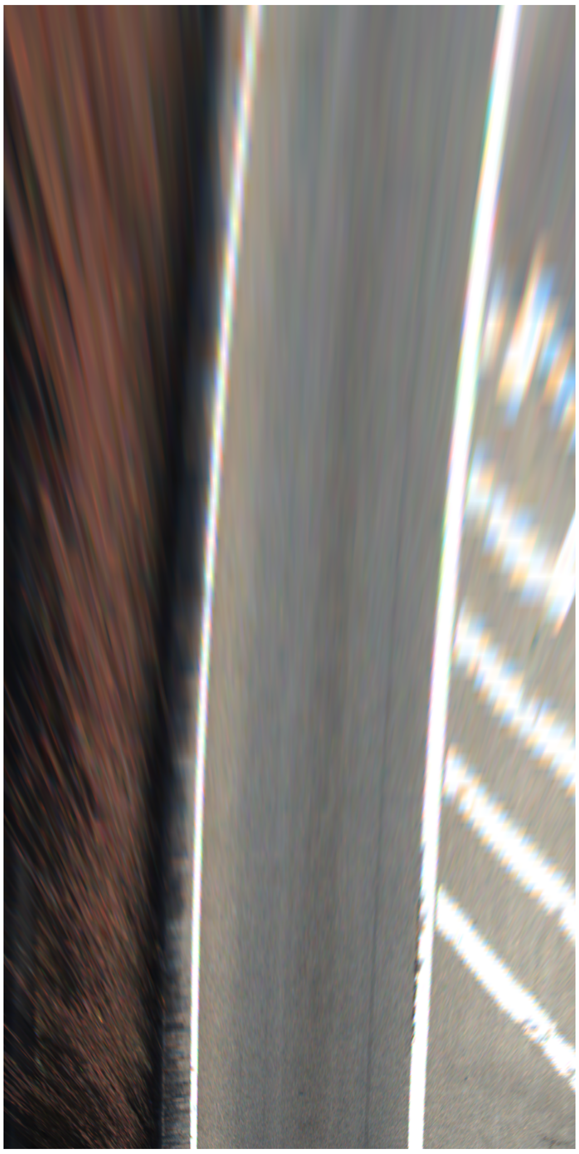


Figure 3.17: Transformed image without exploiting depth information.

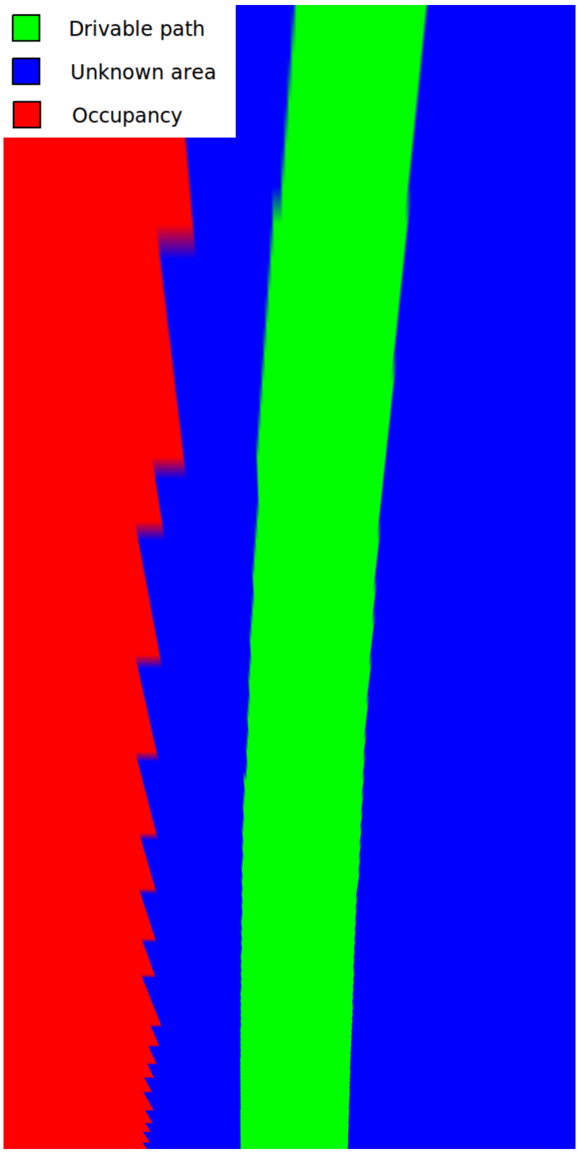


Figure 3.18: Transformed ground truth label of figure 3.17 without exploiting depth information.

From the example image (figure 3.17 and 3.18), it is seen that the spatial resolution reduces towards the edges of the image. This leads to an unrealistic top view, where objects at the edges and further away are blurry. This is due to the fact that the spatial resolution of this area of the image is lower, meaning that more interpolation of pixels is needed in these areas compared to the areas closer to the camera. This process is clarified in figure 3.19 where the transformation from a trapezoid form to a rectangle form is visualised. From the figure it is clear that the top edge will be "stretched" the most while the bottom edge will have the least deformation.

While this blurriness and hence incorrect representation of the truth is considered a disadvantage of the method, there are also some advantages to be considered. As mentioned previously, this method does not use any depth information. The rectangle which is projected into the image in order to transform to top view is computed via the intrinsic camera matrix, K (equation 3.2) and dimensions on the position of the rectangle with respect to the camera frame. Hence, the position of the camera with respect to the road is required. The output of the process is dense which is another advantage. However, it should be noted that the quality of the output differs throughout the image. The quality in the top corners is lower due to the previously mentioned lower spatial resolution in the image.

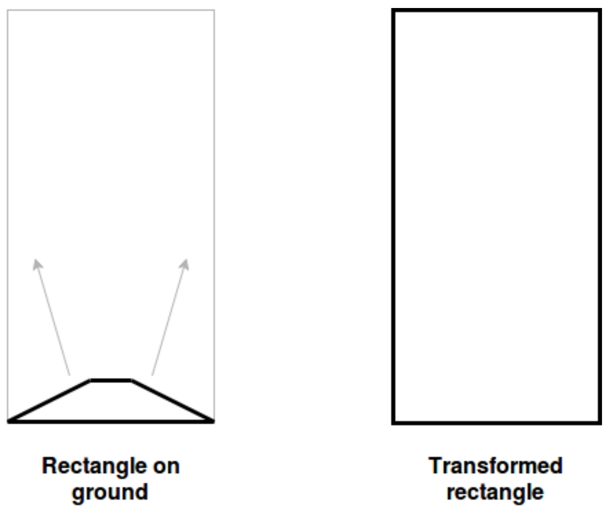


Figure 3.19: Example of shape modification due to the transformation to top view.

3.3.3. With ground truth depth

In the case where ground truth depth information (depicted in figure 3.21) is available it is used to to transform from image space to top view. Since only the data from KITTI will be examined, the setting of this data is taking into account. For the KITTI dataset, sparse depth maps (as images) are available resulting from the 64-beam lidar. These depth maps are directly used to reproject the semantic image segmentation to top view. An overview of the setup is seen in figure 3.20.

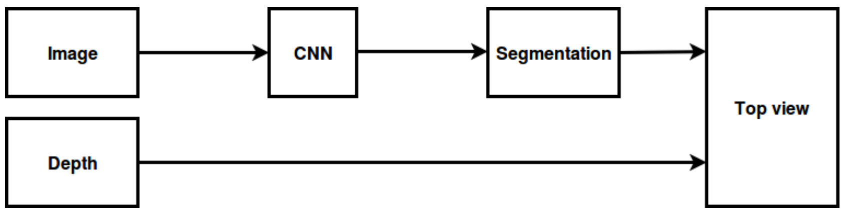


Figure 3.20: Setup for top view creation with ground truth depth.

Top view is created by transforming every image pixel containing depth information and therefore the resulting top view will also be sparse. The transformation follows from the opposite of the conversion from x, y, z points in the camera frame to pixel coordinates in the image space. This is also referred to as Inverse Perspective Mapping (IPM). Writing out the matrix multiplication from equation 3.1 results in the following set of equations.

$$\begin{aligned} s \cdot u &= f_x \cdot x + c_x \cdot z, \\ s \cdot v &= f_y \cdot x + c_y \cdot z, \\ s &= z. \end{aligned} \tag{3.28}$$

These equations are simplified to:

$$z \cdot u = f_x \cdot x + c_x \cdot z, \tag{3.29}$$

and,

$$z \cdot v = f_y \cdot x + c_y \cdot z. \tag{3.30}$$

Extracting x and y from these equations results in

$$\begin{aligned} x &= \frac{z \cdot (u - c_x)}{f_x} \\ y &= \frac{z \cdot (v - c_y)}{f_y} \end{aligned} \tag{3.31}$$

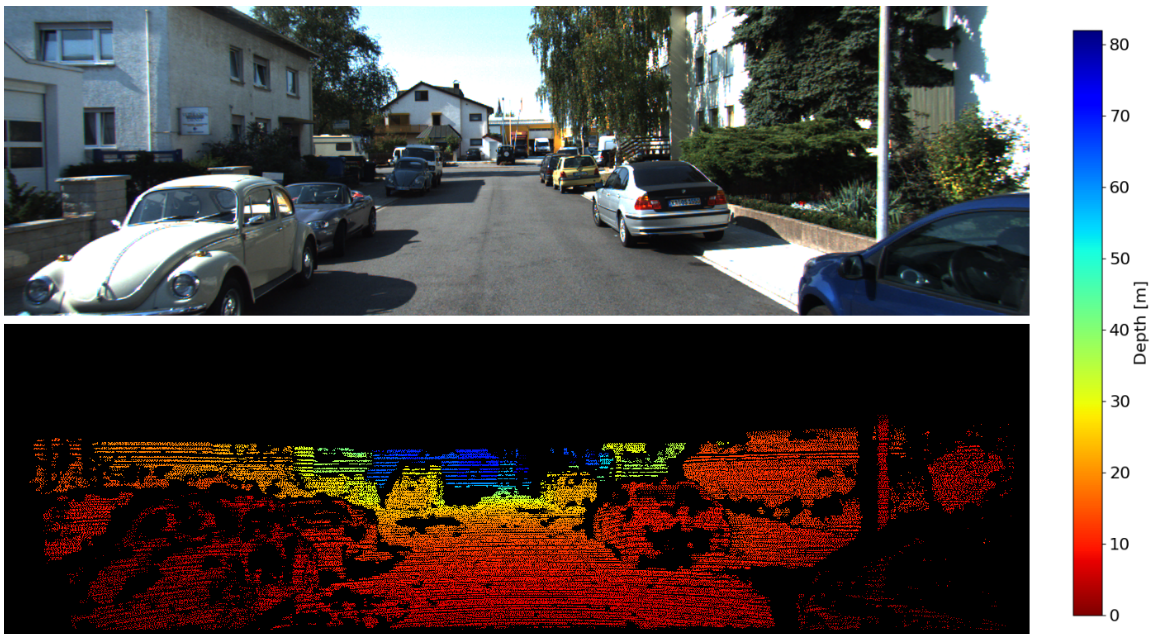


Figure 3.21: KITTI image (top) with corresponding ground truth depth (bottom).

With this transformation x and y-values for every pixel in the image containing depth information is retrieved. The depth information itself is the missing z-coordinate. However, to create top view only the x and z-values are used while the y-values are changed to zero in order to create a plane. Since the output is sparse due to sparse depth information, the resolution of the top view has to set. Resolution in both x and z-direction is set to 0.1 metre.

3.3.4. With depth estimating CNN

[35] resolve the problem of estimating depth based on a single image by proposing a CNN. This architecture is fully convolutional and exploits residual learning (ResNet-50) to create (dense) depth maps of monocular images. Since these depth maps are dense, it is easy to add depth values for each pixel in the segmented image. The approach which includes these dense depth maps to transform from image space to top view space is depicted in figure 3.22.

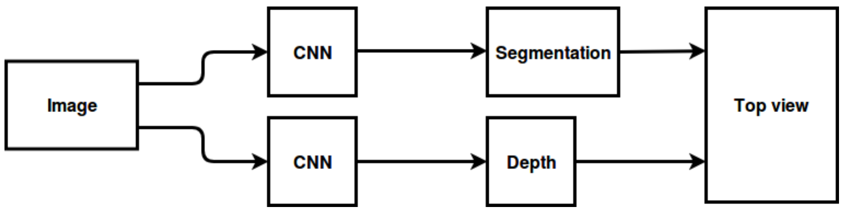


Figure 3.22: Setup for top view creation with depth information from a depth estimating CNN.

Figure 3.23 shows an image of the KITTI dataset and the corresponding estimated depth map, acquired via the aforementioned architecture.

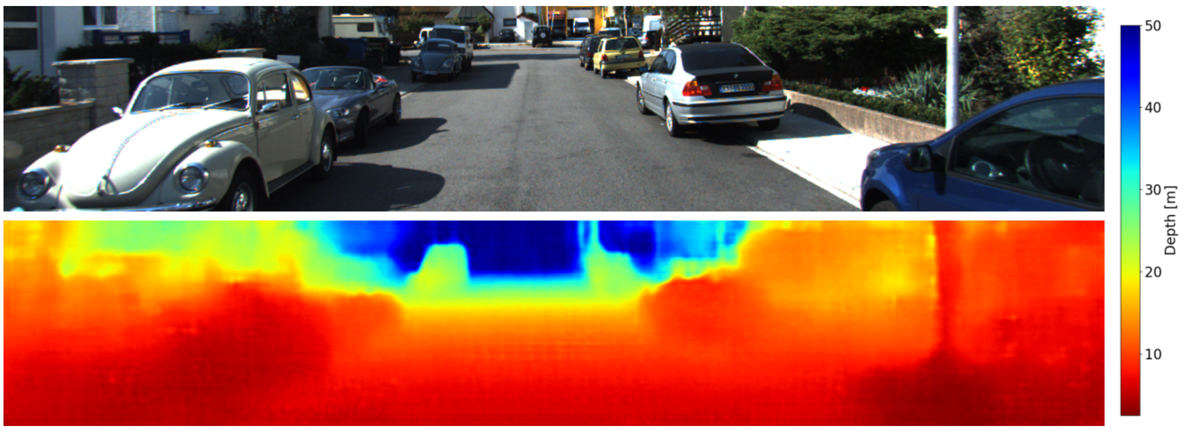


Figure 3.23: KITTI image (top) with corresponding estimated depth (bottom).

3.3.5. Evaluation

The transformation from image space to top view is evaluated in the top view space. Besides an evaluation metric for the importance of depth accuracy in the transformation process, the approach automatically generates an additional metric for evaluating the task of semantic segmentation.

In top view, two out of the three approaches will result in a sparse projection. However, because these two approaches do not produce exactly the same sparse projection (i.e. one is more sparse than the other), an intermediate step should be done in order to be able to compare results of these different methods. This step generates trajectory lines in top view based on the drivable path in top view. The trajectory line is retrieved by obtaining the lateral median of values classified as drivable path. In other words, for every row containing pixels classified as drivable path the median of these drivable path pixels is obtained. Additionally, to make the trajectory less noisy, the median is only acquired when the row contains more than ten pixels classified as drivable path. The trajectory is created for both the estimated drivable path and the ground truth which, ideally, would result in two overlapping lines.

The ideal prediction of two completely overlapping trajectory lines (i.e. prediction is 100% correct) is, however, never obtained in practice. Except the size of the lateral error, it is important where the error occurs in longitudinal direction. In other words, it is important to incorporate at what distance from the vehicle the estimated trajectory and ground truth trajectory separate from each other. To take this in consideration, the value of the difference between estimation and ground truth is scaled by weights. These weights are inversely proportional to the distance from the vehicle, ranging from one for the estimation closest (i.e. smallest longitudinal distance) to the car to zero for the estimation furthest away from the vehicle.

Two parameters are extracted from the trajectory lines. The lateral error is represented by the average difference between the estimated trajectory line and ground truth trajectory line. This value in absolute sense is an additional evaluation measure for the drivable path segmentation. The disadvantage of this additional metric is the lack of a benchmark. The advantage of this method is the practical representation of the value. This metric is calculated as

$$E = \sum_{i=1}^{k} \frac{S_i \cdot |g_i - p_i|}{k}, \tag{3.32}$$

where i represents every row in the top view that contains both a predicted value, p_i, and a ground truth value, g_i, and k notes the number of rows where ground truth value exists. The scale in order to account for the importance of the predicted value is symbolised by S_i. This scale decreases linearly from one in the direct range of the vehicle to zero at the most distant measurement of the vehicle. E is thus a scaled difference metric between the ground truth and prediction.

Another parameter that should be considered is the amount of overlap in longitudinal direction between the estimation and ground truth trajectory. Obviously, the estimated trajectory should ideally stretch over the same longitudinal length as the ground truth trajectory and can be found by the following simple expression:

$$O = \frac{j}{k}, \tag{3.33}$$

where j notes the number of rows where estimated trajectory and ground truth trajectory overlap in longitudinal direction. O is the amount of overlap in longitudinal direction between ground truth and prediction resulting in a value between zero (i.e. no overlap) and one (i.e. full overlap).

Semantic segmentation

Both metrics (equation 3.32 and equation 3.33) can be utilised as additional metrics for the quality of the semantic segmentation. These metrics are an evaluation of the estimated trajectory compared to the ground truth trajectory while using the same method and hence the same depth estimation. In this way, the evaluation is independent of the depth accuracy. A visualisation of the trajectories is depicted in figure 3.24. Since there is no benchmark in this format is available, no comparison with other methods and networks can be made.

Effect of depth accuracy

Besides the additional evaluation metric for semantic segmentation, another evaluation is conducted. By comparing different setups for the transformation to top view while using the same estimated segmentation (i.e. same output of the semantic segmentation network), the effect of depth accuracy is measured because this accuracy is the only difference in each approach.

Figure 3.24 shows two trajectory lines. The white trajectory line is the ground truth trajectory while the green line represents estimated trajectory based on the segmentation of the drivable path. The analysis of these trajectories is visualised by two curves in figure 3.25. The raw difference curve presents the actual difference between the ground truth and estimation throughout an increasing longitudinal distance from the vehicle. The second curve shows the scaled version of the raw difference, taking the significance of the error into account.

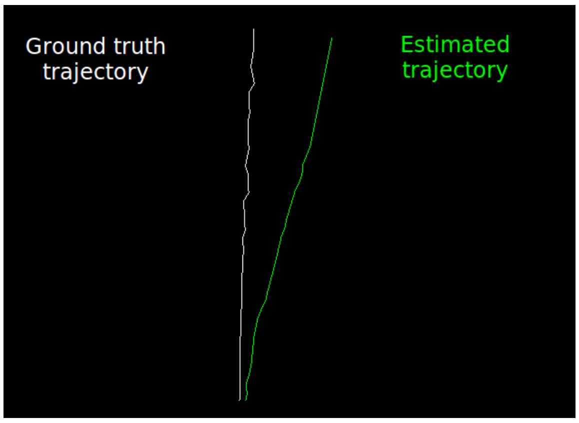


Figure 3.24: Visualisation of trajectories in top view. Green is the estimated trajectory while the white line represents the ground truth trajectory.

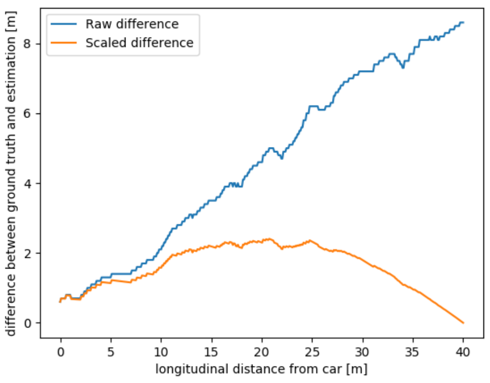


Figure 3.25: Graphical representation of the difference between estimation and ground truth.

3.4. Summary

To avoid the problem of high annotation times which are applicable for full semantic segmentation, a reduction of predefined semantic classes is proposed. This reduction simultaneously produces an approach which is very data efficient, utilising existing datasets to the full extent. The semantic classes are generated via a method proposed by [5] and results in weak ground truth labels. The label generation utilises data from the lidar and camera sensor and has the advantage that the pipeline can be automated, resulting in a computation efficient approach compared to full semantic segmentation. The robust network of SegNet is used to deploy semantic segmentation during test phase.

As highlighted in the chapter 2, domain adaptation is used to reduce the need for huge amounts of training data. However, this technique showed limited abilities when the performance was measured. This problem is referred to as domain shift and is assigned to the differences in data distribution of the domains. In order to get more insight in this problem, ten factors in three categories are examined. A comparison is achieved by the evaluation of these factors, mainly based on the Jaccard index and F_1-measure.

The last part of this chapter describes three setups that are used to transform from image space to top view space. Differences between these setups is quality of the depth information, used for the transformation. One setup is created where no depth information is used to transfer to top view. A second setup describes the possibilities when ground truth depth is available in the form of sparse depth maps and the last setups refers to the situation where depth is estimated via a separate CNN, resulting in a dense (estimated) depth map.

Two metrics are used in order to evaluate the importance of depth information in the process of transforming from image space to top view. One metric calculates the average lateral error between the ground truth and estimated trajectory while the second metric represents the overlap length between the trajectories.

4 Experiments and results

First, the experimental setup is discussed in section 4.1. Here, details about the implementation of the neural network will be discussed. Second, experiments which aim to describe the influence of different factors on the domain shift for the task of semantic segmentation are motivated in section 4.2. The chapter continues with the experiments on the transformation to a usable top view in section 4.3. This is followed by a discussion about the results of both the domain shift and the top view transformation in section 4.4. Conclusively, the chapter ends with a summary in section 4.5.

4.1. Experimental setup

All experiments are carried out by utilising SegNet, which is detailed in section 3.1.3. This convolutional neural network is deployed with a modified version of the Caffe deep learning framework [29] and all experiments are fulfilled using a NVIDIA GeForce 1080 Ti GPU. Caffe is a C++ library with Python and MATLAB bindings for training and deploying convolutional neural networks and other deep models. The framework links file, describing the network and indicating the hyperparameter values which are summarised below:

Parameter	**Value [RobotCar/KITTI]**	**Explanation**
test_iter	75 / 29	Indicates how many test iterations occur per test_interval.
test_interval	500	Indicates after how many iterations the test phase will be executed.
base_lr	0.01	Indicates the base learning rate of the network.
lr_policy	"step"	The learning rate decay policy.
gamma	0.9	Indicates how much the learning rate changes every step.
stepsize	2500 / 250	Indicates at what iteration the next step of training is activated.
momentum	0.9	Indicates amount of the previous weight that will be retained in the next step.
max_iter	13660 / 5300	Indicates at which iteration the network should stop training.
weight_decay	0.0005	Regularisation parameter.

Table 4.1: Settings for SegNet during the conducted experiments.

It has to be noted that when a parameter is referring to *test*, it actually refers to the validation phase. The framework of Caffe, however, addresses this as test phase.

Test_iter indicates how many test iterations should occur per *test_interval*. The *test_interval* represents the number of iterations after which the test phase is activated. *Base_lr* is the initial learning rate of the network. This learning rate will change over time and the fashion in which this happens is determined by *lr_policy*. In our case, the policy is "step" which means that the learning rate drops in steps sizes indicated by the *gamma* parameter. *Gamma* determines to what extent the *base_lr* should change for the next "step".

This policy can mathematically be expressed as:

$$\text{new\_lr} = \text{base\_lr} \cdot \text{gamma}^{\text{floor(iter / stepsize)}}, \tag{4.1}$$

where *iter* is the current iteration and *stepsize* indicates at what iteration we should apply the policy mentioned above and move on to the next "step" in training. *Momentum* is a parameter that defines how much of the previous weight will be retained in the new calculation. *Max_iter* sets the value at which iteration the network should stop training. Regularisation is addressed via the *weight_decay* parameter. This variable indicates the factor of penalisation of large weights and equals λ in equation 3.9.

Another parameter often used in neural networks and other frameworks is epochs. One epoch represents one forward and backward pass of all training images. With the parameters used in the solver file, the number of epochs is calculated as follows:

$$\text{Epoch} = \frac{\text{max\_iter} \cdot \text{batchsize}}{\text{\# training images}}. \tag{4.2}$$

In this equation, the batch size refers to the amount of training images in one pass. Values for batch size, filter depth, learning rate and offset are pointed in the network file instead of the solver file. Hence, these values can manually be adjusted for every layer.

4.2. Domain shift factors

The setup where all target domains are present in the training phase is often not possible because the target domain lacks labelled data. Therefore it is necessary to transfer knowledge from other domains to a certain target domain. While this is possible, figure 3.7 shows that the estimation of the drivable path is not satisfactory when the network is not trained on the target domain. Literature assigns this phenomenon to the domain shift, or stated otherwise, the difference in data distributions between datasets. In order to examine potential causes for this shift, several experiments are carried out, using datasets described in section 2.1.2. These experiments are divided into three categories based on what modifications are made. The category image refers to modifications purely based the image. The second category, geometry, refers to modifications concerning the geometry of the image. The remaining factors are modifications on the setup of the experiments and hence are assigned to the category setup. The table (4.2) below shows these categories, defines every experiment in a category and mentions the section where the experiment is discussed.

Setup		**Image**		**Geometry**	
Factor	**Section**	**Factor**	**Section**	**Factor**	**Section**
Dataset equalisation	4.2.2	Greyscale	4.2.4	Horizon shift	4.2.9
Order of training	4.2.3	Left-hand traffic	4.2.5	Cropping	4.2.10
		Gamma correction	4.2.6	Intrinsic consistent	4.2.11
		Histogram equalisation	4.2.7		
		Number of classes	4.2.8		

Table 4.2: Summary of the conducted experiments, divided into three categories.

The default workflow to examine domain shift factors is shown in figure 4.1. Again, the red rectangle ("CityScapes") in this figure represents the pre-trained network on CityScapes. However, the difference with the workflow in figure 4.2 is the lack of ground truth labels in one of the target domains (WEpod domain). Furthermore, the network is trained in two steps, one for each source domain instead of a mixed dataset containing all domains.

Based on which experiment is executed the modification elements (for training and test phase) in the workflow will be activated. What this element contains is described in each experiment separately and obviously differs per experiment. This whole process of label generation and modifications results in a training set for both KITTI and RobotCar. The pre-trained network is fine-tuned on these newly generated training sets. After the final fine-tuning, the resulting weights are used in SegNet for semantically segmenting the test set.

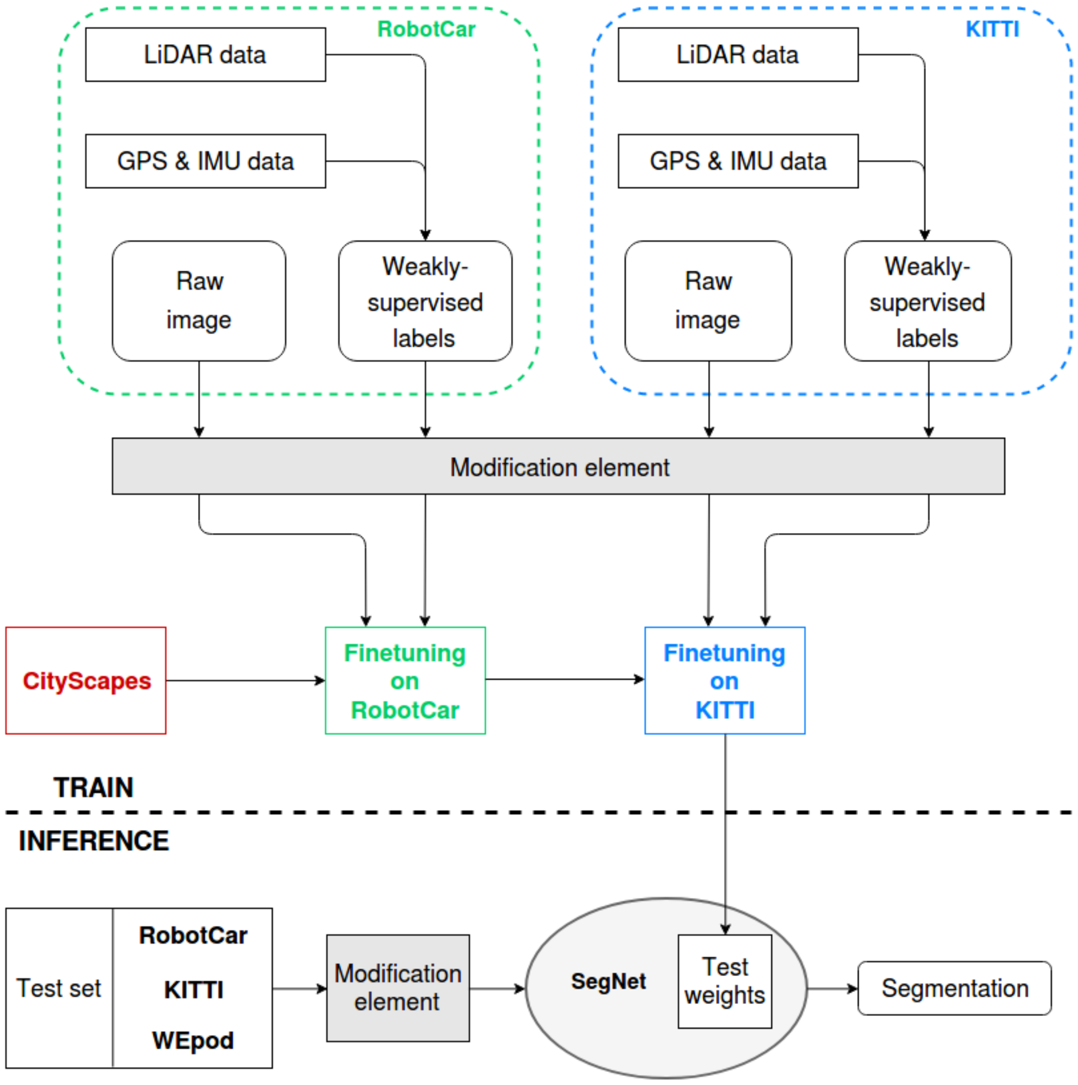


Figure 4.1: The default workflow of the applied approach. The green and blue dashed boxes represent the generation of weak labels.

This workflow contains a double fine-tuning step which is not generic. Fine-tuning usually only has one step. The choice of a two step fine-tuning approach is based on two reasons:

1. The two-step fine-tuning procedure enables the network to add training data and train the network only on the additional data which is very time efficient.
2. Intermediate results give more insight in the process of the neural network, enabling extra analyses. However, these analysis are not shown in this study.

In order to find the influence of different factors on the effect of the domain shift for semantic segmentation of road scenes, ten experiments are carried out. The methodology behind these experiments is explained in section 3.2.2. In the remainder of this section, the setup of every experiment will be described and the results are discussed, leading to some possible explanations for these outcomes. The presented results are summarised in table 4.3 with the best results per category (setup, image and geometry) highlighted in bold. These results all relate to the default setting, noted as "baseline" in table 4.3. They are noted such that values in 4.3 represent absolute evaluation results while tables 4.4 and 4.6 represent evaluation results relative to the baseline. Hence, negative values in these tables suggest a decrease in performance and positive values show an increase in performance.

4.2.1. Default network and upper bound

The default setting does not produce optimal results, which obviously is part of the problem. However, it is stressed that for the remaining part of the thesis it is not intended to obtain state-of-the-art results. The aim is to see performance gains or performance losses as a result of data modification according to experimental setups. Yet, in order to show to what extent the network is capable of achieving good results, an upper bound is created.

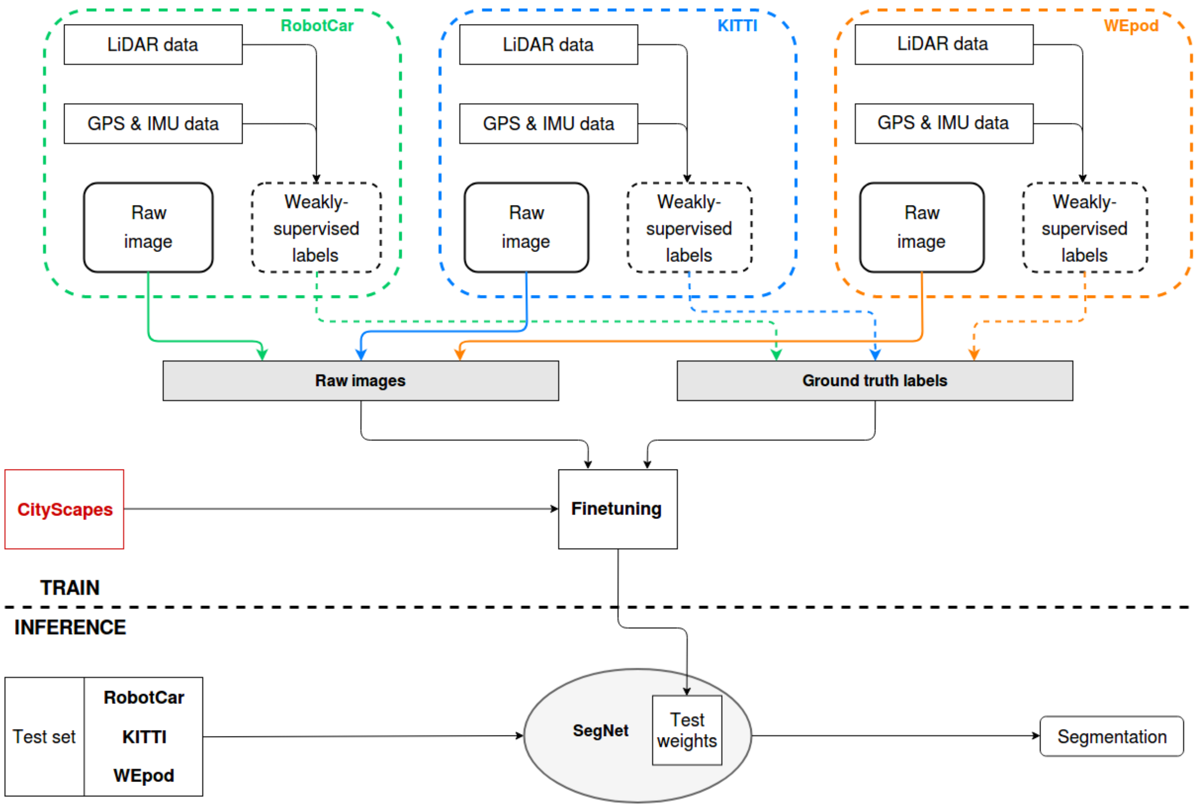


Figure 4.2: The workflow of the approach for the upper bound. The green, blue and orange dashed boxes represent the generation of weak labels. Note: The inclusion of the WEpod domain in the training phase marks a big difference with the default approach.

For the upper bound setup, the data from the target domains is combined in one large dataset with corresponding ground truth labels. Hence, all target domains are present in the training phase and the network is fine-tuned in one step, on the combined dataset. Figure 4.2 depicts the workflow for this upper bound setup. The red rectangle ("CityScapes") in this figure represents the pre-trained network on CityScapes. The process captured in the green, blue and orange dashed squares is the generation of weakly-supervised labels, as described in subsection 3.1.2.

Results of both the upper bound and the default setting are summarised in table 4.3, tagged as "upper bound". Although the results of the upper bound are not competitive with state-of-the-art, all domains achieve relatively good results.

	Metric	**Drivable path**			**Occupancy**		
	[%]	RobotCar	KITTI	WEpod	RobotCar	KITTI	WEpod
Misses		0	4	4			
Baseline	IoU	29	54	24	74	81	38
	F_1	44	66	37	85	89	45
Upper bound[1]	IoU	75	67	80	90	88	78
	F_1	83	78	88	95	93	87

Table 4.3: Evaluation values for the baseline and upper bound of the domain shift experiments.

4.2.2. Data equalisation

While the default network represents realistic situations with datasets having different sizes, it is possible that this difference in number of images influences the created feature space of the network. From the original RobotCar dataset, images are removed in a random fashion to decrease the size to 1060 training images. This size is equal to the size of the KITTI training set and hence, the KITTI training set is not adjusted. Since this experiment only comprises the removal of training images, modifying the size of datasets and leaving the modification element in figure 4.1 empty.

Results

This experiment shows some surprising results. The equalisation results in strong reduction of metrics for the segmentation of drivable path in the WEpod domain. Although the size of the RobotCar training set is reduced by roughly 61%, the metrics on the test set of RobotCar are increasing by a notable amount of 4 percent points for drivable path segmentation. As all other results, this result is found in table 4.4 and table 4.6. The segmentation of occupancies for RobotCar does decrease with the adjustment of the training set by a minimal margin of 3 percent points. Results on the drivable path segmentation on the KITTI domain increases with eight percent points for both IoU and F_1. However, occupancies are segmented more accurate for KITTI and WEpod resulting in an increase for the IoU and F_1 metric. The drivable path segmentation for the WEpod domain decreases drastically by 6 and 9 percent points for the Jaccard index and F_1-measure respectively. No rigorous changes in the number of drivable segments is observed in any domain.

Possible explanations

KITTI test images see an expected increase in evaluation metrics since the share of KITTI training images in the total training set is increased from 28% (1060 KITTI images of 3790 images total) to 50% (1060 KITTI images of 2120 images total) by equalising the dataset. Although a drop in performance was expected for RobotCar since less training data is available, the results show a slight increase of performance for drivable path. Because the removal of around 60% of the RobotCar dataset is too big to be explained as the accidental removal of *bad* training images, this increase is yet unexplained. For segmenting WEpod images, it is likely that training on RobotCar images is more valuable which leads to a decrease of performance for drivable path segmentation. However, this decrease can also be explained by the reduction of data in general.

4.2.3. Order of training

The baseline presented in table 4.3 show results for three domains. From these three domains, two are also present as a source domain (i.e. KITTI and RobotCar). When comparing results for these domains, it can be observed that the network achieves significantly better performance for the KITTI test images than RobotCar test images. In the default setup, the network is fine-tuned on the KITTI domain as a last step. The influence of this second fine-tuning step is examined in this experiment by changing the order of training. It is expected that this experiment will have effect on the segmentation of both KITTI and RobotCar since these domains are represented in the training phase but the WEpod domain remains unseen during training and influence on this dataset is tested. The modification element will stay empty as the only difference is the order of training while the training data is equal to the baseline.

Results

Influence on the source domains during testing are clearly shown in the results. The evaluation metrics increased by a great amount for the drivable path segmentation of the RobotCar test set. This increase of metrics comes at the cost of great reduction in IoU and F_1 for the drivable path segmentation in the KITTI domain. Furthermore, the occupancy segmentation follows the same trend as drivable path segmentation (increase for RobotCar; decrease for KITTI) but by a far less amount than for drivable path.

Results for the unseen domain of the WEpod show clear influence throughout the evaluation metrics. Drivable path segmentation show an improvement of 18 and 20 percent points over the baseline setup while the increase for the occupancy segmentation is high with 63 and 56 percent points for IoU and F_1-measure, respectively. Opposite to the source domains, the WEpod domain sees a bigger increase for the occupancy class than the drivable path class.

Besides these huge improvements in metrics, the experiment also effects the number of drivable path segments. The experiment is close to resolving this problem, only leaving one case where the drivable path segment is not recognised for the WEpod (originally there were four cases). In both the KITTI and RobotCar

[1] It should be noted that the test set of the upper bound differs from the test set used for the baseline.

dataset, all drivable path segments are detected and which is an improvement for KITTI (originally missing four cases) and a stable situation for RobotCar which also did not leave any drivable path segments undetected in the default setup.

Possible explanations

Because of the large differences in the evaluation metrics between the source domains, it is concluded that the network does not generalise well and is dependent on the order of training to produce good results. The WEpod data has more similarities with the RobotCar domain than with the KITTI domain, based on the enormous increase of all evaluation metrics for the WEpod domain. This conclusion agrees with the previous experiment which showed that reducing the size of the RobotCar training set led to worse prediction of the drivable path for WEpod. One of the similarities is the aspect ratio of the raw images. The raw images for WEpod equal 1024 x 512 pixels which is a lot closer to RobotCar (1280 x 960 pixels) than KITTI images (1242 x 375 pixels).

4.2.4. Greyscale

As noticed in the experiment on training order, the aspect ratio of the images from RobotCar and WEpod are relatively similar (i.e. compared to the similarity between KITTI and WEpod). However, there are still differences between the domains. One difference is the colour space of both domains, the WEpod domain consists of greyscale images while RobotCar contains RGB images. In this setup, the original RGB training images of both KITTI and RobotCar dataset are converted to greyscale images in order to match the target images from the WEpod domain. By means of this conversion, the pixel intensities are the only information that the image contains. The modification module is activated for both RobotCar and KITTI which leads to conversion to greyscale images. Obviously, the ground truth labels do not change and will be the same as in the default setting.

Results

This scenario resulted in a minimal increase for the IoU metric (one percent point) and no change in F_1-score for the drivable path segmentation compared to the default setting for the WEpod domain. Slightly bigger improvements are found for occupancies, resulting in an increase of four percent points for both IoU and F_1. In contrast to the results for the unseen domain of the WEpod, the results for KITTI and RobotCar are heavily influenced in a negative way. A decrease in IoU and F_1 is found for the drivable path segmentation in the KITTI domain (10 percent points for both IoU and F_1) and RobotCar domain (25 and 36 percent points for IoU and F_1 respectively). This decrease is also visible for the occupancy segmentation on RobotCar but shows a negligible (positive) influence on the occupancy segmentation of KITTI. The results relative to the baseline can be found in tables 4.4 and 4.6.

Furthermore, it is noted that the amount of drivable path segments dramatically decreases for the both RobotCar and KITTI when the network is trained on greyscale images. In other words, more often there is no drivable path segmentation in this setup than in the default setup. For RobotCar, 68 cases (27% of the RobotCar test set) are not recognised, compared to the original setup with zero failures. KITTI shows 21 cases where it does not recognise a drivable path which is a notable increase over the 4 cases in the setup. However, for the WEpod domain there is only the minimum difference of one in the number of drivable path segments, resulting in more drivable path segments during greyscale training. This setup only misses 3 cases while the default setup misses 4 cases.

Possible explanations

By changing from the RGB colour space to greyscale, three colour values are combined to result in one intensity value with the consequence of a loss of (colour) information. For RobotCar, the road is often reflective, leading to high intensity values when converted to greyscale. These high intensities are scattered spatially and hence do not form a good signature for a drivable path. For KITTI images the road is not as reflective as for RobotCar, however the contrast in the images, is relatively low. Therefore it is harder to detect patterns without colour information.

When the test set consists of greyscale images, it is assumed that a feature space based on greyscale images matches better than a feature space that is created based on RGB images. Therefore, it can be considered surprising that results on WEpod imagery only improves slightly.

4.2.5. Left-hand traffic

Another difference in the images is the fact that RobotCar is recorded in the streets of Oxford (England) where different traffic rules apply compared to the KITTI domain and WEpod domain. An example of a traffic rule which influences the layout of the image and ground truth label is the left-hand traffic rule. The modification element in the workflow contains a procedure in order to flip both training images and ground truth labels, only for RobotCar. Note that this will not change the RGB pixel values and corresponding ground truth label, only the spatial relation differs. KITTI does not experience any changes.

Results

WEpod test images see a notable increase in evaluation metrics for drivable path with 4 percent points. Occupancy segmentation of the WEpod test set is marked by a much larger increase of 39 percent points for both IoU and F_1. KITTI sees a slight decrease of IoU and F_1 for drivable path segmentation and occupancy segmentation of 3 (IoU) and 2 (F_1) percent points. For RobotCar, the decrease in metric values for drivable path segmentation is minimal (maximum of 1 percent point) while there is a notable increase for occupancies (5 and 9 percent points for F_1 and IoU respectively).

Influence is also shown in the drivable path recognition. Both KITTI and WEpod miss to segment drivable paths in 11 and 10 test cases respectively during this experiment while the original setup only miss 4 test cases. RobotCar is stable and, as in the default setup, recognises a drivable path in all test images.

Possible explanations

Generalisation is increased, based on the results for WEpod test images. However, results for KITTI and RobotCar see a slight decrease in performance. The fact that no large increase occurs for these datasets can be explained by the position of the drivable path in the training and test set. RobotCar is driving in a urban environment without clear boundaries for lanes.

Drivable path segmentation, seen from the point of view of the car, always starts from the middle of the image. Road positioning becomes important further away from the vehicle but at the same time, the effect of the experiment diminishes because of the decrease in spatial resolution. The increase of performance on occupancy segmentation for both KITTI and RobotCar is unexpected since the relation between occupancy label and pixel values did not change. One possible explanation for this event leads to the spatial positioning of occupancies. The place of occupancies in the image relate more to each other when the image and label are flipped.

4.2.6. Gamma correction

By applying a gamma correction on the images, the contrast is enhanced by a non-linear operation. This enhancement applies on the luminance value of every pixel in the image. In this experiment the value of γ is set to the value of 0.7 to resemble more to the KITTI dataset. Hence, the modification element applies the gamma correction to the raw images for both training and test images and does not modify the ground truth labels.

Results

Drivable path segmentation within the WEpod domain increases by 6 and 7 percent points for the IoU and F_1 metrics. A negligible increase for occupancy is observed. Both KITTI and RobotCar show completely different metrics as result of the experiment. The drivable path estimation drops with 5 percent points for KITTI while a decrease up to 6 percent points for RobotCar is observed. Where KITTI shows a negligible decrease for occupancies, a notable increase of 7 (IoU) and 5 (F_1) percent points is obtained for RobotCar.

Although, drivable path segmentation showed a decrease in metrics for RobotCar, it keeps the 100% detection of drivable paths, missing no segmentation. This is in contrast to KITTI where the experiment results in an extra 13 missed drivable path estimations (on top of the original 4). The WEpod domain is relatively constant, missing 5 drivable path estimations meaning an increase of only 1 missed recognition.

Possible explanations

The gamma correction only has limited effect on the RobotCar images due to the large amount of brightness values at the end (low or high) of the spectrum, while it is clear from figure 3.9 that the gamma correction has the largest effect in the middle of the spectrum.

4.2.7. Histogram equalisation

The experiment with histogram equalisation is used to modify the intensity values of the image in order to enhance the contrast using the image histogram. The histogram equalisation is applied on both training and test data and therefore the modification element will be activated in both phases. These elements change the raw images and leave the ground truth labels unchanged.

Results

In all three domains the experiment shows a decrease for the drivable path segmentation in all metrics. The quantity however, does differ notably throughout these domains. The largest decreases are noted for RobotCar, lowering the IoU and F_1 metric with 7 and 10 percent points respectively. This decrease is a bit higher than KITTI (5 and 4 percent points respectively). The decrease in evaluation metrics for the drivable path segmentation of WEpod test images is almost negligible equalling 1 and 2 percent point for IoU and F_1.

Similar negligible values are obtained for the occupancy segmentation of KITTI and RobotCar. Difference is that these values present an slight increase for the experiment. The WEpod domain however, sees a great improvement in occupancy segmentation, increasing with 24 percent points for IoU and 27 percent points for F_1.

The experiment did also show some negative effects on the detectability of the drivable path. All domains did recognise less drivable paths than in the default setting. RobotCar missed 4 drivable paths compared to no failures in the default while for KITTI the network did not estimate a drivable path at all in 15 cases while only 4 were missed in the default setting. The WEpod domain showed the worst results with 31 failures to detect a drivable path. This is 27 extra missed drivable paths compared with the default network.

Possible explanations

The contrast enhancement via histogram equalisation shows similar trends as the gamma correction for RobotCar and KITTI. Apparently, the contrast for the WEpod images is increased too much leading to negative influence on the drivable path segmentation. This can be the result of newly introduced *edges* because of the increased contrast.

4.2.8. Number of classes

For this experiment, the ground truth labels have to be modified since there is a reduction in number of classes. This also affects the weight values as calculated in equation 3.7 since the ratio between the classes is shifted. The modification element contains the adjustment of the ground truth labels and the recalculation of the imbalance weights presented in equation 3.7. This modification additionally has some consequences for the evaluation of the experiment. Where the normal setting interprets results on drivable path segmentation and occupancy segmentation, the latter is impossible in this case. There is no class which only contains occupancy and therefore, this segmentation is not evaluated. The only meaningful class in this experiment is the drivable path segmentation and hence only the results of these metrics will be interpreted.

Results

Results on the three different domains give three different insights. Evaluation on the RobotCar domain only sees a small (negligible) added value of the conversion to two classes. The WEpod test set is influenced heavily by the experiment, resulting in an increase of 8 percent points for both the IoU and F_1 metric. This result is close to the opposite effect of the influence on the KITTI dataset. KITTI sees a drastic decrease in metrics of 8 and 9 percent points for F_1 and IoU respectively.

Although the increasing evaluation metrics for the WEpod domain, drivable path segments are less often recognised during this experiment, 13 test cases fail to obtain drivable path where the original setting only had 4 missing estimations. Similar numbers can be presented for the KITTI dataset (failure of 16 images to originally 4 faults occurred) while for RobotCar, there are drivable segments for every test image and thus no changes are seen compared to the default network.

Possible explanations

This experiment results in a more simplistic feature space which is expected to lead to better output for the drivable path segmentation. This is clearly the case for WEpod test images and also happens in a very small amount for RobotCar. The experiment has significant negative influence on the drivable path segment of KITTI test images, that has not been explained.

4.2.9. Horizon

The last category ("Geometry") will modify image geometry instead of adjusting the setup or the image appearance. The first experiment within this category will shift the horizon for the WEpod domain. This experiment is the consequence of a big difference between the three datasets is the position (and hence orientation) of the camera on the recording platform. This difference is especially large between the WEpod platform and the other two platforms. This is due to the low camera height compared to the recording platforms of KITTI and RobotCar. Because of these differences, the vanishing point in the WEpod domain occurs at lower height in pixel coordinates than for KITTI and RobotCar. This difference is resolved in the modification element of the workflow depicted in figure 4.1. In this element, the horizon line is aligned to the horizon lines in KITTI and RobotCar by cropping the upper part of the image.

Results

Because the modification only applies on the WEpod test images, only the results on this part of the test set is highlighted, considering that results for both KITTI and RobotCar are the same as in the default setting. Aligning the horizon line throughout the domains shows positive results on the drivable path segmentation, increasing with 7 and 8 percent points for the IoU metric and F_1 metric respectively. An increase of evaluation metrics does also hold for the occupancy segmentation where a increase of 7 percent points for IoU is found. The F_1 metric shares the same trend, increasing by 7 percent points.

Although the drivable path segmentation increases by a notable amount, there is the minimum decrease of one extra missed recognition compared to the default setting, resulting in a total of five test images with no drivable path segmentation.

Possible explanations

Only cropping the top of the image will result in a shift of the principle point compared to the original situation. Consequence of the horizon line shift is a (small) change in the road layout. Because the road layout resembles KITTI and RobotCar more, an increase of evaluation metrics for the WEpod domain is seen.

4.2.10. Cropping

Resizing training images to a fixed resolution will result in deformations of the objects in the image. Because the initial image size differs among the domains, similar objects (e.g. cars) will be deformed differently. In order to overcome this problem, the modification element in the workflow contains a cropping mechanism at the cost of loss of information. In training phase, this only influences the images of KITTI. The original size of 1242 by 375 pixels is cropped to the fixed input size of the neural network, 480 by 360 pixels. This results in a loss of information on the sides of the image and the top of the image. A similar procedure is followed for the WEpod domain (test set). The WEpod test images are first cropped to the fixed aspect ratio for the network while afterwards, the test images of the WEpod domain are resized to the fixed input size of the network. RobotCar data will not have any changes since the aspect ratio of the raw images and input images is the same. Therefore, the procedure for RobotCar images is the same as for the default setting.

Results

In terms of drivable path segmentation, the unseen domain of the WEpod sees a great increase in both IoU and F_1 which equal 9 and 11 percent points respectively. An even larger increase in the IoU and F_1 metric is shown in the KITTI (both 13 percent points) and RobotCar (both 27 percent points) domain. In case of RobotCar, this is almost an increase of 100%. This experiment however, also shows a decrease for the occupancy segmentation in the WEpod domain of 11 and 15 percent points for IoU and F_1 respectively. This is in contrast with the increase of IoU and F_1 with 8 and 5 percent points for the RobotCar domain. Occupancies in the KITTI domain experience a small decrease in IoU and F_1 of 3 and 2 percent points respectively.

These great increases of drivable path segmentation come at the cost of slightly less annotated drivable paths. For the WEpod domain, this only equals one less (5 in total against 4 original misses) detected drivable path while for KITTI 11 paths are not recognised (which originally are 4 misses). RobotCar domain is stable and recognises all drivable paths (as in the original setting).

Possible explanations

By cropping the KITTI training images to the same aspect ratio, the network will learn features that are similar for both KITTI and RobotCar and is thus more generic because the spatial environment resembles more.

4.2.11. Intrinsic consistency

By adjusting the training set such that it is intrinsic consistent, the differences between the various recording platforms based on the camera characteristics and geometry of the camera are removed. In this experiment it is tested whether the inconsistency caused by the usage of different recording platforms is the root cause of the domain shift problem. The modification element is activated and consists of the three steps, mentioned in section 3.2.2. This applies for both training and test data and effects the raw images and the corresponding labels.

Results

RobotCar and WEpod show a similar trend in the evaluation metrics in this experiment. RobotCar increases by 7 and 8 percent points for IoU and F_1-score respectively while for the WEpod domain these metrics are both increased by 10 percent points. These values are referring to the drivable path segmentation. For KITTI there occurs a drop in performance for the drivable path segmentation of 7 and 5 percent points. All domains have an increase in the evaluation metrics (both IoU and F_1) for the occupancy segmentation ranging from 2 percent points for KITTI to 9 percent points for RobotCar, up to 31 percent points for WEpod test images.

The three domains are affected by the experiment for the detection of drivable paths. RobotCar does not recognise a drivable path in 3 test images while 6 of the WEpod test images are without a drivable path segmentation. The most failures occur in the KITTI domain, where this experiment results in 13 test cases without drivable path segmentation. For all domains this is an increase of detection failures. RobotCar originally does not miss any drivable path segmentation while both the KITTI and WEpod domain miss a drivable path segmentation for 4 test cases.

Possible explanations

Although KITTI has a notable drop in the performance, this experiment shows that the results are more equalising throughout the three domains. Because the domains are intrinsic consistent, the only differences will be found in the environmental changes.

Jaccard index [Δ%]	**Drivable path**			**Occupancy**		
	RobotCar	KITTI	WEpod	RobotCar	KITTI	WEpod
Dataset equalisation	+4	+8	-6	-3	+2	+6
Order of training	+51	-37	**+18**	+19	-3	**+63**
Greyscale	-25	-10	+1	-11	+1	+4
Left-hand traffic	0	-3	+4	+9	-3	**+39**
Gamma correction	-4	-5	+6	+7	-1	+1
Histogram equalisation	-7	-5	-1	+2	+1	+24
Number of classes	+1	-9	**+8**	n/a	n/a	n/a
Horizon	n/a	n/a	+7	n/a	n/a	+7
Cropping	+27	+13	+9	+8	-3	-11
Intrinsic consistency	+7	-7	**+10**	+9	+4	**+24**

Table 4.4: Jaccard index values compared to the baseline for every experiment. Δ% refers to percentage points compared to this baseline. The dash line distinguishes between three categories: setup, image and geometry as mentioned in table 4.2.

Factor	**RobotCar**	**KITTI**	**WEpod**
Default	0	4	4
Dataset equalisation	1	12	4
Order of training	0	0	1
Greyscale	68	21	3
Left-hand traffic	0	11	10
Gamma correction	0	17	5
Histogram equalisation	4	15	31
Number of classes	0	16	13
Horizon	n/a	n/a	5
Cropping	0	11	5
Intrinsic consistency	3	13	6

Table 4.5: Number of missed drivable path segments. In these cases, the network does not estimate a drivable path at all.

$\mathbf{F_1}$**-score** [Δ%]	**Drivable path**			**Occupancy**		
	RobotCar	KITTI	WEpod	RobotCar	KITTI	WEpod
Dataset equalisation	+4	+8	-9	-2	+1	+7
Order of training	+41	-38	**+20**	+12	-2	**+56**
Greyscale	-36	-9	0	-8	0	+4
Left-hand traffic	-1	-2	+4	+5	-2	**+39**
Gamma correction	-6	-5	+7	+5	-1	0
Histogram equalisation	-10	-4	-2	+1	0	+27
Number of classes	+1	-8	**+8**	n/a	n/a	n/a
Horizon	n/a	n/a	+8	n/a	n/a	+7
Cropping	+27	+13	**+11**	+5	-2	-15
Intrinsic consistency	+8	-5	+10	+6	+2	**+31**

Table 4.6: F_1 metric values compared to the baseline for every experiment. Δ% refers to percentage points compared to this baseline. The dash line distinguishes between three categories: setup, image and geometry as mentioned in table 4.2.

4.3. Top View

As motivated in section 3.3, the transformation to top view is important. These experiments will result in two additional evaluation metrics. The first evaluation entails a check on the quality of the drivable path segmentation. The second and last evaluation results in the importance of ground truth depth for the transformation to top view.

Because ground truth depth maps are needed for several setups, only the KITTI dataset is used. Both RobotCar and WEpod have a lidar sensor mounted which can obtain ground truth depth but the mounted lidar is a 4-beam lidar which does not suffice in order to create depth maps. KITTI however, has a 64-beam lidar and hence results in the best possible (sparse) depth maps. A selection from the KITTI test set, used in experiments for the domain shift, is created resulting in 150 images. In this way, every category (road, city, residential) is represented by 50 images.

Throughout the top view experiments, a CNN is used to deploy drivable path segmentation. Since the test data for these experiments only exists out of images of the KITTI domain, the training phase of the CNN is implemented such that segmentation results are better for KITTI than the baseline in table 4.3. This KITTI training set is however, the same training set as used for the other experiments of domain shift.

4.3.1. Without depth

The most basic setting to convert to a top view entails a simple transformation without the need for depth information. It is important to use the same transformation parameters for both the ground truth and the network output of the test images. This results in a clean comparison between the ground truth and estimated drivable path. This comparison acts as an extra metric to see if the estimation is satisfactory. In order to evaluate the quality, four additional metrics are calculated and are shown in table 4.7.

Within the three categories, the road element shows the best performance in all metrics. Both raw error and scaled error have their minimum for road when comparing with the other two categories. The raw error is 0.68 metre for road while it is notably higher for city (0.93 metre) and residential (2.36 metre). The scaled error has a similar trend for all categories.

Additionally, the trajectory which follows from the estimated drivable path is about 90% of the length of ground truth trajectory. This is much better than the 73% and 75% for city and residential respectively. For the count parameter, the road category has all test images classified with a drivable path where for city 49 out of 50 are classified. Residential images are lacking four drivable path estimates, classifying 46 out of 50.

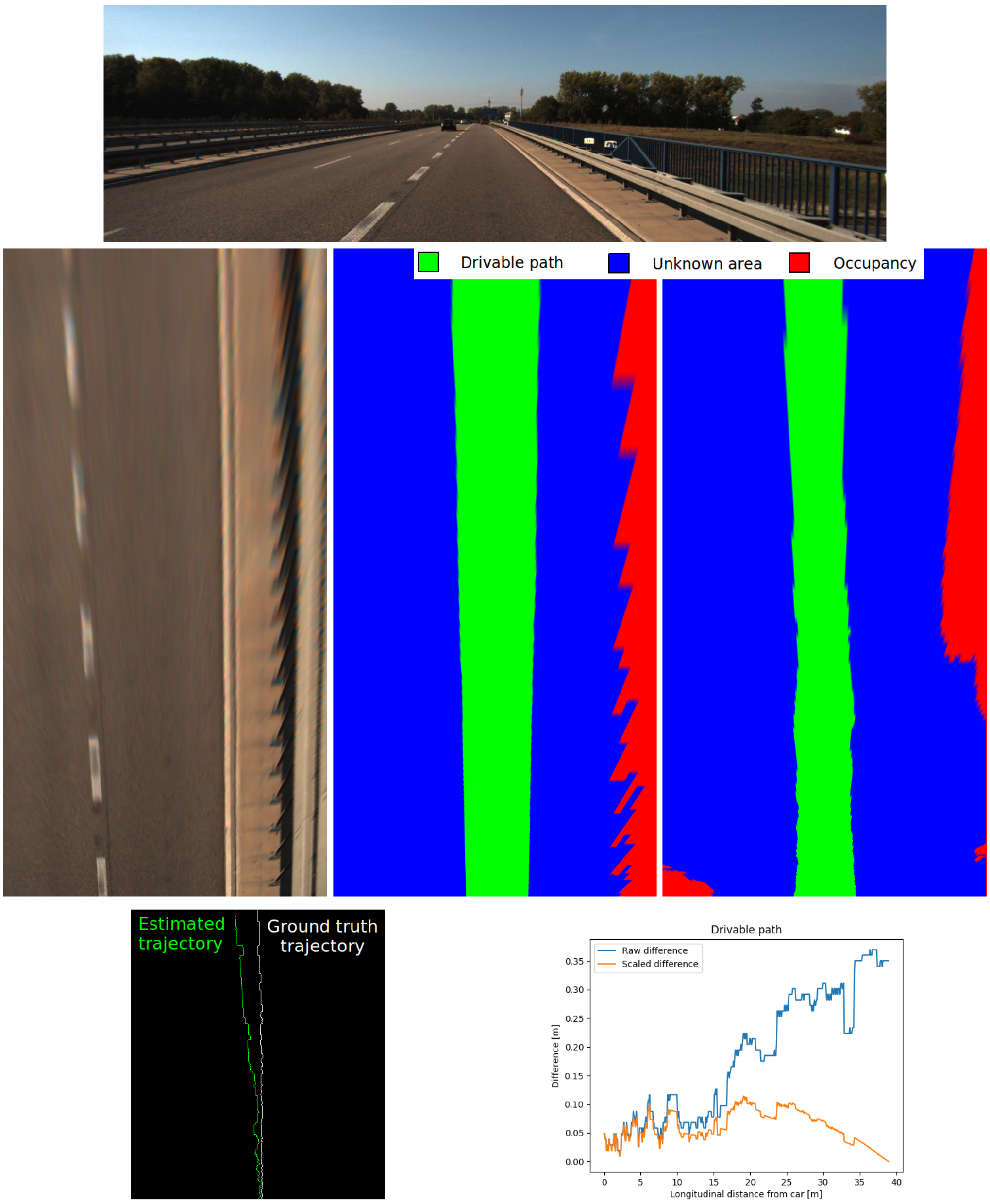
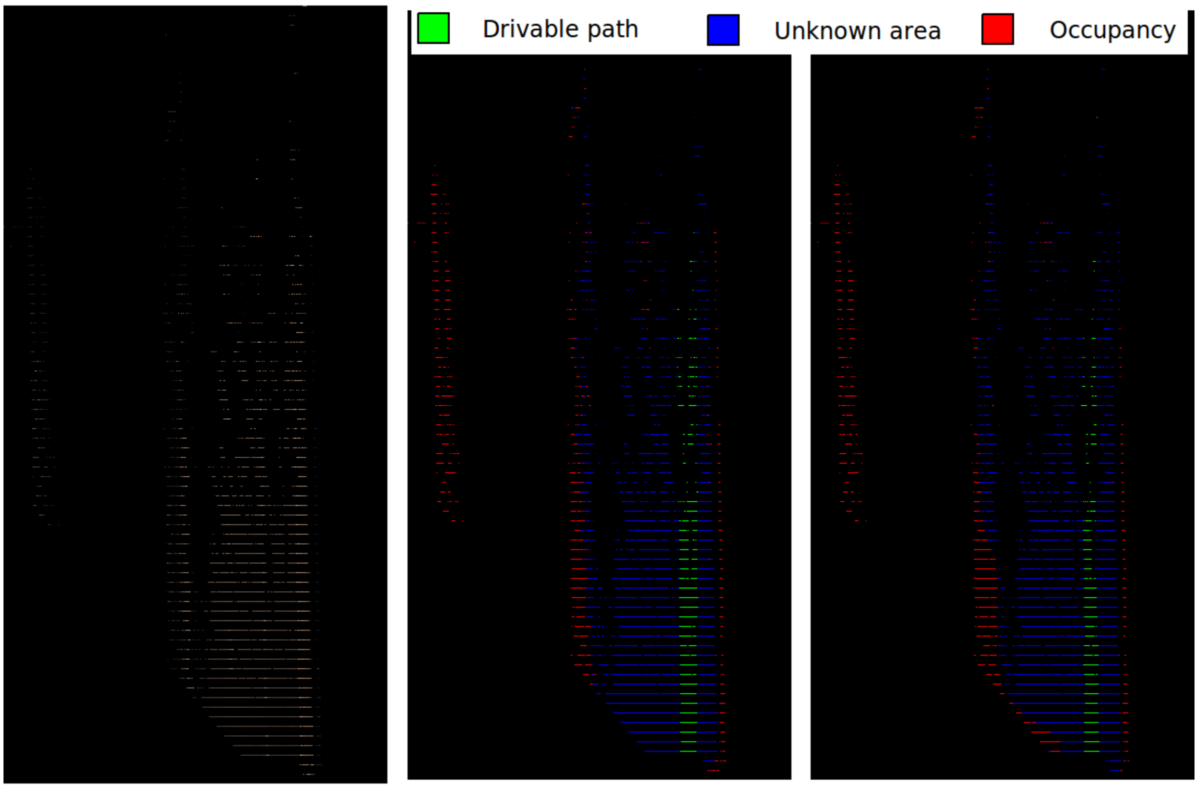
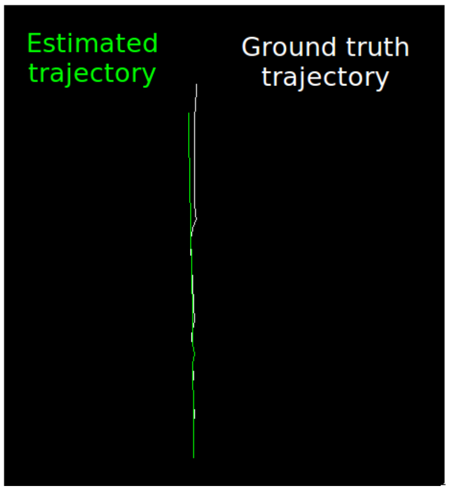


Figure 4.3: The top row show the original KITTI image. The middle row consists of three top views: top view image (left), top view ground truth (middle) and top view output (right). The bottom row consists of two images related to the evaluation. The left image depicts the top view trajectories and the graph on the right shows the raw and scaled error relative to the longitudinal direction.

4.3.2. Ground truth depth





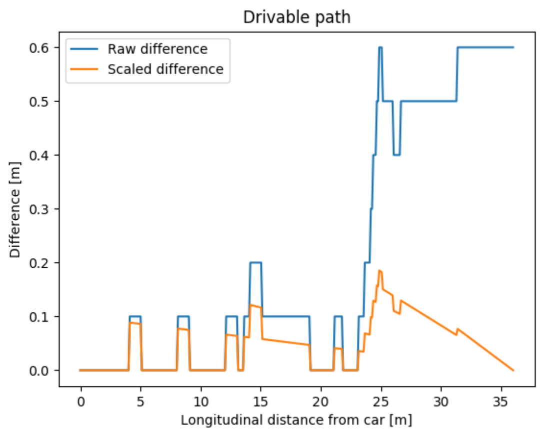


Figure 4.4: The top row show the original KITTI image. The middle row consists of three top views: top view image (left), top view ground truth (middle) and top view output (right). The bottom row consists of two images related to the evaluation. The left image depicts the top view trajectories and the graph on the right shows the raw and scaled error relative to the longitudinal direction.

When depth information is available, the transformation to top view can be achieved utilising this information. When ground truth depth maps are obtained from lidar sensors (as in this case), the resulting depth maps are sparse.

For the raw and scaled errors the same observations as for the first experiment are found. The road category outperforms the other two categories however, the difference with the city category is decreased. Where the raw error of the road category equals 0.61 metre, the city category only has a slightly bigger error resulting in a raw error of 0.64 metre. Residential images perform considerably worse with a raw error of 1.86 metre. The scaled error has a similar trend for all categories.

The trajectory length of the residential category is the largest (80% of the ground truth trajectory length) out the three categories. This is realised based on 44 out of the 50 test images for residential category, meaning it is missing six drivable path estimations. Road imagery achieves 73% of the full trajectory length (obtaining a drivable path in top view for 49 out of 50 images) while city images only reach to 61% of the ground truth trajectory (obtaining a drivable path in top view for 44 out of 50 images).

4.3.3. Estimated depth

Often, depth information is not available. The transformation to top view can still be made by a simple perspective transformation as shown in section 4.3.1. However, it is also possible to estimate depth based on images, resulting in a dense depth map. This is explained in section 3.3.4.

For this approach, the road category obtains the best results for raw and scaled errors. The raw error equalled 0.60 metre while the scaled error is 0.24 metre. These metrics are notably better than the metrics for the city test set which resulted in 0.82 metre and 0.34 metre for raw and scaled error respectively. Drivable path estimates in residential images score the lowest based on the proposed metrics with a raw error of 2.04 metre and a scaled error of 0.76 metre.

The trajectory length of residential images is the highest with 77% of the ground truth trajectory estimated. The drivable path trajectory lengths are similar for road and city images obtain roughly 68% and 69% of the full trajectory length. Except the lengths, the amount of drivable paths are also similar for these test sets, obtaining 49 out of 50 segments for the road images and 48 out of 50 city images. The residential set has 45 out of 50 drivable path estimates.

For the evaluation metrics of semantic segmentation, one trend can be discovered. For all approaches, the proportions within every test set are similar. In all cases, the road category obtains the best results, looking at raw and scaled error. Differences based on these two parameters are large between residential and the other two categories.

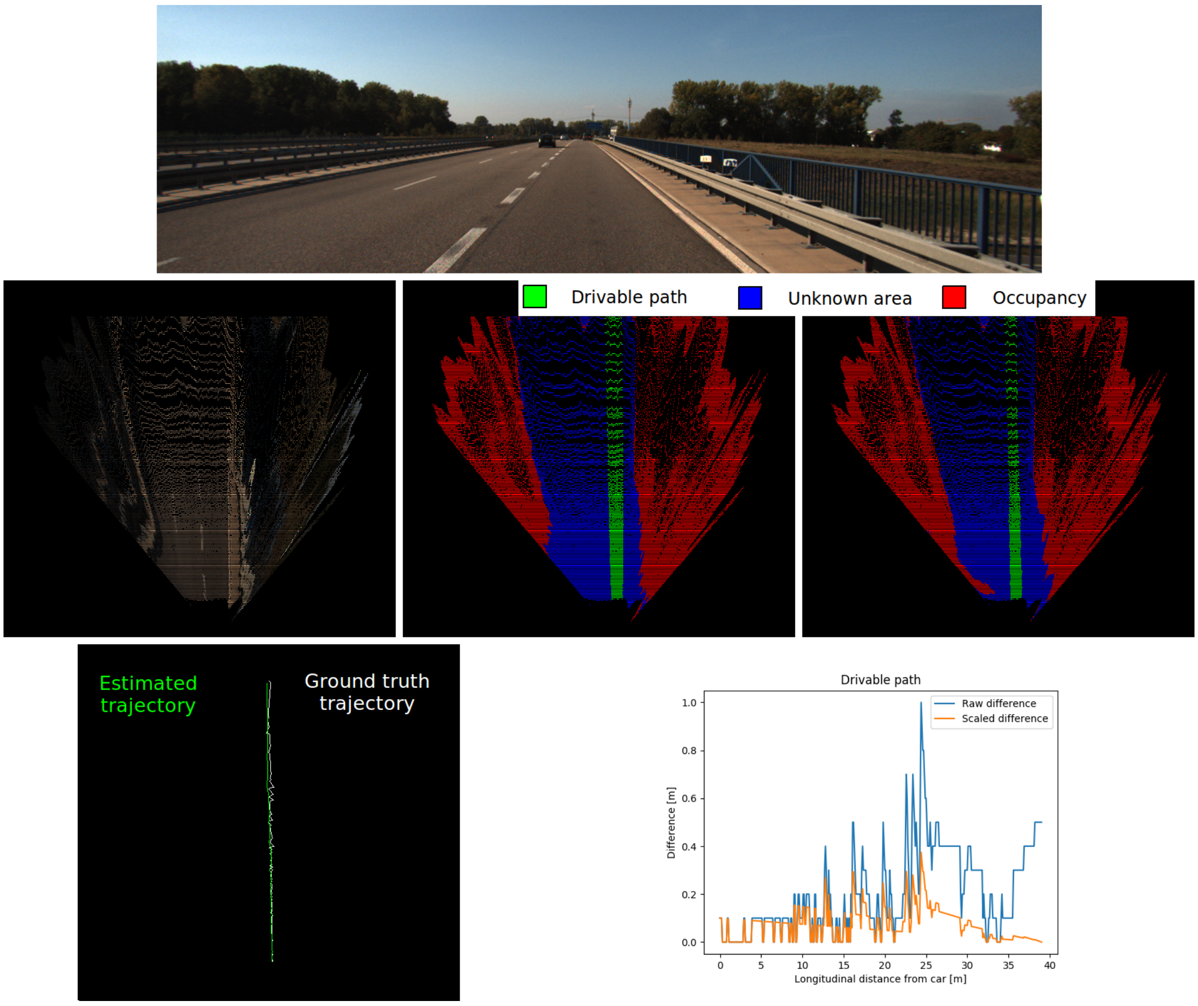


Figure 4.5: The top row show the original KITTI image. The middle row consists of three top views: top view image (left), top view ground truth (middle) and top view output (right). The bottom row consists of two images related to the evaluation. The left image depicts the top view trajectories and the graph on the right shows the raw and scaled error relative to the longitudinal direction.

Category	Setup	Raw error [m]	Scaled error [m]	Overlap length [%]	Count
Road	Without depth	0.68	0.28	**90**	**50**
	With ground truth depth	0.61	0.25	73	49
	With depth estimating CNN	**0.60**	**0.24**	68	49
City	Without depth	0.93	0.42	**73**	**49**
	With ground truth depth	**0.64**	**0.27**	61	44
	With depth estimating CNN	0.82	0.34	69	48
Residential	Without depth	2.26	0.93	75	**46**
	With ground truth depth	**1.86**	**0.72**	**80**	44
	With depth estimating CNN	2.04	0.76	77	45

Table 4.7: Evaluation values for the top view experiments. For the error: less is better. For the overlap length and count: higher is better.

4.3.4. Depth accuracy

By comparing the results from the different approaches, the effect of depth accuracy is analysed. The approach where no depth information is used in the transformation to top view, results in the lowest evaluation metrics out of the three approaches. For the road category an increase of roughly 13 percent compared to the best performance (i.e. approach where depth is estimated via a CNN) is noted. With a difference not topping 2 percent, the other two approaches are really close but the performance in the road category is best for the approach with the depth estimating CNN.

For the city category the difference between approaches is larger. Utilising ground truth depth information leads to the best performance, only slightly less accurate than for the road category (0.64m compared to 0.61m). Using the depth information obtained from a CNN, results in a decrease of 28 percent compared to the ground truth depth method. Again, the transformation without the use of depth information results in the worst performance, decreasing with 45 percent compared to the ground truth depth approach.

Similar results are shown in the residential category, the hardest category out of the three. The approach where ground truth depth is used, computes the transformation the best out of the three options, ending with a raw error of 1.86m. An increase of more than 9 percent for the raw error compared the ground truth depth is found for the estimated depth approach. Roughly 21 percent is the loss of performance when no depth information is used when transforming from image space to top view space.

In table 4.7 four parameters are listed. *raw error* denotes the average difference over all the trajectories where *scaled error* refers to the scaled error between ground truth and estimated drivable trajectory over all trajectories from the test set. This results in one value for every category. The *Length* parameter is expressed in percentage referring to the amount of overlap between the ground truth and estimation in longitudinal direction. The last parameter is *Count* which entails the number of drivable paths estimated, where 50 is the maximum.

4.4. Discussion

During this study, the experiments were conducted based on SegNet, an encoder-decoder network. No other architectures are used to conduct experiments. The problem of the domain shift is not solved by deploying another network since the data contains differences and therefore, is not dependent on the network. However, it would be interesting to what extent the conclusions based on the experiments will coincide. This would rule out any network architecture related issue and hence give more insight in the problem of domain shift.

Synthetic data is becoming increasingly important for improving current state-of-the-art results. The potential of synthetic data is lies in the possibilities of high variety of scene appearances and the ease of generating ground truth. However, the latter is possibly cancelled out by the generation of realistic virtual worlds, which is time consuming [27]. A few possibilities are already existing datasets/games such as GTA V, TORCS, SYNTHIA and CARLA. Synthetic data could be exploited as base for semantic segmentation. The research for influencing factors would be a lot broader since the domain shift also consists of the transformation from synthetic data to real-world data. However, due to time constraints, synthetic data is not utilised in this study.

4.4.1. Drivable path segmentation

Currently, the WEpod does not use images in order to predict its path. The WEpod drives around in a point cloud which is obtained beforehand. Based on this point cloud, the WEpod can locate the static objects of the environment. Within this "static map", a trajectory is planned. When the WEpod is driving in autonomous mode, this trajectory is followed as long as the lasers and radars do not detect any additional obstacles that are predicted to cross the trajectory in the near future.

During the generation of the ground truth labels, the ground scatter is removed from the point cloud. In the process of the removal, a few manually adjustable features (e.g. length of analysed sequence, obstacle height) are set to a threshold. These thresholds are determined on the basis of the domain where the labelling process is applied. As mentioned in section 3.1.2, the amount of future IMU positions that are taken into account equals 50. This number is set to be the best amount when considering the tracks in our training set.

Another parameter which is manually set, is the margin that is taking into account for determining whether points are located on an obstacle or on the road plane. Because of possible inaccuracies in road layout (i.e. ditches or natural slope), there is a margin which defines to what extent points are considered as road. This margin is set to 0.10 metres and based on the training sets for which the labels are generated.

In urban areas, GNSS is not always capable of retrieving a good localisation due to (high) buildings that prevent the signal from reaching the vehicle. The problem of blocked GNSS signals is often referred to as urban canyon. Signals that do reach the vehicle are often reflected of buildings and hence giving positioning of low quality. This error is referred to as multipath. These errors reflect on the drivable path segmentation because the car's position of future poses is used, potentially leading to bad ground truth labels. This is, however, averted by a threshold on the accuracy of the measurements. The threshold of accepted measurements is managed by a parameter, that is set manually.

4.4.2. Domain shift

Throughout this study, the factors are subdivided into one out three categories: network setup, image appearance and geometry. In general, the effect on the drivable path segmentation for the WEpod domain is positively influenced the most by the geometry category. However, this general trend is quantitatively topped by the *order of training* which is part of the network setup category and affects the drivable path estimation the most of all factors.

Factors in the image appearance category often have limited effect on the drivable path segmentation although *histogram equalisation* and *left-hand traffic* are two factors that have major positive influence on the occupancy segmentation of the WEpod domain. The change in the setup where the network is first fine-tuned on KITTI and as a second step fine-tuned on RobotCar. Both drivable path segmentation as occupancy segmentation of the WEpod domain see a huge increase in evaluation metrics due to the change in setup.

During the evaluation, two separate metrics are used: Jaccard index and F_1-measure. Analysing both metrics, it is noticed that, although the absolute values differ, they both follow the same trends. However, in one case the difference in absolute values show a different "leading" factor within a category. In this study, generating images with corresponding labels which are intrinsic consistent show to be the lead factor in the geometry category, based on the Jaccard index where the cropping factor is considered leading factor by the F_1 metric.

Analysing one inference image with corresponding ground truth, both IoU as F_1 metric are always positively correlated. Therefore, when the IoU metric *prefers* case "intrinsic" over case "cropping", the F_1 metric will do the same. However, there is a possibility this changes when the average score over a set of inference images is taken. There is a difference between the Jaccard index and the F_1 metric in quantifying the errors in case "intrinsic" and in case "cropping". In general, IoU metric will penalise single instances of bad segmentation more than the F_1 score and hence, over a set of averages, the IoU score results closer to the worst case performance while the F_1 measure is closer to the average performance. This potentially guides to a different "leading factor" based on the IoU and F_1 metric.

When obtaining average values of over a large set of inference images, both metrics have the same drawback. They both overestimate the importance of (very) small classes. As illustrating example, in an extreme case of semantic segmentation, if an image only has a single pixel of some class, and the network classifies that pixel correctly but also one other pixel wrongly, its F_1 score is 66% and the IoU is even worse at 50%. These severely punished mistakes can play an important role when averaging metric values over a set of inference images. In other words, the metrics weight each error on pixel level inversely proportional to the number of relevant pixels instead of treating them equally.

Another notable result is seen for *left-hand traffic* where a large increase of the occupancy segmentation is noted. This is surprising because the relation between the pixel values (RGB values) of the original image and corresponding ground truth label value are not changed. The spatial relation between pixels did change. This would implies that the network is dependent on spatial locations in the image which is not the goal of the neural network which is assumed to recognise obstacles independent of the spatial location in the image.

Because of the used approach, influence of the factors on seen (RobotCar and KITTI) domains can be separated from unseen (WEpod) domains. In the image appearance category, *histogram equalisation* is the only factor that results in similar trends throughout all domains and for both the classes. *Cropping* also shows this for the drivable path class.

4.4.3. Transformation to top view

From the results in table 4.7, it is concluded that depth information does play an important role for the generation of a correct drivable trajectory. Depth plays a more important role for the lateral error than for the longitudinal error. Moreover, the approach which does not use any depth information has a better longitudinal performance than approaches which utilise (ground truth or estimated) depth information for the transformation.

When the results of lateral error are split for the three categories (i.e. residential, city and road), the presence of depth information in the transformation from image space to top view space is most important for the residential and city category. The road category also notices improvements but these are minimal compared to the other two categories. This is intuitive since the road category has the least amount of turns in the drivable path, resulting in a relatively easy interpretable environment.

The evaluation in top view space has two related metrics: raw error and scaled error. The raw error is gives a good overview of the difference between the estimated trajectory and the ground truth trajectory. However, it does not take into account that the significance of the estimation drops proportional to the longitudinal distance from the vehicle. Therefore, the raw error is scaled. A simple scale is created which represents the importance of the location by a value. These values range from one to zero and are limited to the length of overlap between the ground truth and estimation.

When analysing the results of the raw error and the scaled error, the same trend throughout the results can be seen. Hence, the same conclusions are drawn based on the raw error as based on the scaled error. This is opposed to the evaluation metrics of the domain shift factors where differences between the Jaccard index and F_1-measure led to a discrepancy in one occasion. The fact that the raw error and scaled error in the top view transformation *tell the same story* means that the behaviour in longitudinal direction is similar. Most of the trajectories start relatively accurate while diverging further from the vehicle. This divergence does not necessarily happen at the same absolute longitudinal distance from the car but it does occur at the same part of the full trajectory.

A critical note on the results is the absence of the requirement on a continuous lateral drivable path estimation in order to create a top view trajectory. This potentially results in better overlap length than would be considered useful.

4.5. Summary

This chapter begins with a small description of the experimental setup. This part explains the used hyperparameters of the network. The remainder of the chapter is dedicated to the experiments which are split up into two parts: domain shift factors and top view transformation. The examined domain shift factors are organised such that first it is briefly mentioned what the experiment executes. Afterwards, the results are described and the last part of each experiment consists of a discussion about the possible explanations for these results.

Based on the drivable path of the WEpod domain, factors which show influence on the domain shift are intrinsic consistency, cropping and horizon shift. Other factors only show minimum influence on the drivable path segmentation such as converting to greyscale or even negative influence such as contrast enhancement via histogram equalisation. Reversing the training order is the factor which has the largest positive influence on the drivable path segmentation. This indicates that the RobotCar and WEpod domain are more similar than the KITTI and WEpod domain.

As discussed in chapter 3, a transformation from image space to top view is necessary to use semantic segmentation to its full potential. However, a range of possibilities exists to obtain this transformation. How these transformations differ, depends on the depth information that is used in the conversion to top view: no depth information, ground truth depth information (lidar) or estimated depth information. From the experiments it is concluded that the setup with ground truth depth information leads to the best performance concerning lateral error in two out of three categories (city and residential). For the road category, the best lateral performance is acquired by the transformation without depth information.

5
Conclusions and recommendations

During this thesis, the domain shift between datasets for the task of semantic segmentation is examined by deploying several experiments. Additionally, different setups for the transformation from image space to top view space are examined. In this chapter, the main conclusions of the five research objectives will be presented section 5.1 and suggestions for further research are proposed in section 5.2.

5.1. Conclusions

The conclusions noted in this chapter are organised such that they relate to the main research question and the five sub questions that were determined in chapter 1 and are given in the following subsections.

5.1.1. Deep Learning

How can deep learning be used for estimating drivable paths and what is the role of domain adaptation?

Semantic segmentation of images is a significant method for scene understanding in road scenery. During recent years deep learning, often in the form of convolutional neural networks, has become a dominant tool for semantically segmenting images, exceeding performance of the existing methods based on hand-crafted features. Convolutional neural networks learn feature extraction during the training phase by relating ground truth labels (containing drivable paths) to training images. These features are utilised during test phase, estimating the drivable path from a test image.

A downside of neural networks is that it is fully dependent on the data used during training. Especially the amount of data which is needed in order to obtain state-of-the-art results. Transfer learning is one approach to tackle this problem. It aims to transfer *knowledge* gained from one (source) problem to a different but related (target) problem. In this thesis, a problem consists of a task and a domain. One specific part of transfer learning, called domain adaptation, aims to successfully transfer knowledge assuming the same (source and target) task is deployed in a different domain. Due to this transfer of knowledge, a neural network is able to obtain good drivable path segments without the need of enormous amount of additional data, making the system data efficient.

5.1.2. Efficiency

How to utilise the vast amount of data that is already available?

As concluded from the first sub question, domain adaptation can be used to reduce the required size of the training set. However, this form of training still requires ground truth labels which are often not available in sufficient quantities. To address this problem it is essential to efficiently use the available laser and camera sensor data. This is achieved by creating ground truth labels with limited accuracy and a limited number of classes. Based on the point clouds from the lidar sensor and images from the camera sensor, occupancies can be projected into a label. Additionally IMU and GNSS data is used to construct drivable path segmentation. Due to the difference in quality of different datasets, mainly determined by the number of lidar planes, the amount of details (i.e. number of classes) in the ground truth labels is limited. However, this procedure results in a computation efficient method to obtain labelled data since the pipeline can be completely automated.

5.1.3. Domain adaptation

What factors influence a successful domain adaptation for the task of domain adaptation?

Concluded from the first sub question, domain adaptation can be useful and potentially essential to achieve state-of-the-art results in real world situations where data is limited and not perfect. Successful domain adaptation is defined as the process of transferring *knowledge* of a single task from one domain to a different domain where no drastic drop in evaluation metrics occurs. A range of potential factors is examined and based on the performed experiments, several conclusions are stated.

The cropping experiment where KITTI images are cropped to the same aspect ratio as RobotCar images is successful for the drivable path segmentation of all domains. By the cropping modifications, the feature space describes the features in more generic way such that the classes can be distinguished better. Establishing a training and test set which are intrinsic consistent removes the differences between the recording platforms with respect to camera parameters. This did have positive influence on the WEpod and RobotCar domain, increasing the evaluation metrics for the drivable path segmentation. However, these improvements did not lead to a fundamental change of the results. The shift of the horizon line is also examined and found to be a factor which will have a positive influence on the drivable path segmentation.

Besides these adjustments based on geometry, modifications on image appearance are made. However, these modifications did not have as much effect on the drivable path segmentation as previously mentioned geometry based adjustments. Contrast adjustment by means of histogram equalisation even results in a decrease of performance for drivable path segmentation over all domains. Applying gamma correction results in similar decrease for the RobotCar and KITTI domain but achieves an improvement for the drivable path segmentation of the WEpod domain. The influence of the left-hand traffic in the RobotCar domain is small for the drivable path segmentation in the WEpod domain but large for the occupancy segmentation, boosting performance. Creating the same colour space for all domains by converting the RobotCar and KITTI domain to greyscale resulted in a tremendous decrease in performance for the drivable path segmentation in these domains. It showed to have a negligible effect on the WEpod domain. The last factor in the image appearance category is a modification of the labels, bringing the number of classes from three to two. This resulted in the best improvement for the drivable path segmentation of the WEpod domain.

Equalising the size of the input dataset of RobotCar and KITTI lead to a smaller training set and resulted in a decrease of performance on the drivable path segmentation of the WEpod. This adjustment is part of the setup category. Another change in this category, leading to the largest improvement for the drivable path segmentation of the WEpod domain, is the order of training. The performance gain implies that the RobotCar domain and WEpod domain resemble more than the KITTI domain and the WEpod domain.

5.1.4. Top view transformation

What is the importance of depth information in the transformation from image space to top view space?

In this work scene understanding in the form of semantic segmentation is deployed in 2D image space while the control of the vehicle occurs in top view. Therefore, to utilise the scene understanding, the performed semantic segmentation is transformed from image space to top view space. A variety of setups with different inputs are possible. Three possibilities are examined: transformation without depth information, transformation with (sparse) ground truth depth information, transformation with (dense) estimated depth information.

From the experiments it is concluded that the influence of depth information information differs per category but in all cases is an added value for the transformation to top view trajectories. This conclusion is based on the evaluation metric which takes the differences between ground truth and the estimation into account. The evaluation metric which compares the overlap in longitudinal direction between ground truth and estimation, only shows added value of depth information for the Residential category. The other two categories, road and city, show longest overlap for the transformation where no depth information is used.

5.1.5. Evaluation

How to evaluate drivable path proposals?

The evaluation of drivable path proposals is important in order to assess the method and to determine whether the results are satisfactory. In this work, the drivable path estimation is evaluated at two different levels: image space and top view space.

For the image space, a variety of default evaluation metrics exist and are used within this work. Different evaluation metrics contain different valuable information about the segmentation. The Jaccard index and F_1-measure are the metrics that are considered in image space.

In top view space, a set of *customised* evaluation metrics are created. From the segmentation in top view, a trajectory line is extracted. This trajectory line is compared to the ground truth trajectory which is extracted via exactly the same method (i.e. from the label transferred to top view). A meaningful metric is created by obtaining the difference between the estimated trajectory line and ground truth trajectory line. This difference is scaled to relate the size of the error with the longitudinal distance from the vehicle. This scale assumes a linear decrease of significance, where distant errors are less important. In order to obtain an overview of each method these values are averaged over the length and number of test images, resulting in one evaluation value. It is important to know to which extent there is an overlap between the estimation and the ground truth in the longitudinal direction. This metric is generated as a certain percentage of the ground truth length, assuming the ground truth is the ideal and correct length.

5.1.6. Final conclusion

Is it possible to estimate and utilise drivable paths for the WEpod domain, solely based on data from different domains?

From the sub question about successful domain adaptation, more insight is gained in the domain shift. A variety of factors showed positive or negative influence on the evaluation metrics. However, it is concluded that the domain shift cannot be addressed to one factor solely. Hence, it is possible to estimate drivable paths for the WEpod domain solely based on external training data, only to a very limited extent. The results based on external training data are not satisfactory and do no reach similar results as when the network is trained on the domain itself.

However, the sub question about top view transformation showed that depth information is of additional value for the transformation to top view which is necessary to utilise the estimated drivable path. Experiments showed that in two out of three cases, ground truth depth information was optimal. However, the main research question focuses on data of different domains (i.e. no data from the WEpod is available). Therefore, it is assumed that the ground truth data of the lidar is not available. Based on external data, which do contain depth information, it is possible to estimate depth information based on a single image and hence, it is possible to utilise drivable path segmentations although not being the optimal solution.

5.2. Recommendations

To use this research as a base for future research or use this research as advice for other work some issues should be taken into consideration. In addition some aspects of the research can be further developed or improved and these are mentioned based on the following sections.

5.2.1. Weakly-supervised segmentation

Ground scatter removal is an important procedure during the generation of ground truth labels. The conducted approach is explained in section 3.1.2. This approach works very well under a variety of situations but has its limitations. Therefore, the current method can be replaced by a more sophisticated approach such as MLESAC [64]. The use of such an approach will result in small improvements, mainly correcting ground scatter at significant distances from the vehicle. Except the improvement in approach, the algorithm can be developed further in order to decrease the computation time and thus be less.

The most common case of semantic segmentation is deployed on images. However, recent research also consists of semantic segmentation of 3D point clouds in order to segment road [9] but also full semantic segmentation containing nine classes [62].

Due to the the phenomenon "urban canyoning", the accuracy of GNSS data and hence the amount usable GNSS data in urban areas is greatly reduced. This automatically leads to less accurate drivable path labels as ground truth. In order to avoid inaccurate ground truth labels without necessity of manual correction, optical flow should be considered as solution to this problem.

5.2.2. Domain shift factors

During the experiments for determining the influence of factors on the domain shift, there is specifically chosen for isolating each factor to see what the influence is of each factor separately. However, when aiming for the next step in improving the performance, a combination of factors should be analysed.

During this work, the influence of a domain shift factor is based on a comparison of evaluation metrics which are the result of semantic segmentation in image space and top view space. However, to get more insight in the learned feature space it would be interesting to see the influence of the modifications on the intermediate layers of the network.

It would be interesting to examine the domain shift on more than three classes which is acquired by full segmentation. This will potentially lead to problems with specific classes and and is thus possible to focus and point down the main reason of the domain shift based on the differences in the environment of the domain.

Since the evaluation metrics point out that there is more resemblance between RobotCar and WEpod than between KITTI and WEpod, it would be interesting to further investigate the specific differences between RobotCar and WEpod.

5.2.3. Top view transformation

In certain cases, the estimated trajectory has only a small overlap with the ground truth. For these cases the scale will be zero at the end of the overlap which will be relatively close to the vehicle. This situation is obviously not a good representation of the real world situations. Instead of creating a scale dependent on the overlap of the ground truth and estimation, it is advised to create a scale dependent on the longitudinal distance from the vehicle, regardless of the existence of an estimate in that area.

Bibliography

[1] Lifeng An, Xinyu Zhang, Hongbo Gao, and Yuchao Liu. Semantic segmentation–aided visual odometry for urban autonomous driving. *International Journal of Advanced Robotic Systems*, 14(5): 1729881417735667, 2017.

[2] M Angulakshmi and GG Lakshmi Priya. Automated brain tumour segmentation techniques—a review. *International Journal of Imaging Systems and Technology*, 27(1):66–77, 2017.

[3] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[5] Dan Barnes, Will Maddern, and Ingmar Posner. Find Your Own Way: Weakly-Supervised Segmentation of Path Proposals for Urban Autonomy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, June 2017. URL `https://arxiv.org/abs/1610.01238`.

[6] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, January 2006.

[7] RPA Bormans, RC Lindenbergh, and F Karimi Nejadasl. Influence of domain shift factors on deep segmentation of the drivable path of an autonomous vehicle. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(2), 2018.

[8] Luca Caltagirone, Mauro Bellone, Lennart Svensson, and Mattias Wahde. Lidar-based driving path generation using fully convolutional neural networks. *arXiv preprint arXiv:1703.08987*, 2017.

[9] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. Fast lidar-based road detection using fully convolutional neural networks. *arXiv preprint arXiv:1703.03613*, 2017.

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[12] Yi-Hsin Chen, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun. No more discrimination: Cross city adaptation of road scene segmenters. In *2017 IEEE Int. Conference on Computer Vision (ICCV)*, pages 2011–2020. IEEE, 2017.

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

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[15] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. *IEEE Transactions on Intelligent Vehicles*, 3(2):129–140, 2018.

[16] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[17] Walter Fischer. *Digital video and audio broadcasting technology: a practical engineering guide*. Springer Science & Business Media, 2008.

[18] Jannik Fritsch, Tobias Kuhnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 1693–1700. IEEE, 2013.

[19] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[20] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, volume 2, pages II–II. IEEE, 2004.

[23] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

[24] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.

[25] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proc. of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[27] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017.

[28] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1175–1183. IEEE, 2017.

[29] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[30] Vijay John, Kiyosumi Kidono, Chunzhao Guo, Hossein Tehrani, Seiichi Mita, and Kasuhiza Ishimaru. Fast road scene segmentation using deep learning and scene-based models. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 3763–3768. IEEE, 2016.

[31] Vicky Kalogeiton, Vittorio Ferrari, and Cordelia Schmid. Analysing domain shift factors between videos and images for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(11): 2327–2334, 2016.

[32] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 2017.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[34] Ankit Laddha, Mehmet Kemal Kocamaz, Luis E Navarro-Serment, and Martial Hebert. Map-supervised road detection. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 118–123. IEEE, 2016.

[35] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[36] Xiaohui Li, Zhenping Sun, Dongpu Cao, Zhen He, and Qi Zhu. Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications. *IEEE/ASME Transactions on Mechatronics*, 21(2):740–753, 2016.

[37] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[38] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.

[39] Christian Lundquist, Thomas B Schön, and Umut Orguner. Estimation of the free space in front of a moving vehicle. Technical Report LiTH-ISY-R-2892, Department of Automatic Control, Linköping University, 2009.

[40] Yecheng Lyu, Lin Bai, and Xinming Huang. Real-time road segmentation using lidar data processing on an fpga. In *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*, pages 1–5. IEEE, 2018.

[41] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[42] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International journal of computer vision*, 43(1):7–27, 2001.

[43] Brian McKenzie and Melanie Rapino. Commuting in the united states: 2009, american community survey reports, acs-15. *Washington, DC*, 2011.

[44] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. *arXiv preprint arXiv:1802.05591*, 2018.

[45] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1 (1):33–55, 2016.

[46] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[47] Scott Drew Pendleton, Hans Andersen, Xinxin Du, Xiaotong Shen, Malika Meghjani, You Hong Eng, Daniela Rus, and Marcelo H Ang. Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1):6, 2017.

[48] Eleni Petridou and Maria Moustaki. Human factors in the causation of road traffic crashes. *European journal of epidemiology*, 16(9):819–826, 2000.

[49] Md Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *International Symposium on Visual Computing*, pages 234–244. Springer, 2016.

[50] Akshay Rangesh and Mohan M Trivedi. No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras & lidars. *arXiv preprint arXiv:1802.08755*, 2018.

[51] Stephen J Redding and Matthew A Turner. Transportation costs and the spatial organization of economic activity. In *Handbook of regional and urban economics*, volume 5, pages 1339–1398. Elsevier, 2015.

[52] J Schönberger, M Pollefeys, A Geiger, and T Sattler. Semantic visual localization. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.

[53] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Object class segmentation using random forests. In *BMVC*, pages 1–10, 2008.

[54] Linda Shapiro and George C Stockman. Computer vision. 2001. *Ed: Prentice Hall*, 2001.

[55] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[56] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[57] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297–1304. IEEE, 2011.

[58] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[59] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[60] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, National Highway Traffic Safety Administration, 2015.

[61] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[62] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *3D Vision (3DV), 2017 International Conference on*, pages 537–547. IEEE, 2017.

[63] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. *arXiv preprint arXiv:1612.07695*, 2016.

[64] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156, 2000.

[65] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Computer Vision (ICCV), 2015 IEEE Int. Conference on*, pages 4068–4076. IEEE, 2015.

[66] Xiang-Yang Wang, Ting Wang, and Juan Bu. Color image segmentation using pixel wise support vector machine classification. *Pattern Recognition*, 44(4):777–787, 2011.

[67] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016.

[68] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 4:6, 2000.

[69] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[70] Rick Zhang, Kevin Spieser, Emilio Frazzoli, and Marco Pavone. Models, algorithms, and evaluation for autonomous mobility-on-demand systems. In *American Control Conference (ACC), 2015*, pages 2573–2587. IEEE, 2015.

[71] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Fully convolutional adaptation networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6810–6818, 2018.

[72] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[73] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2017.