

FACULTY OF ENGINEERING OF THE UNIVERSITY OF PORTO



Drawing the Line: Multimodal Lane Estimation for Autonomous Vehicles

Cláudia Maria Eira Ribeiro

Master's in Electrical and Computer Engineering

Supervisor @ FEUP: Jaime dos Santos Cardoso

Second Supervisor @ FEUP: João Nuno Duarte Fernandes

July 20, 2023

Abstract

Lane boundary estimation plays a critical role in autonomous driving and can be performed with the help of various sensors, including cameras and LiDAR. Deep learning approaches have shown promising results for this task and multimodal fusion between camera and LiDAR information can improve the accuracy of this estimation. Although camera-based methods are more commonly employed, they present several limitations in poorly illuminated scenarios and unfavourable weather conditions, such as rainy days. The fusion of information provided by this sensor with LiDAR data can improve the estimation given the complementarity of characteristics from both domains.

This work addresses the underdevelopment of LiDAR-based and fusion methodologies for road and lane estimation, primarily caused by the lack of sufficient training and testing data. This task is performed using semantic segmentation and the SalsaNext architecture is the chosen state-of-the-art model due to its good performance in previous works. Three fusion strategies are proposed, trained, and evaluated using the KITTI-ROAD dataset and a proprietary Bosch dataset. The early fusion network aims to fuse the raw information from the two modalities at an early stage; late fusion combines the results of two separate predictions (one for each of the modalities) and takes place at the decision level; middle fusion merges the camera and LiDAR data at the context module level. All fusion strategies require calibration between the sensors. However, a novel middle fusion strategy is presented, eliminating the need for calibration parameters and introducing a unique approach to feature alignment during training. Finally, an in-depth analysis is performed, and the middle fusion network emerges as the top-performing model. The results highlight the potential of LiDAR and RGB fusion methodologies and pave the way for future advancements in road and lane estimation for autonomous driving.

Resumo

A estimação das faixas da estrada é uma tarefa crucial em condução autónoma e pode ser realizada com a ajuda de sensores como câmaras e LiDAR. Algoritmos de *machine learning* têm vindo a apresentar resultados promissores nesta tarefa, sendo que a fusão da informação destes sensores pode melhorar a precisão desta estimação. Atualmente, métodos que utilizam apenas dados provenientes de câmaras são os mais comuns, no entanto apresentam algumas limitações especialmente em cenários com baixa luminosidade e condições atmosféricas não favoráveis, como é o caso da chuva. Assim, a fusão de informação deste sensor com LiDAR pode ajudar a ultrapassar estes problemas dada a complementaridade de características de ambos os domínios.

Este trabalho aborda o subdesenvolvimento de metodologias baseadas em informação proveniente de LiDAR, e da sua fusão com câmara para a estimativa da estrada e das suas faixas. Este facto é principalmente causado pela falta de dados suficientes para treinar e testar as redes. Esta tarefa é realizada através de segmentação semântica e a arquitetura SalsaNext é o modelo de estado-da-arte escolhido devido ao seu bom desempenho em trabalhos anteriores. Em seguida, três estratégias de fusão são propostas, treinadas e avaliadas usando o *dataset* KITTI-ROAD e um *dataset* proprietário da Bosch. A *early fusion* tem como objetivo combinar as informações brutas das duas modalidades numa fase inicial; a *late fusion* combina os resultados de duas previsões separadas (uma para cada modalidade) e ocorre ao nível da decisão; a *middle fusion* funde os dados da câmara e do LiDAR ao nível do *context module*. Todas as estratégias de fusão requerem calibração entre os sensores. No entanto, é apresentada uma nova estratégia de *middle fusion*, que elimina a necessidade de parâmetros de calibração e introduz uma abordagem original para o alinhamento de características durante o treino. Finalmente, é realizada uma análise detalhada, e a *middle fusion* destaca-se como a estratégia com melhor performance. Os resultados obtidos destacam o potencial das metodologias de fusão de LiDAR e RGB e abrem caminho para avanços futuros na estimação de estradas e faixas de rodagem para a condução autónoma.

Acknowledgements

I would like to express my deepest gratitude to my mentors at FEUP, Professor Jaime Cardoso and Dr. João Fernandes. Their guidance, expertise, and unwavering support played a pivotal role in shaping my dissertation.

I would also like to extend my heartfelt appreciation to my mentor at Bosch, Dr. João Pinto. It was an honor to be mentored by João throughout this entire dissertation. His support, availability, and wealth of knowledge have been instrumental in the successful completion of this research. João's guidance and willingness to share his expertise have not only enhanced my technical skills but also broadened my perspective on the industry. I am incredibly grateful for the opportunity to learn from him and to be mentored by such an exceptional professional.

Additionally, I want to express my sincere thanks to the entire team at Bosch for their support and collaboration. Working alongside such talented individuals has been an honor, and I am truly grateful for the enriching experience.

A special thanks for my friends, who have been there for me every step of the way. Your support, encouragement, and understanding have made this journey even more meaningful.

I would also like to thank my family for their unwavering encouragement and understanding throughout this demanding endeavor. Your belief in my abilities has been a constant source of motivation, and I am incredibly lucky to have such a strong support system.

Finally, I would also like to express my deepest gratitude to my boyfriend, Armando. He has been an incredible source of support and strength throughout every moment of this journey. From offering words of encouragement to providing a listening ear, his unwavering belief in my abilities has been crucial in keeping me motivated and focused.

Cláudia Ribeiro

This thesis was partially supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 202) [Project nº 047264; Funding Reference: POCI-01-0247-FEDER-047264].

*“You could do anything, if you only dared. And deep down, you know it, too. That’s what scares
you most.”*

Sarah J. Maas

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Objectives	3
1.4	Document Structure	3
2	Background	5
2.1	Sensors in Autonomous Driving	5
2.2	Deep Learning	5
2.2.1	Artificial Neural Networks	6
2.2.2	Convolutional Neural Networks	9
2.2.3	Underfitting and Overfitting	10
2.2.4	Regularization	10
2.3	Evaluation Metrics	11
3	3D Point Cloud Processing	15
3.1	Introduction	15
3.2	3D Point Clouds	15
3.2.1	Properties	15
3.2.2	Data Representation	16
3.3	Datasets	17
3.3.1	Overview of the datasets	18
3.4	Sensor Fusion	19
3.5	Fusion Methodologies	20
3.5.1	Early Fusion	20
3.5.2	Late Fusion	21
3.5.3	Middle Fusion	21
3.5.4	What to fuse	21
4	Literature Review	23
4.1	Image Classification Methods	24
4.2	Object Detection-based Methods	24
4.3	Segmentation-based Methods	24
4.4	Anchor-based Methods	25
4.5	Parameter-based Methods	26
4.6	Data Augmentation	26
4.7	State-of-the-art Approaches	27
4.7.1	Camera-based Methods	27

4.7.2	LiDAR-based Methods	29
4.7.3	Fusion Methods	30
4.8	Road Estimation	31
4.9	Final Remarks	33
5	Methodology	35
5.1	SalsaNext Architecture	35
5.1.1	Contextual Module	36
5.1.2	Dilated Convolution	37
5.1.3	Pixel-Shuffle Layer	37
5.1.4	Central Encoder-Decoder Dropout	37
5.1.5	Average Pooling	37
5.2	Framework Implementation	38
5.2.1	Early Fusion	38
5.2.2	Late Fusion	38
5.2.3	Middle Fusion	39
5.3	Objective Functions	39
5.4	Final Remarks	40
6	Experimental Setup	41
6.1	Datasets	41
6.1.1	KITTI-ROAD Dataset	41
6.1.2	Bosch Dataset	42
6.2	Data Preprocessing	43
6.2.1	KITTI-ROAD	43
6.2.2	Bosch Dataset	45
6.3	Data Augmentations	45
6.4	Road and Lane Estimation	45
6.4.1	Unimodal	45
6.4.2	Multimodal	45
6.5	Training Setup	46
7	Results and Discussion	49
7.1	Road Estimation	49
7.1.1	Urban Marked (UM_ROAD)	49
7.1.2	Urban Multiple Marked (UMM_ROAD)	50
7.1.3	Urban Unmarked (UU_ROAD)	52
7.1.4	URBAN_ROAD	54
7.1.5	Bosch Road	56
7.2	Lane Estimation	56
7.2.1	Urban Marked (UM_LANE)	56
7.2.2	Bosch Lane	59
7.3	Qualitative Analysis	61
7.3.1	Urban Marked Road (UM_ROAD)	61
7.3.2	Urban Multiple Marked Road (UMM_ROAD)	61
7.3.3	Urban Unmarked Road (UU_ROAD)	64
7.3.4	Bosch Road	65
7.3.5	Urban Marked Ego-lane (UM_LANE)	67
7.3.6	Bosch Lane	71

7.4	Final Discussion	71
8	Conclusion and Future Work	77
8.1	Future Work	78
	References	81

List of Figures

1.1	The six levels of automation in the SAE J3016 standard. Extracted from [3]. . . .	2
2.1	Deep Neural Network with one input layer, two hidden layers and one output layer. Adapted from [12].	6
2.2	Example of a convolutional layer, extracted from [21].	10
2.3	Example of a pooling layer extracted from [22].	10
2.4	Confusion matrix for a binary classification task.	11
2.5	Illustration of the metric Intersection Over Union (IoU).	14
3.1	Illustration of the different fusion methods. Adapted from [2].	20
4.1	Previous lane estimation methodologies.	23
4.2	Illustration of the PolarMix [52] data augmentation techniques.	27
4.3	Overview of the CLRNet architecture, extracted from [53].	28
4.4	Overview of the baseline network introduced in K-Lane, extracted from [6]. . . .	30
4.5	Diagram of the FusionLane network, extracted from [33].	31
4.6	CFECA (Cross Fusion Efficient Channel Attention) module and fusion strategy extracted from [32].	32
5.1	SalsaNext architecture (extracted from [44]).	36
5.2	Proposed early fusion framework.	38
5.3	Proposed late fusion framework.	39
5.4	Proposed middle fusion framework.	39
6.1	Depth maps resulting from the upsampling operation.	44
7.1	Results for the UM_ROAD category in the um_000036 sample.	62
7.2	Results for the UM_ROAD category in the um_000074 sample.	63
7.3	Results for the UMM_ROAD category in the umm_000088 sample.	64
7.4	LiDAR and RGB data used for the middle fusion in range view. Example for the sample image uu_000029.	65
7.5	Results for the UU_ROAD category in the uu_000029 sample.	66
7.6	Results for road segmentation in the Bosch dataset for an example test sample. . .	67
7.7	Results for road segmentation in the Bosch dataset for an example test sample. . .	68
7.8	Results for road segmentation in the Bosch dataset for an example test sample. . .	68
7.9	Results for road segmentation in the Bosch dataset for an example test sample. . .	69
7.10	Example of samples with considerable cross-talk effects.	69
7.11	Results for the UM_LANE category in the um_000005 sample.	70

7.12 Results for lane markings segmentation in the Bosch dataset for an example test sample.	72
7.13 Results for lane markings segmentation in the Bosch dataset for an example test sample.	72
7.14 Results for lane markings segmentation in the Bosch dataset for an example test sample.	73
7.15 Results for lane markings segmentation in the Bosch dataset for an example test sample.	73
7.16 Results for lane markings segmentation in the Bosch dataset for an example test sample.	74
7.17 Results for lane markings segmentation in the Bosch dataset for an example test sample.	74

List of Tables

3.1	Comparison of lane datasets. Based on the table from [6].	18
6.1	Dataset statistics of the KITTI-ROAD dataset. Extracted from [65].	42
6.2	Split of KITTI-ROAD dataset used for road estimation.	42
6.3	Split of KITTI-ROAD dataset used for ego-lane estimation.	42
6.4	Split of the Bosch dataset used.	43
7.1	Road estimation results for the UM_ROAD category in KITTI_ROAD, with data augmentation.	51
7.2	Comparison between our best performing model in the UM_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark.	52
7.3	Road estimation results for the UMM_ROAD category in KITTI_ROAD, with data augmentation.	53
7.4	Comparison between our best performing model in the UMM_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark.	53
7.5	Road estimation results for the UU_ROAD category in KITTI_ROAD, with data augmentation.	55
7.6	Comparison between our best performing model in the UU_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark.	56
7.7	Road estimation results for the URBAN_ROAD category in KITTI_ROAD, with data augmentation.	57
7.8	Comparison between our best performing model in the URBAN_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark.	58
7.9	Road estimation results in the Bosch dataset, without data augmentation.	58
7.10	Comparison between our best performing model in the UM_LANE category with the state-of-the-art models from the KITTI-ROAD benchmark.	59
7.11	Road estimation results for the UM_LANE category in KITTI_ROAD, without data augmentation.	60
7.12	Lane line estimation results in the Bosch dataset, without data augmentation. . .	61

Abbreviations and Symbols

2D	Two-Dimension
3D	Three-Dimension
AdaGrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
ADAS	Advanced Driver Assistance Systems
AF	Activation Function
ANN	Artificial Neural Network
AP	Average Precision
ASPP	Atrous Spatial Pyramid Pooling
BEV	Bird's Eye View
CFECA	Cross Fusion Efficient Channel Attention
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
ECA	Efficient Channel Attention
FP	False Positive
FPR	False Positive Rate
FN	False Negative
FNR	False Negative Rate
FOV	Field of View
HAF	Horizontal Affinity Field
HDA	Hierarchical Deep Aggregation
IDA	Iterative Deep Aggregation
IoU	Intersection over Union
LBEV	LiDAR Bird's Eye View
LiDAR	Light Detection And Ranging
LSTM	Long Short-term Memory
mAP	Mean Average Precision
MLP	Multilayer Perceptron
MSE	Mean Square Error
NN	Neural Network
PR-Curve	Precision-Recall Curve
PRE	Precision
RADAR	Radio Detection and Ranging
RAM	Random-Access Memory
REC	Recall
ReLU	Rectified Linear Unit
RMSProp	Root Mean Squared Propagation
ROI	Region of Interest
SAE	Society of Automotive Engineers
SGD	Stochastic Gradient Descent
Tanh	Hyperbolic Tangent Function

TN	True Negative
TP	True Positive
UM	Urban Marked
UMM	Urban Multiple Marked
UU	Urban Unmarked
VAF	Vertical Affinity Field
VLDNet	Vision-based Lane Region Detection Network
YOLO	You Only Look Once

Chapter 1

Introduction

Through the past decades, the autonomous driving field has undergone significant progress. The need for safer roads has triggered the development of intelligent transportation in order to achieve fully automated perceptive systems. Research has shown that human errors cause the majority of road accidents, with the leading cause being associated with the timing faults of the drivers [1]. Autonomous driving systems aim to replace parts of the human driving activity, ensuring that these errors are avoided and providing a more care-free experience.

The vision navigation system of an intelligent vehicle has to be able to understand its surrounding environment and the estimation of the position of lanes is a crucial task when referring to autonomous driving and Advanced Driver Assistance Systems (ADAS). In more recent years, Deep Learning technologies have gone through major improvements, presenting promising results for this task. Driverless vehicles need to be built upon three major requirements: having a perception system capable of recognizing the driving environment with high accuracy; good performance under various situations and not favourable weather conditions; being able to evaluate the scene in real-time since many times vehicles are driven at high speeds [2].

This research was undertaken at Bosch Car Multimedia S.A. as a part of the THEIA project, a collaborative initiative with the University of Porto aimed at improving the sensory capabilities of autonomous vehicles.

1.1 Context

The amount of human monitoring and supervision needed and the operation design domain that a vehicle can operate in, determine the level of automation for that vehicle. The current standard adopted by the Society of Automotive Engineers (SAE) defines six different levels of automation [3], as shown in Fig. 1.1. In level 0, the driver is the only responsible for all the driving tasks. In levels 1-2 the human drivers still have environmental monitoring responsibilities notwithstanding the assistance provided by the autonomous driving technologies. In levels 3-4 the driver may have to assist the driving activity, but the vehicle is capable of performing several driving functions

by itself. The final level of automation, level 5, is the ultimate goal in autonomous driving, where the driver and the system do not need to communicate in any way.

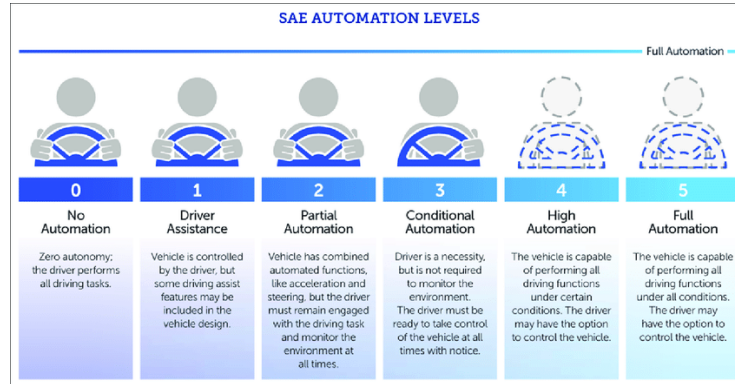


Figure 1.1: The six levels of automation in the SAE J3016 standard. Extracted from [3].

Lane estimation aims to determine the accurate location and trajectory of the ego lane and neighbouring lanes. It is the key to smooth traffic flow and occupant safety and, therefore, an increasingly studied topic within the development of intelligent transportation. This is a critical feature in the vision navigation system of intelligent vehicles and, with the advance of artificial intelligence technologies, the autonomous driving field has taken extra steps towards environment perception in driving scenarios by adopting deep learning approaches [4]. Lane lines, such as solid white, broken white, and dotted white lines, provide important information to drivers regarding their expected behaviour and interactions with other vehicles. In this regard, lane estimation can also help to alert the drivers of changes in the topology of the road [5]. Furthermore, lane estimation systems are able to ensure a robust trajectory prediction making them a key task in the development of autonomous vehicle systems.

The estimation of the position of lanes allows the resolution of the drivable area providing essential information for the perception system. This is a complex task that requires a precise detection and the identification of, not only lane lines, but also the merging, splitting and ending of lanes [5]. There are a lot of hurdles when it comes to lane estimation since a real environment can present very distinct and unexpected situations [4]. Some of the most common problems that affect this task are the invisible lane painting due to ageing, the covered lane marks due to shadows or other cars on the road and illumination changes.

1.2 Motivation

Generally, lane estimation is performed using computer vision solutions on RGB data [6]. The contrast between the dark asphalt and the bright road markings in the visual information captured by cameras mounted on the vehicle is used to recognize lane regions [7]. Cameras are a quite popular option due to their low cost and high-resolution. Most of the datasets available nowadays rely on camera images [8] since they provide rich textured information and allow collecting objective

amounts of data. However, relying solely on visual information is not a wise approach, as several environmental factors could blur the boundaries between roads and markings and lead to potentially deadly errors [9]. Cameras are very sensitive to lighting changes making them not suitable for nighttime and poorly illuminated scenarios. As LiDAR (Light Detection And Ranging) sensors become more affordable and offer unprecedented resolution, using data collected by these sensors is becoming a promising option for multimodal lane estimation. They can avoid the influence of lighting conditions and provide accurate depth information. However, their resolution is worse compared to cameras. Through the 3D point cloud, the information collected with LiDAR can be used to construct a compilation of data points in three dimensions like in the real world.

1.3 Objectives

The main goal of this dissertation is to study lane and road estimation approaches in the context of autonomous driving. As such, three main stages can be described regarding the scope of this work. The first step consists of a more theoretical study of the sensors used for lane estimation followed by the concept of point clouds, their representation, and structure. This analysis serves as a foundation for the description of current state-of-the-art single-modal and multimodal architectures. Furthermore, the most relevant datasets for lane estimation are mentioned, as well as the ongoing questions regarding the fusion between RGB images and LiDAR point clouds. In the second stage, an in-depth comparison between state-of-the-art methodologies is performed. Camera-based, LiDAR-based and fusion-based architectures are thoroughly analysed and their advantages, disadvantages and performances are collected and compared. Lastly, after the establishment of the preferred characteristics from state-of-the-art architectures, the last stage aims to propose a multimodal lane and road estimation approach based on deep learning. The method will fuse RGB and LiDAR information and its evaluation will be done using a benchmark dataset. Moreover, the results of the experiments conducted will be analysed, and some conclusions will be drawn.

1.4 Document Structure

This document is divided into eight chapters. Chapter 1 introduces a general overview of the context of this dissertation and the current motivation for the writing of this document. Chapter 2 presents the main sensing modalities that will be used and establishes important concepts for the understanding of deep neural networks. Then, Chapter 3 specifies the processing of 3D point clouds as well as a comparison between datasets with lane annotations. Moreover, the main questions regarding sensor fusion are depicted. Chapter 4 divides current lane estimation algorithms in different categories, and an in depth comparison between state-of-the-art models is conducted. Also, road estimation is introduced as an additional task to be performed in the scope of this dissertation. Chapter 5 presents the methodology adopted for the experiments carried out and Chapter 6 introduces the experimental setup utilized. Chapter 7 covers the results obtained, and a thorough

analysis is performed. Finally, in Chapter 8 the main conclusions are drawn and the directions for future work are proposed.

Chapter 2

Background

This chapter will present the sensing modalities of interest used in autonomous driving. Afterwards, a general overview about Deep Learning and artificial neural networks will be given. To finalize, some of the most common evaluation metrics for deep learning architectures will be described.

2.1 Sensors in Autonomous Driving

The need for more robust and reliable perception systems in autonomous driving has triggered the search for different sensors other than cameras. The use of LiDARs has been a topic of study, and it has shown promising results. Other modalities like RADAR (Radio Detection and Ranging) and Ultrasonic sensors have also been used for deep multimodal perception in autonomous driving. Since the scope of this dissertation focuses on camera and LiDAR data, only these two sensing modalities will be explored.

Camera images provide detailed visual texture information of the surrounding environment, however they are not capable of presenting depth information that can be useful for lane estimation. Images captured by visual cameras are sensitive to weather and lighting conditions [2].

LiDAR (Light Detection And Ranging) can be defined as a system that sends pulses of laser and measures the time for the reflected light to return to the receiver. The direction and distance of each pulse are recorded and allow the composition of a sparse three-dimensional world [2]. LiDAR sensors are able to measure the reflective intensity of each received light pulse making it possible to collect additional data. This information is then stored in the format of a point cloud.

2.2 Deep Learning

Deep Learning is a type of Machine Learning architecture that relies on Artificial Neural Networks composed of multiple layers. They are capable of extracting high level features from a raw input and have been very successful in solving computer vision tasks, as described in [10]. Many methods use deep networks for scene understanding in autonomous driving and all of them

have achieved better results when compared to traditional methods. However, Deep Learning algorithms need to be trained using large amounts of data in order to achieve a good performance.

2.2.1 Artificial Neural Networks

Neural networks (NNs) can be described as a set of nodes or units connected by direct links. Every link is used to propagate the output of an activation function from one node to another and has a numeric weight associated that determines the strength and sign of the connection [11]. The base unit of a neural network is called a neuron, and it is responsible for computing the weighted sum of the inputs, adding a bias value and applying the activation function. Typically, deep neural networks consist of an input layer, several hidden layers and an output layer as presented in Fig. 2.1. The purpose of the input layer is to process raw data transforming it into an appropriate representation that can be propagated to the following layers. The hidden layers are capable of learning complex patterns in data by adjusting connections between the nodes. The number of hidden layers is variable, as well as the number of neurons that compose each one. Finally, the output layer provides the result from the accomplished task of the network.

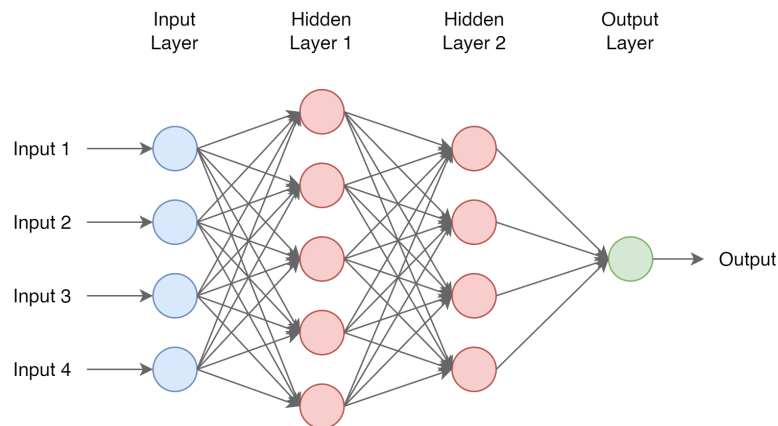


Figure 2.1: Deep Neural Network with one input layer, two hidden layers and one output layer. The distinction of these three types of layers is done through a colour scheme: blue neurons compose the input layer, red neurons represent the hidden layers, green neurons constitute the output. Adapted from [12].

2.2.1.1 Activation Functions

Activation functions (AFs) in Deep Learning are responsible for the computation of the weighted sum of inputs and biases. This result decides if a neuron should be fired or not. Through a form of gradient processing such as gradient descent, AFs produce an output for the NN containing parameters as the data [13]. These functions have evolved through the years, and can be categorized in linear and non-linear.

- **Sigmoid Function**

The Sigmoid AF or logistic function is a non-linear activation function mostly employed in feedforward neural networks. It is used for predicting probability outputs and has been widely utilized in binary classification problems and logistic regression tasks. It provides an output between 0 and 1 and is given by equation (2.1).

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.1)$$

- **Softmax Function**

The Softmax function computes the probability distribution from a vector of real numbers outputting a range of values from 0 to 1. The sum of the probabilities must be equal to 1. This activation function is mostly used to address multi-class models returning the probability of each class. The predicted task will have the highest value of probability, and this function can be described by equation (2.2).

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (2.2)$$

- **Hyperbolic Tangent Function (Tanh)**

The Tanh function produces a zero centred output between -1 and 1 allowing the support for the back-propagation process [13]. When compared to the sigmoid function, it reaches better training performance for multi-layer neural networks [14]. Even so, Tanh can only attain a gradient of 1 when the input is 0 producing some dead neurons that can limit the functioning of the network. Equation (2.3) provides the computation of this function.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.3)$$

- **Rectified Linear Unit (ReLU) Function**

The Rectified Linear Unit (ReLU) Function was proposed by [15], and it is nowadays, one of the most used activation functions in Deep Learning applications. It performs a threshold operation to every input element given by equation (2.4).

$$f(x) = \max(0, x) = \begin{cases} x_i & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (2.4)$$

This function eliminates the vanishing gradient problem by rectifying the values of the inputs that are less than zero to zero. Its computation is relatively fast since it does not include operations such as divisions and exponentials, improving the overall computation speed of the network. In general, it presents a better performance when compared to the previous functions. However, the ReLU function is more easily overfitted than the sigmoid function.

2.2.1.2 Loss Functions

The Back-propagation algorithm was developed to solve the problem of unknown error values at the hidden layers of a neural network. In 1986, Rumelhart *et al.* [16] introduced this concept as a way for hidden layers to adapt their weights based on the back-propagation of the errors in the output layers. It consists of a gradient computation of the loss function. Consequently, the choice of an adequate cost function becomes a crucial task when fitting a neural network. The most common cost functions are:

- **Mean Square Error (MSE)**

This metric is widely applied in regression problems. The calculation of the Mean Square Error is done through the average square difference between true values (Y_i) and predicted values ($\hat{Y}_i$). The resulting value is called the prediction error. Therefore, the MSE of a model can be calculated by equation (2.5), where n is the total number of observations.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (2.5)$$

- **Cross-Entropy**

The Cross-Entropy loss function also called logarithmic loss or logistic loss is mostly used in classification problems. The probability of each predicted class is compared to the desired class output 0 or 1. The loss is calculated through the penalization of the probability based on the distance from the actual value. Since the penalty is logarithmic, larger scores correspond to large differences (closer to 1), and smaller scores are associated with small differences (closer to 0). The formula for the Cross-Entropy loss of a model consists in a sum of all the classes probabilities, as expressed in (2.6), where C corresponds to the total number of classes, and n is the total number of observations.

$$CE = -\frac{1}{n} \sum_{i=1}^n \sum_{c=1}^C (Y_i \log(\hat{Y}_i)) \quad (2.6)$$

2.2.1.3 Optimizers

In a deep learning model, parameters need to be adjusted so that the loss function can be minimized. Optimizers can be used to perform this adjustment and also to find the optimal value for the weights and biases of the network [17]. These optimal values can be determined by iteratively evaluating the loss function of the model on a training dataset and using the calculated gradients to update the model's parameters. This technique is often called gradient descent. An important hyperparameter in an optimizer is the learning rate, which determines the speed at which the model's parameters are updated. A large learning rate implies that the parameters will suffer more significant updates, which can speed up the training process, but can make the model prone to overshooting the optimal values. Some of the most common optimization algorithms used in

Deep Learning are the Stochastic Gradient Descent (SGD) [18], the Adaptive Gradient Algorithm (AdaGrad) [19], the Root Mean Squared Propagation (RMSProp) [18] and the Adaptive Moment Estimation (Adam) [20]. Nonetheless, only the SGD and Adam optimizers will be described more in depth.

- **Stochastic Gradient Descent (SGD)**

The Stochastic Gradient Descent (SGD) [18] is a variant of gradient descent widely used in Machine Learning problems. It approximates the true gradient of the loss function with respect to the model's parameters by using only a small, randomly selected subset of the training data, called a mini-batch, to calculate the gradients at each step. Since SGD utilizes mini-batches, it achieves faster training times as well as constant computational time. This algorithm converges regardless of the number of training samples, however it has a slow asymptotic convergence.

- **Adaptive Moment Estimation (Adam)**

The Adam optimizer [20] is an extension of the SGD algorithm. Through the employment of moving averages of the gradient and the square gradient, it is able to adjust the step size of the update. As such, Adam can automatically adjust the learning rate for each parameter, which makes it more efficient and effective than SGD. This algorithm combines the advantages of AdaGrad and RMSProp eliminating the lack of bias correction present in RMSProp with momentum. Other techniques explored in this algorithm such as gradient clipping and weight decay further improve the stability and performance of the model during training.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks or CNNs [21] are a type of neural network with at least one layer that uses a convolution operation rather than a standard matrix multiplication. CNNs possess three types of layers: convolutional layers, pooling layers and fully-connected layers.

The convolutional layer will calculate the scalar product between the weights of the input volume-connected region and the neurons whose output is related to particular regions of the input. The weights also make up a square matrix known as the kernel, and the so-called filters are made up of a collection of kernels. A visual representation of a convolutional layer is present in Fig. 2.2. As an output of the convolutional processes, each convolutional layer produces a feature map by applying a filter to the input data. To further reduce the number of parameters in that activation, the pooling layer will then simply downsample along the spatial dimensionality of the input. Fig. 2.3 demonstrates an example of a pooling operation. The fully-connected layers will next carry out the identical tasks as in conventional ANNs and make an effort to derive class scores from the activations, which can subsequently be used for classification. Additionally, it is proposed that ReLU be applied in between these layers to enhance performance.

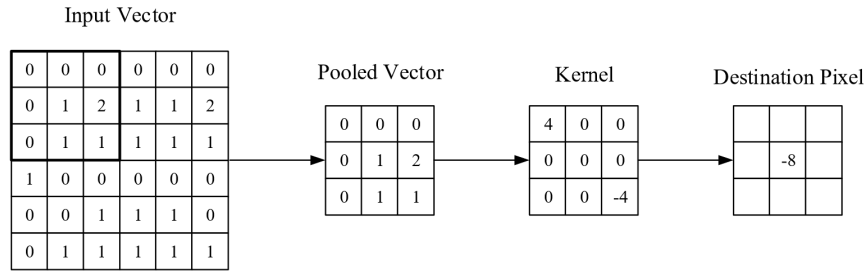


Figure 2.2: Example of a convolutional layer, extracted from [21]. The input vector is placed over the kernel's central element, which is then computed and replaced with a weighted sum of any surrounding pixels and itself.

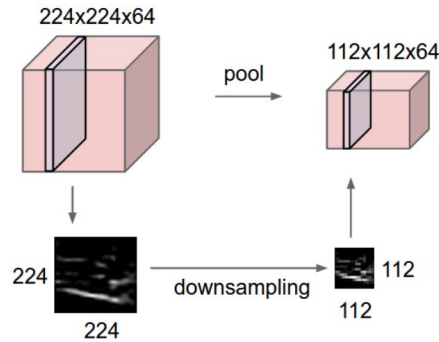


Figure 2.3: Example of a pooling layer extracted from [22]. With filter size 2, stride 2, the input volume of size $224 \times 224 \times 64$ is pooled into the output volume of size $112 \times 112 \times 64$.

2.2.3 Underfitting and Overfitting

Besides the first objective to develop a model that performs well on the training data, the second aim of a good Deep Learning algorithm is to possess the capability of generalizing to new inputs. Many times, the performance of a model on yet-unseen data can be much worse in relation to the training. This phenomenon is called overfitting. It occurs when the neural network is modelling the training data too well by learning overly specific patterns in those specific inputs, that are linked to data noise instead of label information. Consequently, the testing error will be much higher than the training error. On the other hand, underfitting occurs when the model is not complex enough to learn the underlying patterns in the data. Therefore, it is not able to make accurate precisions. In order to solve these problems, a more adequate architecture can be used, the amount of data can be increased, or regularization techniques can be employed.

2.2.4 Regularization

A fundamental problem in Deep Learning is the development of an algorithm that will perform well both on the training data and on new inputs. Several strategies have been used to reduce

the test error and are usually called regularization techniques [23]. Their aim is also to prevent overfitting by adding a penalty term to the loss function of the model, which encourages it to have small weights and to be less complex. This helps the model to generalize better to unseen data and to avoid overfitting. This method is used in L1 (Lasso Regression) and L2 (Ridge Regression) regularization techniques. L1 regularization penalizes the total of the weights' absolute values, whereas L2 regularization penalizes the total of the weights' squares.

Other types of regularization include Early Stopping and Dropout. The former consists of stopping the training process before the network fully converges. The performance of the network on the validation dataset is monitored, and the training process is ceased when it starts to degrade. Dropout regularization works by randomly dropping out, or setting to zero, a number of output features of the layer during training.

Data Augmentation can also be seen as a regularization method. It increases the size of the training set by applying random transformation to existing data. In this way, the amount of training data increases as well as its diversity improving the robustness of the network.

Finally, Batch Normalization performs a normalization of the output of each layer, so that they have a mean of 0 and a standard deviation of 1. This accelerates the training and improves the ability of the network to generalize new data.

2.3 Evaluation Metrics

When designing a deep learning network, two different tasks can be addressed: regression or classification. The regression task corresponds to the prediction of a continuous value, rather than a discrete label. The input is mapped with a continuous output variable and the algorithms can be divided into linear and non-linear regression.

The classification task refers to the mapping of a given input with a discrete output variable. The input domain is divided into different classes based on a decision boundary defined by the neural network. Classification algorithms can be binary or multi-class and some of the most important evaluation metrics are [24]:

- **Confusion matrix**

If we consider the lane estimation task as a binary classification problem, the resulting confusion matrix will be similar to Fig. 2.4.

		True Class	
		Positive	Negative
Predicted Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Figure 2.4: Confusion matrix for a binary classification task.

The number of correct predictions is given by the diagonal of the confusion matrix and corresponds to the values of True Positives (positive samples correctly classified as positive) and True Negatives (negative samples correctly classified as negative). For multi-class problems, there must be one row and one column for each class.

- **Accuracy**

The Accuracy is typically used to evaluate classification tasks, however, it can also be utilized to assess the performance of a model performing a regression task. This indicator contrasts the number of samples that were successfully identified with the overall sample count. Its equation computes the sum of the number of true positives with the value of true negatives, divided by the sum of true positives, true negatives, false positives and false negatives, as in equation (2.7).

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.7)$$

- **Precision (PRE)**

Precision relates to the ability of a model to only identify relevant objects. It is calculated through the division of the number of true positives by all detections, as in equation (2.8), where TP corresponds to the number of true positive samples and FP is the number of false positives.

$$Precision = \frac{TP}{TP + FP} \quad (2.8)$$

- **Recall (REC)**

Recall is a performance metric that presents the ability to find all relevant cases. In other words, it is the percentage of positive predictions given all the ground truths. It is given by the division between the number of true positives and the sum of the true positives and false negatives. Recall can be expressed as in equation (2.9), where FN corresponds to the number of false negative samples.

$$Recall = \frac{TP}{TP + FN} \quad (2.9)$$

- **Precision-Recall Curve (PR-Curve)**

The PR-Curve, which is a plot of the precision (y-axis) and the recall (x-axis) for various confidence levels, allows the identification of only relevant ground-truth objects. In terms of the graphical representation, typically a high area under the curve tends to indicate both high precision and recall. However, it is very sensitive to small variations in the ranking of examples resulting in a zigzag-like curve. To overcome this problem, [25] performed an interpolation that, for each recall level, R , selects the maximum precision measured, as presented in equation (2.10).

$$P_{interp}(R) = \max P(\tilde{R}) \quad (2.10)$$

- **False Positive Rate (FPR)**

The False Positive Rate, in a classification task, measures the proportion of false positives versus all positive predictions (the sum of false positives and true positives), as given by equation (2.11).

$$FPR = \frac{FP}{FP + TP} \quad (2.11)$$

- **False Negative Rate (FNR)**

The likelihood that a true positive may be missed by the test is known as the False Negative Rate, given by equation (2.12).

$$FNR = \frac{FN}{FN + TP} \quad (2.12)$$

- **Average Precision (AP)**

The Average Precision is obtained through the area under the PR-Curve. It is useful to assess the performance of a model without using a particular decision threshold. The AP is obtained by considering the maximum precision $P_{interp}(N)$ whose recall is greater than N , as expressed in equation (2.13).

$$AP = \frac{1}{N} \sum_{R=1}^N (P_{interp}(N)) \quad (2.13)$$

- **Mean Average Precision (mAP)**

The Mean Average Precision calculates the mean value of the Average Precision for each class, as given in equation (2.14), where N is the number of existing classes and AP_i consists of the AP in the i th class.

$$mAP = \frac{1}{C} \sum_{i=1}^C (AP_i) \quad (2.14)$$

- **Intersection-over-Union (IoU)**

This metric is based on the Jaccard Index and measures the overlapping area between the ground-truth, B_{GT} and the predicted bounding box, B_P , divided by the area of union between them. IoU is mostly used in applications of object detection and is presented in Fig. 2.5.

- **F1 Score (MaxF)**

The F_1 score or F-measure is generally used in binary classification systems. It is the harmonic mean of the proportions of true positive predictions among all positive predictions made by the model and true positive predictions among all cases of true positive predictions in the dataset. Higher F1 scores, which range from 0 to 1, indicate superior performance. Since it gives both precision and recall equal weight, it is a useful metric to employ when trying to strike a balance between them. Equation (2.15) denotes this metric.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2.15)$$

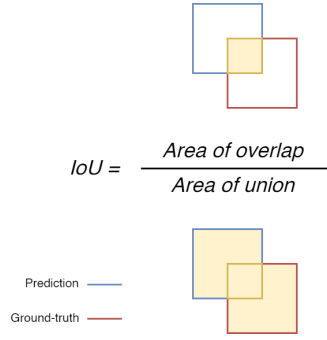


Figure 2.5: Illustration of the metric Intersection Over Union (IoU). The blue box refers to the predicted bounding box; the red box denotes the ground-truth bounding box.

A variation of this metric called the F_β -measure allows prioritizing precision or recall based on how critical they are for the case or depending on the importance we want to give each one. This metric is affected by the distribution of classes and is given by equation (2.16).

$$F_\beta = (1 + \beta^2) * \frac{\text{Precision} * \text{Recall}}{\beta^2 * \text{Precision} + \text{Recall}} \quad (2.16)$$

During this dissertation, the F_1 score will be referred to as “MaxF” to be consistent with the name of this evaluation metric in the KITTI-ROAD dataset.

Chapter 3

3D Point Cloud Processing

3.1 Introduction

3D point cloud processing can be defined as a process to analyse and modify 3D point clouds. The goal is to achieve the best quality, the lowest storage and the fastest transmission of information [26]. This chapter will present the baseline knowledge regarding 3D point clouds, their representation, and their properties. Afterwards, the available datasets with lane annotations will be described and compared. Finally, the topic of sensor fusion will be addressed as well as several underlying questions such as how to fuse the data and what to fuse.

3.2 3D Point Clouds

The 3D point cloud is the most common storage form for LiDAR information. It presents a complete 3D image with the data collected from the sensor by rendering it as pixels. Point clouds can be defined as a set of data points in 3D space, where each point can be individually represented. In the context of this dissertation, point clouds will refer to the data captured by LiDAR sensors.

3.2.1 Properties

LiDAR point clouds provide depth and reflectance information about the environment [2]. The depth information of a point cloud can be defined as a set of Cartesian coordinates (x, y, z), whereas the reflectance information is given by intensity. However, point cloud representations present several downsides that affect the identification of points, making lane detection a difficult task. The following three fundamental properties of LiDAR come from its data acquisition method.

- **Occlusion:** The 3D point cloud will automatically suffer from occlusion when the LiDAR sensor acquires data from a single viewpoint. The reflected light beams collected by the sensor belong to the surface of the surrounding environment facing it. Therefore, objects that are located behind the surrounding environment will not be captured by the LiDAR

since they are visually blocked. These objects can be partially or completely invisible to the LiDAR causing the occlusion phenomenon.

- **Sparseness:** The point density produced by LiDAR scanning changes according to the distance of objects. Given the regular spacing of LiDAR lasers, the sensor will collect fewer 3D points as the distance from objects increases.
- **Lower resolution:** When compared to the data of typical 2D sensors such as cameras, point clouds present lower resolution. This sparsity between points can lead to difficulties in the recognition of smaller objects in the driving scenario.

3.2.2 Data Representation

Point clouds can be described as a set of unordered 3D points where each point is formed by four values. The fourth number pertains to the light pulse intensity, r , whereas the first three values provide the Cartesian or spherical coordinates with respect to the LiDAR coordinate frame. This definition can be described as a set of N points: $S = \{(p_i, a_i)\}_{i=0}^N$, where p_i represents the 3D coordinate of the i th point, and a_i refers to other attributes of the same point. In the scope of this dissertation, we will consider $a_i = r_i$.

- **Raw points**

The most simple form of representing 3D point clouds is to create a matrix where each 3D point in the set S is represented by a row. This representation includes no pre-processing steps and salvages all the original information in the points. However, geometric properties of 3D points are not explored.

- **3D Voxelization**

In 3D voxelization the 3D space is discretized into voxels that can be combined to represent a single point. This representation can be implemented through the partition of the 3D space into equally-spaced and non-overlapping 3D voxels. Voxelization in three dimensions allows the creation of a hierarchical structure where voxels have a uniform spatial size. It enables the usage of standard techniques such as 3D convolutions. However, specific properties of point clouds are not taken into account and the representation can become immensely sparse leading to many empty voxels. Moreover, there is a compromise between resolution and memory that can limit the performance of methods using this representation.

- **Range-view**

A real-time LiDAR scan consists of a series of range measurements gathered from a specific location with a certain angular field of view. Range-view organizes 3D points into a 2D range-view image, where the x -axis relates to the azimuth angle and the y -axis dictates the elevation angle of the laser beams. Each pixel in the 2D range-view image corresponds to a frustum in the 3D space and contains the range from the LiDAR to the closest 3D point in the frustum.

- **Bird's eye view**

The bird's eye view (BEV) can be interpreted as a special case of 3D voxelization, where the height dimension is ignored. The 3D voxels are projected to a BEV image and a 2D top-view image is created where each pixel represents a pillar space. This is a common representation technique in previous lane estimation methods due to the easy application of 2D vision-based techniques.

3.3 Datasets

The training of a deep neural network requires an enormous amount of data, and it must include diverse driving conditions to promote the network's accuracy and robustness. Another determining factor for good datasets is the quality of the data that substantially influences the performance of deep multimodal perception systems. Several benchmark datasets with lane lines annotations are available, and in this section they will be analysed and compared.

The KITTI Road [27] dataset contains pixel-level annotations for lane lines and three categories of road scenes: unmarked, marked and multiple marked urban roads. However, only a few hundred lane markings are annotated which is not enough to train deep neural networks. It is also a very unbalanced dataset which makes it not suitable for training the network for real-life applications.

TuSimple [28] was released in 2017 and contains 6408 images annotated with lane markings. Although this dataset contains a considerable amount of images, there are no annotations for occluded lane markings which are of utmost importance for perception systems in autonomous driving.

The data in CULane [29] was collected over several days in Beijing and by different drivers. This city provides many challenging driving scenarios improving the diversity of the collected data. This dataset contains more than 55 hours of videos and 133,235 frames. The lane lines were annotated manually using cubic splines and occluded lanes were also annotated based on the context. However, only four lane markings are annotated in each scenario.

The CurveLanes [30] dataset was released with the intent of addressing more challenging curve lanes. It contains 150K lanes images with 650K annotated lane labels. This dataset contains images with difficult scenarios such as curves and multi-lanes, and a higher resolution when compared to previous benchmark datasets such as TuSimple [28].

The BDD100K [8] is one of the largest camera-based driving video datasets. It includes scenes with diverse weather conditions as well as different geographic and environmental surroundings. The lane markings are labelled with eight categories: road curb, crosswalk, double white, double yellow, double other colour, single white, single yellow, single other colour. The attributes of continuity (full or dashed) and direction (parallel or perpendicular) are also labelled. The distribution of types of lane markings is similar for parallel/vertical and full/dashed lanes. When it comes to the lane category, single white lanes are the most common.

The Unsupervised LLAMAS dataset [31] proposes an automatic annotation of lane markers for the collected data. The lanes in the 100,042 images were labelled through highly-accurate maps and an optimization step to improve accuracy. This method allows the annotation of thousands of LiDAR and camera images with limited human effort. However, the data in LLAMAS was only collected on highways limiting the diversity of the training scenarios. As such, the performance of methods tested in this dataset does not provide accurate results for real-life driving environments.

DeepLane [9] is a private dataset containing 55,168 frames of both camera and LiDAR data from urban and highway environments. In both the camera view and the overhead view, the annotations show the actual lane borders. A localization system and HD maps with dense ground height ground truth are provided via dense 3D measurements and annotated lane graphs. The fact that this dataset is not public means that not many derivative works have been conducted.

In comparison to camera based methods, the use of LiDAR point clouds is less studied in part because of the lack of public datasets and benchmarks for LiDAR lane estimation networks. KAIST-Lane or K-Lane [6] is a recent dataset that was developed to fill in the gap of LiDAR based information for lane detection. It is the largest open LiDAR lane dataset (15382 frames) that allows the training of lane detection algorithms in both urban roads and highways. In K-Lane, the front camera images were calibrated with LiDAR point clouds in order to lead the way for further detection studies using multimodal sensors. The sets provided include various road conditions and demanding scenarios such as curved lanes, sharp curves, crowded traffic situations with lane occlusions and merging lanes (converging and diverging). The LiDAR point clouds were projected into BEV in order to produce the ground truth labels. The candidates for lane lines were obtained through the thresholding of the intensity measure and each lane is represented by a one-pixel-wide line.

3.3.1 Overview of the datasets

Name	Year	Sensors	Num. Lanes	Size (frames)	Public	Road Type
KITTI Road [27]	2013	Camera, LiDAR	n.a	579	Yes	Urban
TuSimple [28]	2017	Camera	5	6408	Yes	Highway
CULane [29]	2018	Camera	4	133235	Yes	Urban, Highway
CurveLanes [30]	2020	Camera	6	150K	Yes	Urban, Highway
BDD100K [8]	2020	Camera	n.a	100K	Yes	Urban, Highway, Other
LLAMAS [31]	2019	Camera, LiDAR	n.a	100.042	Yes	Highway
DeepLane [9]	2019	Camera, LiDAR	n.a	55168	No	Urban, Highway
K-Lane [6]	2022	Camera, LiDAR	6	15382	Yes	Urban, Highway

Table 3.1: Comparison of lane datasets. Based on the table from [6].

Although not every dataset with lane markings is denoted in this section, it is possible to observe that the number of solid datasets containing only camera information is higher than datasets containing both modalities, as presented in Table 3.1. This is one of the factors that contribute to the underdevelopment of lane estimation architectures that use LiDAR information. With the release of the new multimodal dataset KAIST-Lane [6], it is hoped that more studies on LiDAR lane detection will be conducted in the future. Since camera images have also been calibrated with LiDAR point clouds, this dataset can be used for studies using sensor fusion.

3.4 Sensor Fusion

The use of LiDAR-camera fusion information can be motivated by several reasons: camera images contain abundant information but do not work well under weak and different lighting conditions due to limited photosensitivity, whereas LiDARs can avoid the influence of light; LiDAR-only methods can easily identify the input data associated with each cell. However, point clouds present difficulties at large distances due to sparsity and limit the resolution in perception, something that does not affect high-resolution images; although cameras can provide high resolution information, they cannot directly provide depth information in opposition to 3D points from LiDAR; many times it is necessary to project front camera images into 2D bird's eye view for motion planning, which often causes lane distortions. On the other hand, LiDAR point clouds do not suffer from distortion in the BEV-projection.

For these reasons, a combination of the two types of data could benefit the lane estimation task. This multimodal fusion consists of an aggregation of data from different sensors and the feature fusion can be done at different levels. However, the question of when and how to fuse is still a topic of discussion [2]. The goal is to take the data from different types of sensors and try to find the optimal combination process to obtain the most accurate perception of the surroundings. It has been proven that working with multimodal information can improve the performance of neural networks and enhance feature extraction. The advantage of using data from multiple sources is their complementarity, and since the information is completely uncorrelated, the predictions will be more robust. The development of a deep multimodal perception system is particularly difficult for autonomous driving due to the requirements for real-time performance, robustness, and accuracy. It is essential that these systems provide reliable decision-making to ensure the safety of drivers in complex driving situations. There are three main methods for data fusion illustrated in Fig. 3.1: early fusion, which consists of combining raw multimodal data and then feeding the result to a network; late fusion, which merges the output of two different networks for each type of data; middle fusion, that presents a compromise between the last two methods.

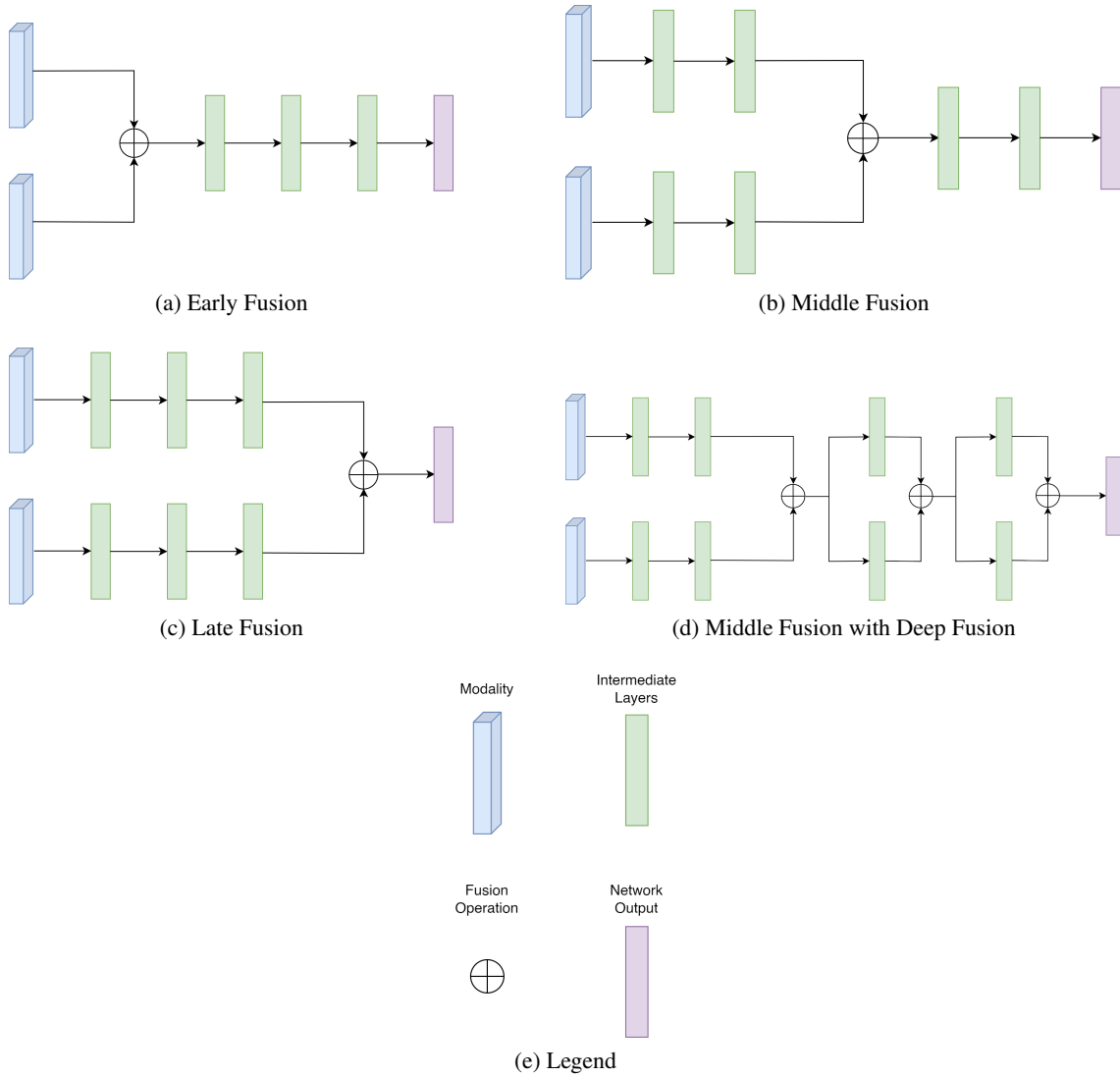


Figure 3.1: Illustration of the different fusion methods. Adapted from [2].

3.5 Fusion Methodologies

3.5.1 Early Fusion

The combination of raw data from different sensors is done during the preprocessing phase. The result is given as an input to a neural network and involves low calculation and low storage requirements. Since the network learns the combination of features from both modalities at an early stage, the information in the raw data can be fully exploited. However, this method is susceptible to spatio-temporal data misalignment caused by imbalances in the sensors. Moreover, in early fusion there is the need for data alignment and the replacement of a sensing modality implies a complete retraining of the network.

3.5.2 Late Fusion

In late fusion, the output of several pipelines is gathered to achieve the final result. For each data type a different network is trained. As a consequence, if a new sensor modality needs to be included or replaced there will be no interference with the other networks. This method is very flexible and modular, but has a high computational cost and storage needs.

3.5.3 Middle Fusion

This fusion is done at the feature processing stage, and it has a wider option of fusion locations. On the other hand, it is not easy to determine the best method for this type of fusion in a specific network architecture. It enables the network to learn cross-modalities with different feature representations and at different depths. Zhang *et al.* [32] also implemented a middle fusion operation in order to compare the results with the early fusion version. Two stages of middle fusion were tested, and it was concluded that the inference time increased as the fusion stage moved later. Deep fusion is a type of middle fusion where the different modalities are fused several times across the network. FusionLane [33] implemented a fusion operation every time there was a compression of the feature map. In this way, the network could learn the best fusion strategy.

3.5.4 What to fuse

Another question in sensor fusion relates to what data should be fused. There are several ways to process and represent LiDAR point clouds and camera images separately, however, their characteristics are different and an optimal fusion of features must be achieved.

Camera images provide high-resolution and rich textured information about the driving scenarios. However, most of the time objects are occluded and the scale of an object can vary in the camera image plane. Since the bird's eye view (BEV) is commonly used for LiDAR point clouds, the adaptation of camera images to the BEV plane can achieve promising results [2].

LiDAR point clouds can be discretized of the 3D space into 3D voxels followed by the correspondence between the points and the voxels. Due to the sparsity and irregularity of LiDAR points, this method often results in several empty voxels. One way to reduce the impact of the data loss caused by these empty voxels is the implementation of a k-NN search point cloud interpolation performed in [32]. For each blank pixel it calculates the weighted average of the normalized distance of the three closest points. Another way to process point clouds and to avoid voxelization is to directly learn over 3D LiDAR points in continuous space vectors. The last method to process LiDAR information consists of the projection of 3D point clouds onto a 2D grid-based feature map in order to process them via 2D convolutional layers.

Chapter 4

Literature Review

In this chapter, existing lane estimation algorithms will be divided into different categories and the foundations of each one will be analysed. Subsequently, a few state-of-the-art architectures will be a target of a more in depth description. Finally, some conclusions will be drawn and the baseline for this dissertation will be set.

Before diving into what is lane estimation and the different architectures developed for this task it is important to underline the difference between the terms detection and estimation. Both are associated with lane in previous works, but their meaning is somewhat different. The terms detection and estimation are closely related but work with different types of inputs. For example, detecting a lane line in a road is equivalent to determining that a lane exists and where it is located. After this detection we might want to estimate its course for situations where it is occluded or has faded. In this way, estimation deals with finding the best parameters from a set of measurements. Since the scope of this dissertation comprises both methods, the terms' lane detection and lane estimation will be used interchangeably in this document.

The research on lane estimation has come a long way, but there are still some limitations that overburden the development of a robust identification system such as lighting conditions and partial occlusion. However, with multi-layer nonlinear transforms, deep learning has achieved impressive results in lane detection systems. Many deep learning algorithms have been tested in order to tackle the lane detection challenge and CNN-based methods have presented valuable results. The different lane estimation methods can be divided into several categories as in Fig. 4.1.

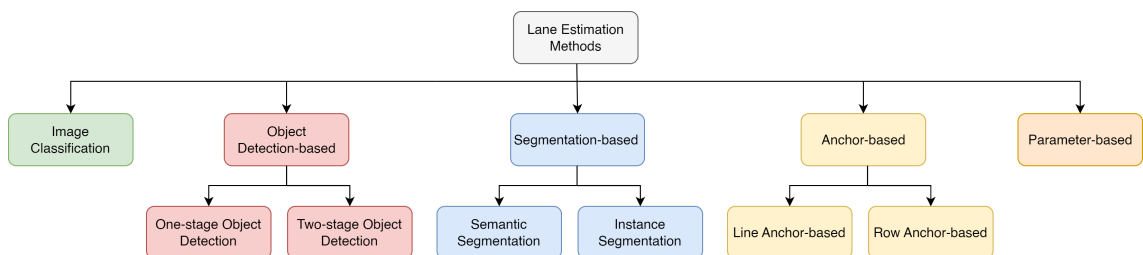


Figure 4.1: Previous lane estimation methodologies.

4.1 Image Classification Methods

Image classification aims to distinguish objects present in the input but does not provide the location of lanes. In order to achieve the detection of lanes it is necessary to perform a mapping between some prior knowledge of the location and the CNN mapping function. This idea was developed in DeepLane [9], but due to the limiting aspect of requiring prior knowledge of the locations, this strategy does not work well with lane detection.

4.2 Object Detection-based Methods

Lane line detection can be seen as an object detection task whose goal is to locate and recognize multiple objects in a scene. Deep learning approaches for object detection can be divided into two categories: one-stage and two-stage detections.

In one-stage detections, the classification is done through a mapping of the feature maps to bounding boxes. Architectures like YOLO [34] and G-CNN [35] are able to generate the coordinate value relative to the position of the category without performing first a region proposal. One stage detectors are faster and more easily optimizable but lack in accuracy.

In two-stage detections, candidates are first generated by CNN or traditional methods, such as Hough transforms and various types of filters. The regions of interest are identified, extracted, verified and classified by assigning them to a category. In most cases, two-stage detections achieve better results due to the refinement paradigm and the region of interest (ROI) generation. However, they require more complex training as well as higher inference time. Fast R-CNN [36] is one of the most common two-stage object detection algorithms and achieves a better performance in terms of time cost when compared to the state-of-the-art network R-CNN [37]. The Faster R-CNN architecture [38] introduces a new region proposal method based on RPN, improving its predecessor Fast R-CNN.

4.3 Segmentation-based Methods

From the perspective of line shape description, segmentation-based methods can be used as a solution for the lane detection task. They can be divided into two approaches: semantic segmentation and instance segmentation.

Semantic segmentation associates each pixel of an image to a given class label. In autonomous vehicles some important classes to identify are, for example, traffic signs, other vehicles, pedestrian and road features such as lanes. This type of approach allows a more complete understanding of the surrounding environment. However, it has been shown to be time-consuming and computationally intensive.

Spatial CNN [39] proposes a message passing mechanism between pixels across rows and columns in every layer. This approach improves the detection of long continuous shapes with less apparent visual clues, which is the case of lane lines. However, it has been proven to be slow

for real-time applications. In CurveLane-NAS [30] the curve lane detection problem is addressed using an architecture that can capture long and short-range curve information. VLDNet [40] incorporates an encoder-decoder network using semantic segmentation. ResNet [41] and UNet [42] are combined in order to achieve more reliable lane features, outperforming several previous methods. This architecture is divided into two stages: ResNet-50 performs down sampling of the image and identifies the necessary features, whereas UNet decodes them and recognizes the attributes defining the lane candidate features.

Although semantic segmentation was first advanced to process camera images, more recent methods have been proposed for segmenting LiDAR points as well. Yecheng Lyu *et al.* introduces a road segmentation architecture using LiDAR data on an FPGA [43]. This architecture is based on a CNN model with no pooling layers, making it possible for all the internal results to be stored in the same memory space directory without the need for allocation or reshaping. In SalsaNext [44] an improvement in both accuracy and reliability in real-time segmentation is achieved. This architecture is built upon the SalsaNet model [45], and it is based on an encoder-decoder architecture. SalsaNext introduces a new context module replacing the ResNet encoder blocks in SalsaNet. It presents epistemic and aleatoric uncertainty estimations providing reliable predictions which are crucial for safety-critical identification systems.

Finally, several methods fuse camera and LiDAR information to perform semantic segmentation. This is the case of FusionLane [33] that utilizes the results of the semantic segmentation to construct a high-precision map of the driving scenarios. Unlike other methods, LiDAR point clouds are converted into bird's eye view instead of camera images. FusionLane addresses the lane detection task as a multi-classification task and successfully fused LiDAR and camera data through a dual-input branch structure.

Instance-level segmentation considers each lane as a separate instance. The goal is to achieve the identification of objects of the same class and divide them into different instances. A common problem with this approach is the need to define in advance the number of instances which constrains the classification to a predefined and fixed number of lanes. Most methods present this downside, however, LaneNet [46] proposes an architecture able to cope with a variable number of lanes and lane changes. It casts each lane as its own instance that can be trained end-to-end. LaneNet applies a learned perspective transformation that, in contrast to bird's eye view, is not fixed and predefined. This method is able to cluster each lane line during training rather than leaving this task to post-processing, increasing real time efficiency.

4.4 Anchor-based Methods

Anchor-based methods can be divided into two classes: line and row anchor-based methods.

In line anchor-based methods, lines are used as references to regress lanes. LaneATT [47] integrates a novel anchor-based attention mechanism capable of aggregating global information. An input image is taken, and a boundary line is generated through a CNN. This network creates

a feature map that is pooled to extract the anchor's features. Finally, those features are combined with a set of global features outputted by an attention module.

In row anchor-based methods the prediction of the probable lane cell is done for each predefined row on the image. CondLaneNet [48] introduces a conditional lane detection strategy based on conditional convolution and row-wise information. Its architecture solves two problems: the fact that segmentation-based shape description is inefficient for lane lines due to their high level of freedom; the inaccuracy of instance detection for slender and curved lines. CondLaneNet is based on the prediction of the starting points of lane lines. However, they can be difficult to identify in some cases resulting in inferior performance of this method. Furthermore, Zequn Qin *et al.* [49] developed a novel formulation with row-base selection that significantly reduces the computational cost. Their method also tackles the no-visual-clue problem by utilizing information from other locations such as other lanes and the road shape.

4.5 Parameter-based Methods

The detection of lane lines is done through a regression of parameters in opposition to point regression performed in other methods. PolyLaneNet [50] proposes a polynomial regression approach that achieved competitive results when compared to existing state-of-the-art methods. On the other hand, Ruijin Liu *et al.* [51] presents an end-to-end architecture capable of directly outputting the parameters of a lane shape model without the need to perform post-processing.

4.6 Data Augmentation

Data Augmentation is a technique designed to increase the size and the diversity of a dataset overcoming the overfitting problem by improving the generalization capability of the network. This is achieved through different types of modifications to the original data such as geometric transformations, colour modifications, cropping and rotations.

Most of the data augmentation techniques have been developed to work with 2D images and have presented great potential for the training of robust DNN models with limited training images. However, very few of these methods work well on LiDAR point clouds. To get beyond physical restrictions on the laser beam's propagation, point cloud data has to be enhanced. Considering data sparsity of point clouds, they are predicted to get sparser the farther away from the LiDAR scan, therefore data augmentation techniques must get around this constraint to accurately represent the real world. The most popular data augmentation techniques for point clouds are frustum dropout, additive Gaussian noise application, flipping in the x-z and y-z planes of the LiDAR coordinate frame, and rotation along the z axis of the LiDAR coordinate frame. Recently, Aoran *et al.* [52] developed a point cloud augmentation technique that preserves point cloud integrity while also allowing for its enrichment. PolarMix [52] can be used in a variety of network topologies such as object detection and semantic segmentation. It edits and mixes points along the azimuth direction performing two cross-scan augmentation methods: scene-level switching, where two circular

LiDAR scans are sliced along the azimuth axis and their point clouds are swapped as illustrated in 4.2(a); the second method involves taking point cloud instances from one LiDAR scan, rotating them repeatedly along the direction of the scan (to make many copies), and then pasting the rotated instances into another LiDAR scan as in 4.2(b).

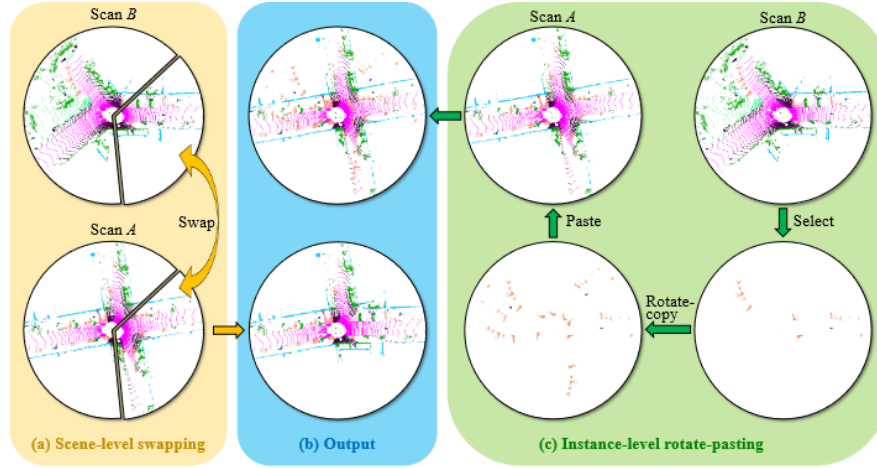


Figure 4.2: Illustration of the PolarMix [52] data augmentation techniques: (a) Demonstration of the scene-level swapping with exchange sector of LiDAR scans A and B cut with given azimuth angles; (c) To create several copies of the cut point instances, the instance-level augmentation rotates point instances around the z-axis by a variety of azimuth angles before pasting them into scan A. (b) Demonstration of the result of augmentation performed in scan A by the two approaches.

4.7 State-of-the-art Approaches

Several approaches for lane detection have been outlined so far, but in this section some of the best performing architectures will be described in more detail. The covered methods can be divided into three categories: camera-based, LiDAR-based and fusion-based.

4.7.1 Camera-based Methods

Regarding camera-based architectures there are two important ones to denote: CLRNet and LaneAF.

- **CLRNet [53]**

This method achieved state-of-the-art accuracy in CULane [29]. It combines low-level and high-level features for the lane detection task due to their complementarity. The first step in this architecture consists of the lane prior. Due to the thinness and length of lanes, a predefined lane prior might help the network to better localize them. To represent a long line, the common rectangular bounding box is replaced for equally-spaced 2D-points. As such, lanes are represented by a sequence of points, *i.e.*, $P = \{(x_1, y_1), \dots, (x_N, y_N)\}$, and

$y_i = \frac{H}{N-1} * i$, where H is the image height. Equal samples of the points' y coordinate are taken vertically throughout the picture. In the same way, every x coordinate associates with the respective y_i . This representation is called the lane prior. The second component of this model is the cross layer refinement that aims to utilize both low-level and high-level features to detect lanes with more detailed features. Shallow low-level features present more contextual information and deep high-level features strongly respond to full objects with richer semantic meanings. The goal of this refinement structure is to develop a feature pyramid with high-level semantics throughout by utilizing a ConvNet's pyramidal feature hierarchy, which contains semantics at various levels from low to high. The backbone is ResNet [41] and L_0, L_1, L_2 refer to the features generated by FPN [54]. From the highest level, L_0 , the cross-layer refinement eventually descends to L_2 , as presented in Fig. 4.3(a). After the assignment of the lane priors to the feature maps, the ROIgather step performs

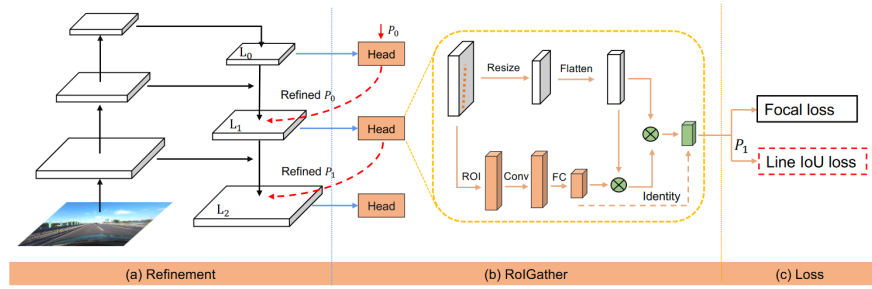


Figure 4.3: Overview of the CLRNNet architecture, extracted from [53]. (a) The feature maps are generated from the FPN [54] structure. Afterwards, lane priors are refined from high-level to low-level features. (b) Further contextual information exploitation by each head. (c) Improvement of the regression of the lane priors through Line IoU loss, as well as classification.

convolutions to collect adjacent features for each lane pixel using the recovered ROI features obtained through ROIALign [37, pp. –]. Finally, the proposed Line IoU loss is based on the Intersection over Union metric and predicts the lane as a whole, improving the performance. Its formulation is given by equation (4.1).

$$L_{LIoU} = 1 - \frac{\sum_{i=1}^N (d_i^o)}{\sum_{i=1}^N (d_i^u)} \quad (4.1)$$

Where, $\frac{d_i^o}{d_i^u}$ corresponds to the calculation of the IoU between the extended line segment and its ground truth.

- **LaneAF [5]**

The LaneAF method is the best-performing architecture in the LLAMAS [31] dataset so far. It is an instance segmentation approach based on a convolutional neural network and

introduces affinity fields as a way to cluster and associate pixels that belong to the same lane entity. Unlike many methods that use ResNet or ENet as the backbone of the network, LaneAF makes use of the DLA-34 [55] backbone. Deep layer aggregation, which combines semantic and spatial fusion for better localization and semantic interpretation, is a technique used by the DLA family of models. They use two types of aggregation: hierarchical deep aggregation (HDA), which concentrates on combining features from all modules and channels, and iterative deep aggregation (IDA), which concentrates on fusing resolutions and scales. This architecture is also trained to forecast both horizontal and vertical affinity fields in addition to binary lane segmentation masks. Both horizontal and vertical affinity field (HAF, VAF) are vector fields capable of assigning a unit vector to each pair of (x,y) locations. The ground truth vector field mapping is used to compute the HAF vectors for each lane point for each row y in the image. The VAF vectors are computed similarly. A decoding process is used to group foreground pixels into lanes during testing after a model has been trained to predict the HAFs and VAFs. The algorithm goes row-by-row from the bottom to the top of the image assigning every foreground pixel to the respective lane.

4.7.2 LiDAR-based Methods

Although the previous camera-based methods are capable of achieving good results, they have some inherent problems. For example, most architectures in the CULane [29] benchmark exhibit a considerable performance decline (of roughly 20%) for nighttime and bright light settings compared to their daytime performance. However, due to the underdevelopment of LiDAR datasets, methods that make use of this type of information are less developed and many utilize proprietary datasets in order to train and validate the networks.

- **K-Lane [6]**

In K-Lane, a baseline network that utilizes global feature correlators is presented. The overall structure of the model is shown in Fig. 4.4. The first step of the network consists of the BEV encoder that converts a 2D pseudo-image created from a 3D point cloud into a 2D BEV feature map. Two types of encoders are introduced: point projector and pillar encoder. Given that the pillar encoder performs slightly worst but has faster inference times, it is suggested to be used for real-time applications. In contrast, the point projector generates a BEV feature map using convolutional neural networks after projecting point clouds into the horizontal plane. The Global Feature Correlator (GFC) utilizes patch-wise self-attention networks to correlate high resolution features. These types of networks are able to identify correlations between distant grids from the very beginning of backbone development preserving high resolution information. This is a valuable capability since lane lines are sparse, thin and stretched along the entirety of the point cloud. In order to detect lanes, the problem is formulated as a multi-class segmentation task. The detection head is capable of distinguishing lane classes and lane shapes and consists of two segmentation heads,

each containing a series of two-layer shared-MLPs (shared Multilayer Perceptrons) with a non-linear activation in the middle.

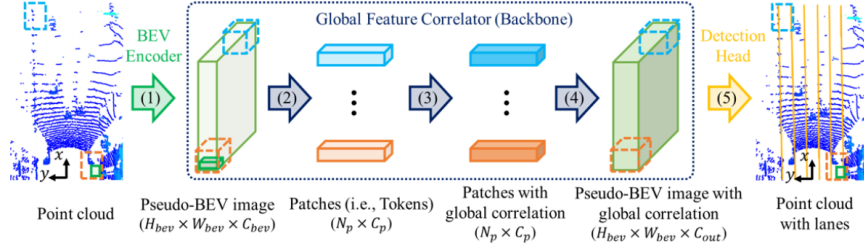


Figure 4.4: Overview of the baseline network introduced in K-Lane, extracted from [6]. (1) BEV encoder; (2) reshape and per-patch linear transform; (3) Transformer or Mixer block; (4) reshape and shared MLP; (5) detection head. H_{bev} (height); W_{bev} (width); C_{bev} (number of channels of the pseudo-BEV image); C_{out} (number of channels of the output pseudo-BEV image); N_p^2 (number of total patches); C_p (number of channel per patch in the global correlation).

4.7.3 Fusion Methods

- **FusionLane [33]**

This method fuses camera and LiDAR information every time there is a compression on the feature map, so it is performing a deep fusion operation. The first step in this architecture is to process both types of data. While camera images are transformed into BEV through an inverse perspective mapping, the LiDAR information undergoes a transformation into a three-channel BEV (LBEV). The value of the first channel corresponds to the intensity of points, the second value relates to the average height information of the points and the third value refers to the standard deviation of the height value of the points and its eight neighbours. The backbone is based on the modified Xception network and can be divided into four parts. The encoder module performs convolution operations in both camera and LiDAR inputs. Since there are several fusion operations, the network can learn the best fusion strategy through the data. The modifications done to the original network are the added batch normalization after every convolutional layer and the replacement of the max pooling by a convolutional layer. After, the Atrous Spatial Pyramid Pooling (ASPP) module is responsible for the capture of multi-scale information. Then, the LSTM (long short-term memory) module has the ability to only remember crucial information. It helps to reduce the negative impact caused by errors in the camera branch. Finally, the decoder module will perform bilinear up-sampling two times restoring the image to its original size. Following a 1×1 convolution operation, the original input image is directly combined into the second upsampling process, and the outcome demonstrates that it significantly enhances the image's ability to recover details. An important aspect to denote is that all low-level features come from the input of the LBEV. The overall structure of FusionLane can be seen in Fig. 4.5.

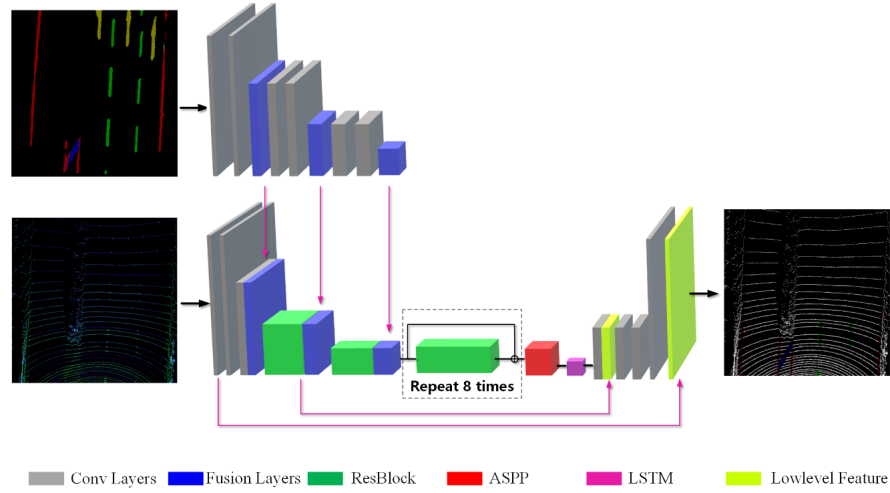


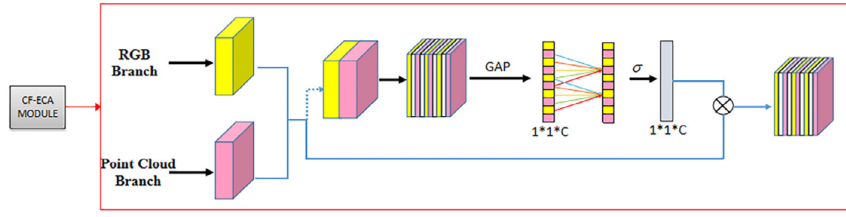
Figure 4.5: Diagram of the FusionLane network, extracted from [33].

- **Channel Attention in LiDAR-camera Fusion for Lane Line Segmentation [32]**

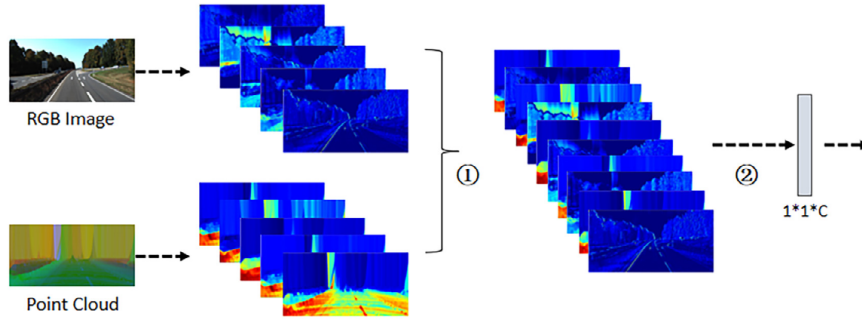
This work introduces a novel multimodal fusion method based on a channel attention mechanism. The baseline model comprises U-Net [42] and ResNet-34 [41] blocks and to lessen the loss of spatial information brought on by the downsampling procedure, each block in the encoder is connected to the equivalent block in the decoder. Regarding the feature fusion between the camera and LiDAR point clouds, the height information is utilized, after the completion of the sparse point cloud, as an extra filtering condition. To create a pseudo-three-channel picture, the height and distance data are overlaid on the reflection coefficient. The point cloud is where the pseudo-three-channel picture gets all of its information. This helps to decrease noise interference. Due to the sparsity of point clouds, a k-NN ($k=2$) search interpolation is applied in order to fill in the empty voxels. The attention mechanism aims to assign different weights to different channels given their importance. This means that they will have different degrees of influence in the resulting feature map outputted by the fusion layer. An ECA module was introduced so that the model complexity is lower and the dimensionality loss is reduced. Moreover, to take advantage of the complementarity of multimodal information, the feature maps of the camera images and the point cloud are cross spliced and then, the attention mechanism is applied. This approach is presented in Fig. 4.6 as well as the fusion strategy adopted.

4.8 Road Estimation

Another task closely related to lane detection is road estimation. Without the recognition of the road area, an autonomous vehicle would not be able to perform reliable decisions in the navigation process. Since lanes are intrinsically related to the road, this will also be a topic of study in this dissertation.



(a) Cross fusion efficient channel attention module



(b) Lidar and camera images cross stitch feature channels during the fusion process

Figure 4.6: CF-ECA (Cross Fusion Efficient Channel Attention) module and fusion strategy extracted from [32]. In (b), the first operation refers to the cross stitching of the feature channels and the second one represents the weight calculation of the same channels after the fusion.

Several datasets with road annotations are available, but the KITTI-ROAD dataset [56] was created specifically for road and ego-lane estimation. Many works have been submitted for benchmarking but one of the top-performing models at the time of the writing of this document is SNE-RoadSeg+ [57]. This network is an updated version of the SNE-RoadSeg [58] algorithm and addresses the difficulty of real-time execution. This architecture is a free-space detection algorithm that performs data fusion between RGB images and depth/disparity images [59]. The SNE+ module is responsible for a more accurate surface normal estimation. The depth/disparity pictures are converted into surface normal maps and merged with the features gathered from the RGB images, allowing precise free-space recognition. The RoadSeg+ block employs a data fusion DCNN (deep convolutional neural network) that follows the common encoder-decoder structure. This is also the first DCNN that adopted the deep supervision paradigm, which consists of supervising intermediate layers to increase network performance.

PLARD [60], on the other hand, utilizes LiDAR information and overcomes the issue of its adaptation to the visual data space. This network implements data space adaptation by aligning the LiDAR points with the perspective view through altitude difference-based transformations. Then, a cascaded fusion structure facilitates feature space adaptation between LiDAR and camera features.

4.9 Final Remarks

When performing an evaluation of CLRNet [53] in the CULane dataset [29], a significant performance drop occurs for nighttime conditions. This fact extends to LaneAF [5] that exhibits a drop of 18.77% between daytime and nighttime situations. In contrast, the network proposed in K-Lane [6] has a minimal performance deterioration between day and night scenarios. This is due to the robustness of LiDAR information in the presence of challenging light conditions. However, this model has shown a decrease in performance when detecting sharp curves. In these situations, camera data might help the identification providing more complete information.

Due to the different datasets used for the three categories of algorithms mentioned before, it is difficult to directly compare methods that use distinct sensors. In order to evaluate the performance of the lane detection results using various sensors, Bai *et al.* [9] conducted several experiments using a proprietary dataset enabling the comparison of the performance of networks trained with several types of data. According to the experiments carried out, it is clear that models that combine both LiDAR and camera information achieve the highest performance. When performing a more in depth analysis of the results, this statement proves to be especially true on highways where vehicles move more quickly. Therefore, even the aggregation of numerous sweeps, does not fully compensate for the far regions that still contain relatively few LiDAR points. In these cases, the much denser camera images can partially make up for this factor. The LiDAR only model presents worse results than the fusion one, however, the camera-only method produces the lowest performance.

When referring to the fusion methods described earlier, the datasets used to train and test each network are different, but both are adapted from the KITTI dataset. From the results obtained in FusionLane [33], it is clear that sufficiently accurate semantic segmentation is not achieved if the network depends just on one type of sensor, whether a camera or a LiDAR. The solution lies in the efficient merging of data from several sensors. In Channel Attention [32], different fusion stages are tested and the best performing one corresponds to the early fusion using the CFECA module. The FP of this attention module is lower when compared to modules like A2Net [61] and CCNet [62]. As such, it is more suitable for lane line segmentation tasks.

It is possible to conclude that lane estimation methods that fuse camera and LiDAR information have a better chance of outperforming current state-of-the-art architectures. The complementarity of characteristics from both domains can provide more accurate and reliable results in the lane detection task.

Chapter 5

Methodology

This chapter describes the methodology employed for conducting the experiments, as well as the framework implemented for the sensor fusion task. After presenting the state-of-the-art on lane estimation, and more specifically, sensor fusion for this task, the optimal methodology remains unknown. As such, three different types of fusion strategies were implemented and studied and their architecture is depicted at the end of this chapter.

Semantic segmentation was the chosen methodology to perform road and lane estimation. It allows for a detailed understanding of the scene with a high level of granularity, which is crucial for this task. From the state-of-the-art architectures, SalsaNext [44] presents a good performance in terms of accuracy while maintaining efficient inference speed. Due to its ability to capture fine-grained data and manage variances, as well as its multi-scale feature fusion and effective information flow, it results in reliable predictions. As such, this model was used to perform road and lane estimation. This chapter further delves into this architecture and its characteristics.

5.1 SalsaNext Architecture

SalsaNext [44] was the chosen architecture to perform road and lane segmentation and can be seen as the next version of SalsaNet [45]. This model uses a bottleneck compression rate of 16 and adheres to the standard encoder-decoder design. The decoder portion mixes upsampled features from the residual blocks and the encoder section has a collection of ResNet blocks. In contrast to SalsaNet, the ResNet encoder blocks were swapped for new residual expanded convolution stacks with progressively larger receptive fields, and a pixel-shuffle layer was added to the decoder. The dropout method was modified by omitting the first and last network layers in the dropout process, which increased the functions of very fundamental features (such as edges and curves) in the segmentation process. This types of features are of great significance when it comes to roads, lanes and road markings. Average pooling was used in place of stride convolutions in the encoder to create a lightweight model, a crucial characteristic for real-time applications since the segmentation of driving scenarios requires a fast performance.

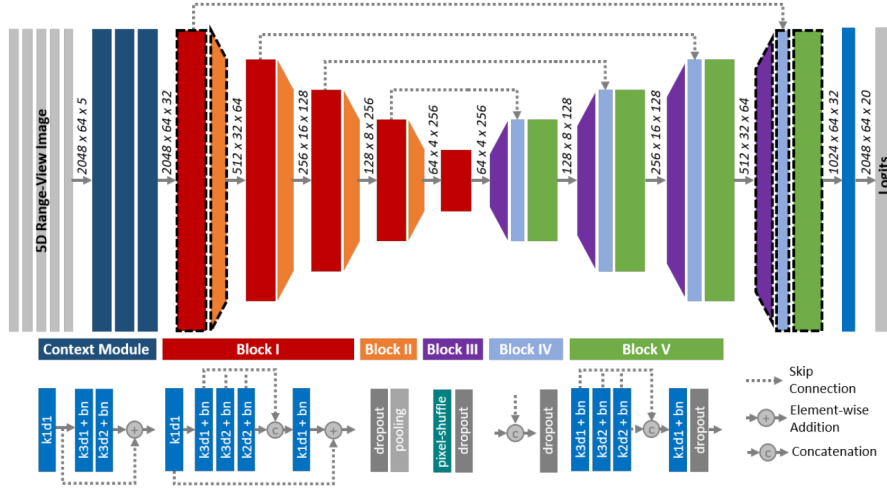


Figure 5.1: SalsaNext architecture (extracted from [44]). Blocks whose edges are dashed are ones that do not use the dropout. The kernel size, dilation rate, and batch normalization are represented by the layer elements k , d , and bn , respectively. Blocks I and II are part of the encoder, whereas Blocks III, IV and V constitute the decoder portion. At the end of the network there is a classification layer.

The encoder in SalsaNext employs a series of stacked sparse convolutional layers, followed by batch normalization and ReLU activation functions. These layers progressively extract hierarchical features from the input point cloud, capturing both local and global contextual information. The use of sparse convolutions allows for the efficient processing of sparse point cloud data.

The decoder part of SalsaNext performs the upsampling operation to recover the spatial resolution of the segmented point cloud. It uses transposed sparse convolutional layers combined with skip connections from the corresponding encoder layers. These skip connections help to preserve fine-grained details during the upsampling process.

To enhance the robustness and uncertainty estimation of the segmentation, SalsaNext introduces Monte Carlo dropout sampling during inference. During test time, dropout is applied, and multiple forward passes are performed with different dropout masks. This process generates an ensemble of predictions, which is then used to estimate the per-point uncertainty of the segmentation results. This uncertainty estimation is valuable in scenarios where confidence or reliability information is required for the segmentation outputs.

5.1.1 Contextual Module

The network's dearth of contextual data is one of the primary problems with semantic segmentation. To solve this, SalsaNext has a residual dilated convolution stack that merges a bigger receptive field with a smaller one by adding 1×1 and 3×3 kernels right at the beginning of the network to aggregate the context information in distinct areas. This aids in the acquisition of both more precise geographical data and the larger environment.

5.1.2 Dilated Convolution

Extracting spatial characteristics requires the use of receptive fields, and enlarging the size of the kernel would be a simple way to capture additional descriptive spatial details. However, this has the disadvantage of significantly increasing the number of parameters. Instead, a new combination of a series of dilated convolutions was created, with effective receptive fields of 3, 5, and 7 (Block I in Fig. 5.1) to swap out the ResNet blocks in the original SalsaNet encoder. In order to provide the network additional access to the fused features emanating from different depths in the receptive field, each dilated convolution output is concatenated, and a 1×1 convolution followed by a residual connection is performed. Every residual dilated convolution blocks precedes a dropout and a pooling layers.

5.1.3 Pixel-Shuffle Layer

Transpose convolutions, which have a high parameter count and are computationally costly layers, are used in the original SalsaNet decoder. By switching the pixels from the channel dimension to the spatial dimension, the pixel-shuffle layer [63] (Block III in Fig. 5.1), which takes advantage of the learned feature maps to construct the upsampled feature maps, replaces these conventional transpose convolutions. Before supplying the pixel-shuffle outputs to the skip connection (Block IV in Fig. 5.1), the filters are also doubled on the decoder side.

5.1.4 Central Encoder-Decoder Dropout

This technique involves inserting dropout only in the central encoder and decoder layers of the network while excluding the first and last layers. The reason behind this is that the lower layers of the network are responsible for extracting fundamental features like edges and corners, which remain consistent throughout the data distribution. If these layers are dropped out or removed, it hampers the network's ability to effectively construct more complex features in the deeper layers. By applying dropout to the intermediate layers, the network prevents overfitting and encourages better generalization by preventing the formation of overly dependent features.

5.1.5 Average Pooling

The downsampling in the original SalsaNet model was done using strided convolutions, which adds new learning parameters. SalsaNext shifts to average pooling for the downsampling and as a result less RAM is used. Average pooling reduces the spatial dimensions of the feature maps by taking the average value within each pooling region. It helps to downsample the feature maps and retain the most salient information, reducing computational requirements and providing a form of regularization in the network. In addition, a 1×1 convolution is applied to align the channel numbers with the overall number of semantic classes following the decoder unit. The soft-max classifier receives the final feature map and computes the classification scores for pixels.

To address the internal covariant shift, each convolution layer in the SalsaNext model uses a leaky-ReLU activation function and is followed by batch normalization. Afterwards, dropout is added, following batch normalization.

5.2 Framework Implementation

As mentioned in Section 3.4, there are multiple benefits of combining two sensor modalities for perceptual tasks. Since different sensors have distinct natures, their complementarity can be explored for a more accurate estimation. In this work, three different types of fusion strategies were explored in order to find the optimal methodology.

Originally, the SalsaNext architecture was developed to conduct semantic segmentation of 3D LiDAR point clouds using a range view representation. However, to proceed with the work described in this document, several modifications were made to the original architecture.

The work on SalsaNext was built upon the baseline implementation from the THEIA project.

5.2.1 Early Fusion

In the early fusion strategy, the network must learn features that mix both sensory modalities from scratch. In this instance, fusion occurs at a very low abstraction level and the raw RGB and LiDAR information are calibrated, concatenated, and fed to the network. The pipeline implemented for the early fusion methodology is represented in Fig. 5.2.

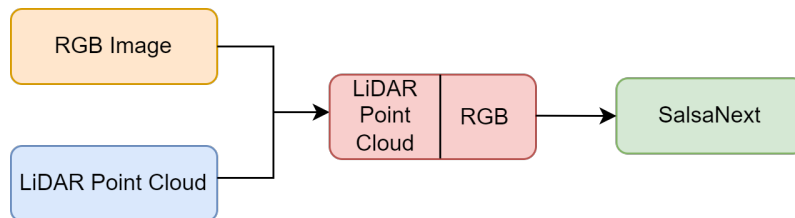


Figure 5.2: Proposed early fusion framework. The LiDAR Point Cloud was processed and concatenated in the depth dimension with the RGB image before feeding the result to the SalsaNext network.

5.2.2 Late Fusion

In late fusion, two different and parallel streams process the RGB and LiDAR images independently, providing two separate predictions. As such, the SalsaNext architecture was not modified, but an additional fully connected network was added at the end, as shown in Fig 5.3. The merging of LiDAR and camera data takes place at the very end of two distinct processing branches. Fusion in this situation takes place at the decision level.

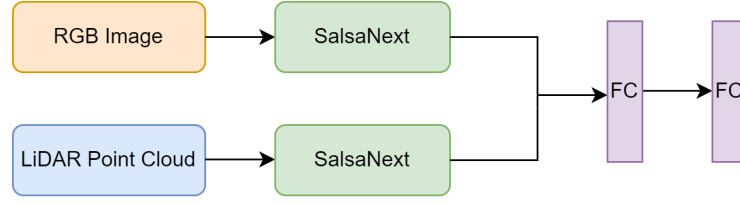


Figure 5.3: Proposed late fusion framework. The predictions from the RGB and the LiDAR branch were concatenated in the depth dimension and then fed to the network composed by the two fully connected layers.

5.2.3 Middle Fusion

Finally, for the middle fusion strategy, the SalsaNext network was modified to contain two different context modules, one for the RGB images and another for the LiDAR point clouds. These two context modules were concatenated in the depth dimension before feeding the result to the rest of the network. Instead of 32 channels like in the original architecture, the input for the first ResNet block had 64 channels (32 from the LiDAR context module and 32 from the RGB context module). An illustration of this modification is presented in Fig. 5.4. We also tested the robustness of this architecture to the lack of calibration parameters between camera images and LiDAR points.

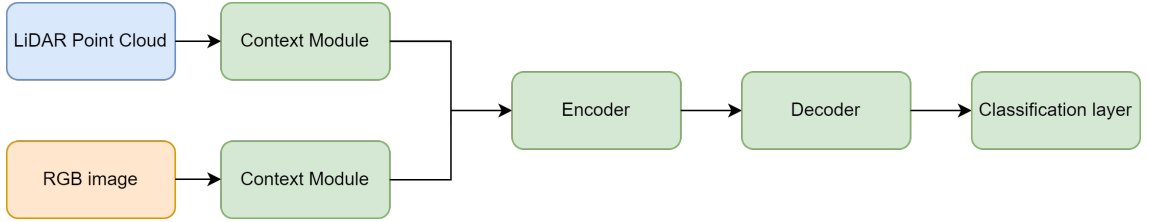


Figure 5.4: Proposed middle fusion framework. The results of the two context modules were concatenated in the depth dimension and given as an input to the encoder.

5.3 Objective Functions

In SalsaNext, a hybrid objective function was used, linearly combining the softmax cross-entropy loss with the *Lovasz-Softmax* loss [64]. In order to add more value to underrepresented classes, the softmax cross-entropy loss was weighted with the inverse square root of the class frequency as:

$$L_{\text{wce}}(y, \hat{y}) = - \sum_i \alpha_i p(y_i) \log(p(\hat{y}_i)), \alpha_i = \frac{1}{\sqrt{f_i}} \quad (5.1)$$

where y and $\hat{y}$ refer to the true and predicted class labels, f_i is the frequency of the i^{th} class and p is the probability distribution. The *Lovasz-Softmax* loss aimed to maximize the IoU score, i.e., the

Jaccard index following the given equation:

$$L_{\text{ls}} = \frac{1}{|C|} \sum_{c \in C} \overline{\Delta_{J_c}}(m(c)), m_i(c) = \begin{cases} 1 - x_i(c) & \text{if } c = y_i(c) \\ x_i(c) & \text{otherwise} \end{cases} \quad (5.2)$$

where $|C|$ stands for the class number, $\overline{\Delta_{J_c}}$ characterizes the Lovasz extension of the Jaccard index, $x_i(c) \in [0, 1]$ and $y_i(c) \in \{-1, 1\}$ retain the class c predicted probability and the pixel i 's ground truth label, respectively. The $m_i(c)$ is the vector of pixel errors for class c .

The late fusion network composed by two fully connected layers used the NLL loss as its primary objective function.

5.4 Final Remarks

Semantic segmentation is a well-suited approach for road and lane segmentation due to its ability to assign pixel-level labels to an image. Since the border of the road and lanes are of utmost importance, this methodology allows for the delineation of different classes or regions within the scene.

The three fusion strategies presented aim to improve the segmentation compared to unimodal approaches. The combination of the camera and LiDAR information can provide more accurate and reliable results, but since there is not a consensual answer on the best fusion strategy, further research is needed to explore the advantages and disadvantages of each strategy.

In the next chapter, the experimental setup for the experiments conducted is described.

Chapter 6

Experimental Setup

This chapter describes the experimental setup adopted in the work conducted during this dissertation. First, it depicts the datasets used to train and validate the proposed solutions. Then, it presents the data pre-processing steps and describes the road and lane estimation inputs for the different fusion strategies, as well as the training parameters employed.

6.1 Datasets

6.1.1 KITTI-ROAD Dataset

The KITTI-ROAD dataset [65] consists of synchronized camera pictures and LiDAR point clouds with ground truth, calibration parameters, and evaluation scripts. The RGB images vary in resolution, ranging from 320×1216 to 376×1248 . Each LiDAR point cloud has 4 channels (x, y, z, reflectivity) and a field of view of 360° . This dataset is extracted from the KITTI dataset at a minimum spatial distance of 20m and is composed of two parts: a training set consisting of 289 frames with real-world examples and a testing set constituted of 290 images with only online evaluation, both gathered throughout several days in different environments such as city, rural and highway. All frames present situations of relatively low traffic density meaning the road is almost always completely visible.

To avoid obscuring the advantages of integrating camera pictures with additional sensing modalities, it is crucial to note that the majority of the training samples were taken in very good weather and lighting circumstances. Road sceneries are divided into three categories: urban marked (UM), urban multiple marked (UMM), and urban unmarked (UU) as described in Table 6.1. A fourth category, urban road (URBAN_ROAD), was also taken into account when calculating a cumulative score for the first three sets.

Annotations for two different road terrain types are available: "road" refers to the road area, and "lane" refers to the ego-lane, *i.e.*, the lane the vehicle is currently driving on. However, the latter is only available for the UM category reaching a total of 95 training frames with annotations for this class. The annotation files are provided as label images in the camera space.

Table 6.1: Dataset statistics of the KITTI-ROAD dataset. Extracted from [65].

Abbreviation	# train	# test	Description
UU	98	100	urban unmarked
UM	95	96	urban marked two-way road
UMM	96	94	urban marked multi-lane road
URBAN	289	290	all three urban subsets

Both a perspective view and a bird’s eye view (BEV) are available for the pixel-based evaluation, and the maximum F1-measure (MaxF), average precision (AP), precision (PRE), recall (REC), false positive rate (FPR), and false negative rate (FNR) are the metrics employed. The MaxF in BEV is the main metric for comparing various approaches in the KITTI-ROAD dataset.

Since the ground truths for the test set of the KITTI-ROAD dataset are not provided, and the submissions for the online evaluation are limited to peer-reviewed and submitted papers, for now, we decided to split the given official training set into train, validation, and test according to Table 6.2. For the ego-lane estimation task, the split adopted is presented in Table 6.3. Once the methodology implemented is validated, we could proceed to the submission for the online evaluation.

This dataset was selected because it provides annotations for both road and ego-lane segmentation. Additionally, it includes information from the two modalities under study (camera and LiDAR).

Table 6.2: Split of KITTI-ROAD dataset used for road estimation.

Set	# samples
Train	214
Validation	25
Test	50

Table 6.3: Split of KITTI-ROAD dataset used for ego-lane estimation.

Set	# samples
Train	66
Validation	9
Test	20

6.1.2 Bosch Dataset

This is a Bosch proprietary dataset. The total number of frames annotated with lane points come to a total of 2946 at the time of the writing of this document. These frames were split into train, validation, and test sets as presented in Table 6.4.

For each LiDAR point cloud there is a corresponding camera image. However, the ground truths are only available for the LiDAR information. Moreover, the calibration parameters between the camera and the LiDAR were not ready.

Table 6.4: Split of the Bosch dataset used.

Set	# samples
Train	2072
Validation	460
Test	414

In contrast to the KITTI-ROAD dataset, the Bosch dataset contains distinct weather conditions such as rainy and foggy scenarios, as well as daytime and nighttime sequences. The frames were gathered throughout several days in distinct environments such as highways, rural areas, cities, and tunnels. The data was captured by one LiDAR sensor operating at a scan frequency of 10Hz.

Regarding semantic segmentation, this dataset provides labels for 40 classes. However, due to the focus of this dissertation we only considered the “road” and “lane markings” classes, while the remaining classes were assigned to the ‘background’ class.

The Bosch dataset was used due to its higher resolution compared to KITTI-ROAD, as well as the availability of lane lines annotations. Moreover, this dataset is specific from the THEIA project.

6.2 Data Preprocessing

6.2.1 KITTI-ROAD

Since the rectification process resulted in the RGB pictures to having different sizes, a cropping operation was used to ensure that each training example and its corresponding ground truth had the same size of 320×1216 . The cropping was performed ensuring that road pixels were not discarded.

Several types of representations of the LiDAR points were tested, mainly the projection to the camera space and the range view representation. For the first approach, LiDAR point clouds were transformed into three-channel tensors that have the same width and height as camera images and have one 3D spatial coordinate encoded for each channel. It is simple to construct relationships between colour intensities and 3D information by using this representation. The Velodyne HDL64 produces a point cloud of about 100k points, each of which is identified by its spatial coordinates in the LIDAR coordinate system, $p = [x, y, z, 1]^T$. It is possible to calculate the column, u , and row, v , positions where its projection intersects the camera plane using the LiDAR-camera transformation matrix, T , the rectification matrix, R , and the camera projection matrix, P . The transformation given by equation (6.1) was applied to every point in the point cloud and the ones that fell outside the image or were not in front of the camera ($z < 0$) were discarded.

$$[u, v, 1]^T = PRTp \quad (6.1)$$

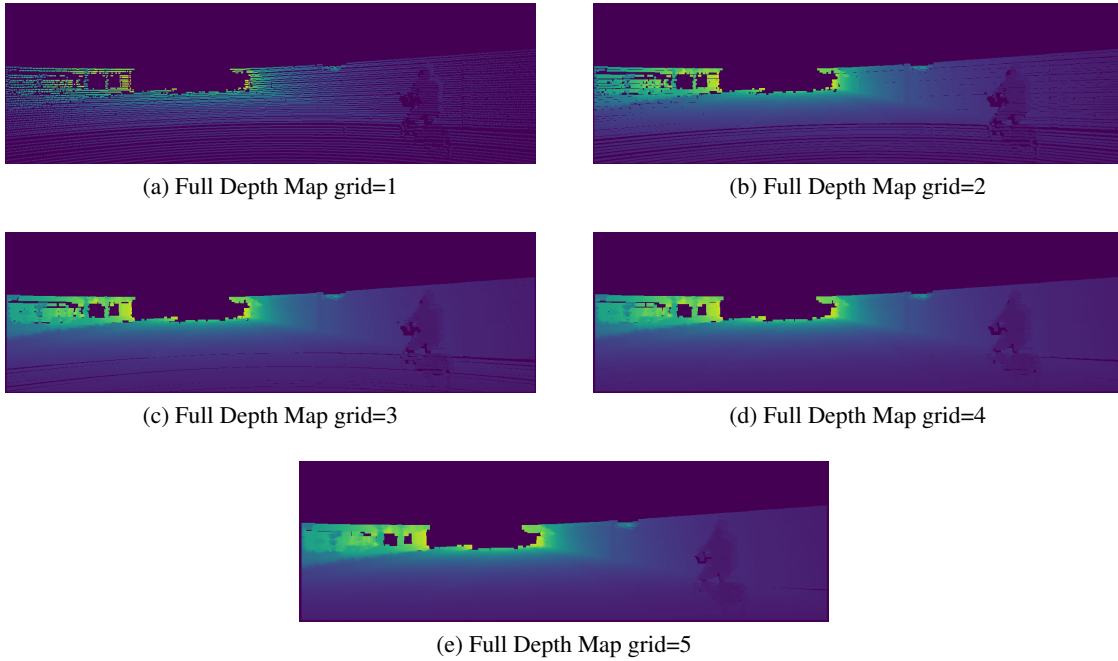


Figure 6.1: Depth maps resulting from the upsampling operation.

However, we noted that most pixels of the resulting image were set to zero because none of the laser beams hit the corresponding regions. To mitigate this fact, an upsampling¹ operation was performed on the LiDAR points by calculating the depth information for every (x, y) location and its neighbourhood. For instance, when the grid is equal to 4, a 9×9 neighbourhood is used, and a weighted depth value is calculated according to the distance of those neighbours. The distance of the neighbourhood operates as the weight of the sum process and the quality of the depth maps increases with the grid size. However, so does the computational time. Fig. 6.1 shows an example of the resulting depth maps depending on the size of the grid chosen. This transformation has the disadvantage of discarding the reflectivity values collected by the LiDAR. Applying the same operation to the reflectivity channel could potentially have a negative effect on the model's performance, particularly in scenarios where sharp edges are an important learning feature, such as lane detection. The size of the resulting depth maps was 320×1216 .

To perform the range view transformation, since the annotations for the point clouds were not given, we had to perform an operation to obtain the labels in the range view. Using the transformation given by equation (6.1), it was possible to obtain the column and row positions where the projection of the points intersects the camera plane. Then, for each of these positions, the corresponding label was gathered from the original label image, and a new range view image was assembled with this information. The range view projections had a size of 512×64 , with a vertical FOV of $(-25^\circ, 3^\circ)$ and a horizontal FOV angle of 90° .

¹Code for the upsampling operation available on: <https://github.com/balcilar/DenseDepthMap>

6.2.2 Bosch Dataset

Given that at the time of the writing of this document, the calibration parameters between the camera images and the LiDAR point clouds were not available, we were not able to perform the same transformations as for the KITTI-ROAD dataset. The LiDAR point clouds were projected into a range view representation of size 512×256 , with a vertical FOV of $(-9^\circ, 9^\circ)$ and a horizontal FOV angle of 52.51° .

6.3 Data Augmentations

Some data augmentation techniques were implemented to train the networks, creating new artificial training data from existing samples. This strategy helps to increase the diversity of the data leading to a better generalization capability of the model. Given the limited size of the KITTI-ROAD dataset, a simple mirror-flipping operation was performed to increase its size by a factor of 4, (2 for LiDAR information and 2 for camera images). Additionally, the LiDAR point clouds have also undergone a simple normalization procedure.

6.4 Road and Lane Estimation

Due to the lack of public datasets with lane line annotations for semantic segmentation, we performed road and ego-lane² estimation on the KITTI-ROAD dataset. Since there were only 95 frames with ego-lane annotations we used the pre-trained models in road estimation for lane estimation as well. In this way, the models already had a head start from a similar task. The Bosch dataset was used to perform road and lane markings segmentation.

In order to study the impact of combining two distinct modalities for road and lane segmentation we performed several experiments with only one type of sensor as well as the combination of both LiDAR and camera information.

6.4.1 Unimodal

The RGB and LiDAR data were used separately to perform the segmentation tasks. This approach allowed us to assess the influence of each modality in multi-sensor approaches. In addition, all representations of LiDAR point clouds described in Section 6.2.1 were used to train the SalsaNext model so that we could understand the impact of the different projections on the overall performance.

6.4.2 Multimodal

As described in Section 5.2, three types of fusion techniques were developed and for each one several experiments with different data representations were performed. It is important to denote that

²the lane the vehicle is currently driving on

the Bosch dataset was only used to train the middle fusion network without calibration parameters due to the limitations of the data.

- **Early Fusion**

Given the sparseness of LiDAR points when projected into the camera space, the raw fusion of characteristics between LiDAR points and RGB images would not benefit from the dense information of the latter. To mitigate this fact, an upsampling operation was performed to the LiDAR points as mentioned in Section 6.2.1. The resulting dense depth image was concatenated with the RGB images in the depth dimension resulting in a tensor of size $4 \times H \times W$, where the channels were (depth, R, G, B). In order to study the impact of the depth maps' density, the network was trained with maps resulting of several grid sizes ranging from grid=2 to grid=5. The grid size of one was not tested since it was similar to the projection without upsampling.

- **Late Fusion**

In late fusion, two concurrent streams processed the RGB and LiDAR images separately. For the LiDAR branch, two different representations were adopted: the point clouds were converted into a collection of 2D images with the same spatial resolution as the camera images; the dense depth maps with several grid sizes. The outputs were concatenated in the depth dimension and fed through the model composed of two fully connected layers.

In order to train the network that combined the results of the two branches into a single prediction, we used the results of the SalsaNext model, in the training set, for both modalities.

- **Middle Fusion**

For the middle fusion, two context modules were created, one for the RGB information and another for the LiDAR points. We tested the robustness of this architecture to the lack of calibration parameters by transforming the 3D point clouds into a 2D LiDAR range view image and resizing the camera images to the same size as the range view samples. In this way, there was no direct correspondence between the pixels of the range view image and the RGB image. The labels used were also in range view.

Another approach was employed, utilizing the calibration parameters between the LiDAR and the camera. The LiDAR point clouds were transformed into the dense depth maps with several grid sizes and the RGB images maintained their original size. As such, we could conclude about the necessity of pre-calibrating the inputs of the two sensors or if the network was capable of learning it itself.

6.5 Training Setup

SalsaNext was end-to-end trained using the ADAM optimizer and an initial learning rate of 0.01, that during training was decayed by 0.01 after each epoch. Additionally, an L2 penalty was used with $\lambda=0.0001$ and a momentum of 0.9. For the KITTI-ROAD dataset, it was trained with a batch

size of four, dictated by GPU memory restrictions. All training's had a duration between 50 and 120 epochs. The maximum number of epochs was defined as 150 with an early-stopping callback, and the patience value was equal to 30. The late fusion network was also trained with the ADAM optimizer in 200 epochs with a learning rate of 0.01.

All training processes were performed on a Tesla V100-SXM2 32GB GPU with an Xeon Gold 6150 CPU.

Chapter 7

Results and Discussion

This section reports the results obtained from the experiments carried out. First, it presents and analyses the quantitative results for every road and lane category. Then, it performs a qualitative analysis by showcasing some predictions of our models. Finally, it extensively analyses the fusion methodologies and draws conclusions from the results obtained. It is important to note the abbreviations adopted during this analysis: "LiDAR in cam space" refers to the simple projection of the LiDAR points into the camera space without any processing; "LiDAR RV" denotes the LiDAR range view representation; "LiDAR DDM" refers to the LiDAR dense depth maps with several grid sizes.

7.1 Road Estimation

As noted in Section 6.1.1, the evaluation in the KITTI-ROAD benchmark is done in separate categories: UM_ROAD, UMM_ROAD, UU_ROAD, and URBAN_ROAD. The evaluations were conducted in the BEV perspective to be consistent with the online evaluation in the KITTI-ROAD benchmark. However, the networks that performed the predictions in range view were evaluated in that projection. In the case of the Bosch dataset, the evaluations were conducted using the range view projection. The results for road estimation on this dataset will be referred to as "Bosch Road" whereas the results for lane line estimation will appear as "Bosch Lane" throughout this dissertation.

7.1.1 Urban Marked (UM_ROAD)

For the Urban Marked category, the quantitative results in Table 7.1 show that the best-performing architecture was the middle fusion of the RGB images with the LiDAR dense depth maps with a grid size of four. This approach achieved a MaxF rate of 96.39%, a small FPR of 2.67%, and high values of precision and recall.

The model performed worse when the input was the projected LiDAR points in the camera perspective. When compared to the raw LiDAR coordinates, dense and accurate road recognition results are considerably more easily obtained from high-level feature maps with rich semantic

information. Thus, this simple projection without any pre-processing was not ideal for road estimation. On the other hand, the LiDAR dense depth maps have presented promising results.

The methods based on the projection of the LiDAR point clouds into the range view representation presented a high rate of false positives. This is because the colours, textures, and heights of the road regions and non-road areas are quite similar, making it difficult for features retrieved from LiDAR point clouds to distinguish between them. The compression resulting from the transformation into this representation might have also affected this metric since the division between the road and non-road areas was possibly more overlapped.

In the early fusion modality, the increase of the grid size in the LiDAR dense depth maps had minimal effects on the MaxF value that ranged between 96.04% and 96.39%. With values of precision ranging from 95.48% to 96.39% and recall numbers from 95.99% to 97.48%, this fusion presented a balanced performance. Moreover, the FPR going from 2.42% to 3.12% suggested an improvement compared to the "LiDAR RV" and "LiDAR in cam space" unimodal networks.

For the late fusion, the use of LiDAR dense depth maps achieved significantly better results compared to the employment of the projected LiDAR points in the camera space. These dense maps, particularly with a grid size of three, reduced by 4.78% the FPR and increased the MaxF value by 2.7%.

Since we could not use the same test set to evaluate our algorithms due to access limitations, it was not possible to directly compare our results with the benchmarking in the KITTI-ROAD dataset. Nevertheless, Table 7.2 presents a comparison between our best model and four of the best models in the KITTI-ROAD benchmark.

Our best method does not surpass any of the existing KITTI-ROAD benchmark architectures for the MaxF metric. However, for the average precision metric, we achieved better results than any of the state-of-the-art models, with an improvement of 1.87% in relation to the largest value in the leaderboard. However, these results could be affected by the different test frames and the fact that the number of samples used to train our algorithms was less than the ones used by the methods depicted in the leadership board. As such, these are promising results that could be improved with access to the full dataset.

7.1.2 Urban Multiple Marked (UMM_ROAD)

In the UMM category, the best performing algorithm, in terms of quantitative results, was the middle fusion of the LiDAR range view images with the camera images resized to the same shape. This implementation does not require calibration parameters between the camera and LiDAR, something that has not been widely explored in the literature and especially in the architectures tested in the KITTI-ROAD dataset.

Table 7.3 exhibits the results obtained for the different fusion strategies and inputs. The second best-performing architecture was the early fusion of the LiDAR dense depth map with grid size four and the RGB image.

Despite the comparatively poorer results obtained through the late fusion of LiDAR DDMs and camera information when compared to other fusion techniques, this approach has demonstrated an

Table 7.1: Road estimation results for the UM_ROAD category in KITTI_ROAD, with data augmentation.

Input	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Unimodal						
RGB	95.77	90.60	95.63	95.91	2.95	4.09
LiDAR in cam space	90.88	84.59	89.02	92.91	7.72	7.19
LiDAR RV	96.31	95.25	95.25	97.39	4.75	2.61
LiDAR DDM						
grid=2	96.04	90.74	95.79	96.30	2.85	3.70
grid=3	96.38	90.52	95.54	97.23	3.06	2.77
grid=4	96.39	90.99	96.06	96.72	2.67	3.28
grid=5	95.75	91.38	96.49	95.01	2.33	4.99
Early fusion						
LiDAR DDM+RGB						
grid=2	96.47	90.46	95.48	97.48	3.12	2.52
grid=3	96.19	91.29	96.39	95.99	2.42	4.01
grid=4	96.45	90.82	95.88	97.04	2.82	2.96
grid=5	96.35	91.27	96.37	96.32	2.45	3.68
Middle fusion						
LiDAR RV/RGB	96.03	90.03	94.38	97.74	5.62	2.26
LiDAR DDM/RGB						
grid=2	96.28	90.30	95.30	97.27	3.23	2.73
grid=3	96.36	91.57	96.69	96.03	2.21	3.97
grid=4	96.76	91.54	96.66	96.86	2.26	3.14
grid=5	96.43	91.45	96.57	96.28	2.31	3.72
Late fusion						
LiDAR in cam space/RGB	93.24	85.07	89.55	97.24	7.66	2.76
LiDAR DDM/RGB						
grid=3	95.94	90.78	95.79	96.10	2.88	3.90
grid=4	95.43	90.37	95.35	95.51	3.18	4.49

Table 7.2: Comparison between our best performing model in the UM_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark. The state-of-the-art models presented consist of the three top-performing architectures in the MaxF score and the best-performing model in the AP metric.

Method	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Our best model						
Middle fusion LiDAR DDM/RGB (grid=4)	96.76	95.54	96.66	96.86	2.26	3.14
State-of-the-art						
RD-RoTr [66]	97.28	92.66	97.37	97.19	1.20	2.81
SNE-RoadSegV2 [58]	97.25	93.52	97.48	97.03	1.14	2.97
PLARD [60]	97.05	93.53	97.18	96.92	1.28	3.08
SNE-RoadSeg [67]	96.42	93.67	96.59	96.26	1.55	3.74

enhancement over the architecture solely reliant on RGB information. Notably, the incorporation of depth values appears to have effectively complemented the dense camera information, leading to improved estimation outcomes.

The RGB modality attained a MaxF of 96.19%, demonstrating a high degree of road estimating accuracy. However, it tended to mistakenly categorize non-road pixels as road, as seen by its considerably higher false positive rate (FPR) of 2.18%.

Several grid sizes for the "LiDAR DDM" modality were explored. The MaxF, which ranges from 96.10% to 96.83%, remained largely constant as the grid size increased. Consistently high precision (PRE) and recall (REC) scores showed a balanced trade-off between precise detection and few false negatives. The false positive rate (FPR) ranged between 2.65% and 3.09%, and the false negative rate (FNR) varied between 3.18% and 5.00%. These findings imply that LiDAR DDMs perform well in road estimate, especially with a grid size of 3.

For the early fusion strategy, increasing the grid size resulted in negligible differences in MaxF, ranging from 96.65% to 97.21%, similar to the individual modalities. Notably, as compared to the findings from unimodal RGB and LiDAR projected in the camera space, the fusion of LiDAR DDM and RGB consistently improved the FPR values. With the precision value ranging from 96.55% to 98.15% and the recall ranging from 95.19% to 96.76%, the fusion produced a balanced performance.

This was the worst performing category of all and according to Table 7.4, the results obtained do not transcend the ones of the state-of-the-art. Nevertheless, the MaxF value is only 0.75% lower than the best-performing algorithm.

7.1.3 Urban Unmarked (UU_ROAD)

In the Urban Unmarked category, similarly to the Urban Marked, the overall best performance was achieved by the middle fusion of the LiDAR DDM (grid=4) with the RGB images with a MaxF

Table 7.3: Road estimation results for the UMM_ROAD category in KITTI_ROAD, with data augmentation.

Input	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Unimodal						
RGB	96.19	93.33	97.71	94.72	2.18	5.28
LiDAR in cam space	91.52	86.63	90.34	92.73	9.73	7.27
LiDAR RV	96.68	93.92	96.53	96.83	3.47	3.17
LiDAR DDM						
grid=2	96.52	92.82	97.15	95.90	2.76	4.10
grid=3	96.83	92.55	96.85	96.82	3.09	3.18
grid=4	96.68	92.56	96.86	96.49	3.07	3.51
grid=5	96.10	92.90	97.24	95.00	2.65	5.00
Early fusion						
LiDAR DDM+RGB						
grid=2	97.21	93.29	97.67	96.76	2.26	3.24
grid=3	96.65	93.73	98.15	95.19	1.76	4.81
grid=4	97.19	93.44	97.84	96.55	2.09	3.45
grid=5	96.86	93.61	98.01	95.74	1.90	4.26
Middle fusion						
LiDAR RV/RGB	97.38	93.97	96.87	97.89	3.13	2.11
LiDAR DDM/RGB						
grid=2	96.98	92.95	97.29	96.67	2.64	3.33
grid=3	96.76	93.63	98.04	95.50	1.87	4.50
grid=4	97.13	93.93	98.38	95.91	1.55	4.09
grid=5	96.57	93.26	97.64	95.52	2.27	4.48
Late fusion						
LiDAR in cam space/RGB	94.06	87.75	91.58	96.69	8.73	3.31
LiDAR DDM/RGB						
grid=3	96.70	92.76	97.20	96.20	2.59	3.80
grid=4	96.01	92.77	97.21	94.84	2.54	5.16

Table 7.4: Comparison between our best performing model in the UMM_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark. The state-of-the-art models presented consist of the four top-performing architectures in the MaxF score.

Method	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Our best model						
Middle fusion LiDAR RV/RGB	97.38	93.97	96.87	97.89	3.13	2.11
State-of-the-art						
SNE-RoadSeg+ [57]	98.13	95.52	98.01	98.25	2.19	1.75
SNE-RoadSegV2 [58]	98.10	95.63	97.98	98.22	2.23	1.78
RD-RoTr [66]	98.09	94.74	97.74	98.45	2.50	1.55
LRDNet+ [67]	97.98	94.28	97.67	98.29	2.58	1.71

score of 96.37%. According to Table 7.5, this is followed by the early fusion of the LiDAR DDM (grid=4) with the RGB images. The late fusion network achieved the worst results of the three types of fusion strategies.

When it comes to the unimodal networks, the MaxF value of the LiDAR range view representation increased to 93.44% compared to the LiDAR in camera space. This improvement suggests that range view images provide more relevant information for road estimation.

For every network using the LiDAR dense depth maps, the MaxF score remained consistent along the different grid sizes. However, the early fusion of the LiDAR DDM with a grid size of four achieved the highest value of 97.19%.

The middle fusion in the range view domain presented results slightly lower compared to early fusion but still surpassed all unimodal networks in terms of performance.

Table 7.6 compares our best-performing architecture in this category with the state-of-the-art models. Although our network did not outdo the MaxF metric, the results were quite close and could potentially be larger with access to the complete dataset.

7.1.4 URBAN_ROAD

The Urban Road category is a combination of the three categories above-mentioned, i.e., it aggregates the results of all samples. In quantitative terms, the network with the best performance was the middle fusion of the LiDAR DDM (grid=4) with the RGB images, as presented in Table 7.7. This was not surprising given that this method was the best performing in both UM and UU categories.

Among the unimodal strategies, the LiDAR DDM (grid=4) achieved the highest MaxF score of 96.10%. It had a precision of 95.65% and an equally large recall value of 96.56%. Moreover, it achieved the lowest false negative rate when compared with the remaining unimodal networks.

The MaxF results for the early fusion methodology ranged from 96.11% to 96.38% indicating an overall good performance for every grid size of the dense depth maps. The middle fusion in the range view space also achieved a solid MaxF result, although it exhibited a false positive rate of 4.27%, indicating a slightly higher number of incorrect road predictions. The late fusion strategy, as expected from the analysis of the previous results, was the worst-performing fusion strategy.

Ultimately, the results indicate that the fusion of the LiDAR dense depth maps with RGB information consistently improved road estimation performance across different grid sizes. Early fusion techniques provided a fair trade-off between minimum false positives and accurate detection. Middle fusion approaches demonstrated strong outcomes and the LiDAR DDM with grid size of four combined with the RGB information emerged as the top-performing architecture across almost all categories, showcasing superior performance. Late fusion strategies, particularly the fusion of the LiDAR DDM with a grid size of three, have also shown some potential. Nonetheless, this fusion stage exhibited the lowest performance for all categories. Enhancing the results of this fusion strategy may be achieved with a more complex network for combining the predictions from both branches.

Table 7.5: Road estimation results for the UU_ROAD category in KITTI_ROAD, with data augmentation.

Input	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Unimodal						
RGB	95.27	88.80	95.20	95.35	1.59	4.65
LiDAR in cam space	87.88	74.77	85.87	89.99	4.88	10.01
LiDAR RV	93.44	88.92	96.54	90.54	3.46	9.46
LiDAR DDM						
grid=2	92.58	86.88	93.09	92.08	2.25	7.92
grid=3	92.40	87.77	94.07	90.78	1.89	9.22
grid=4	94.76	87.47	93.74	95.79	2.11	4.21
grid=5	93.95	88.99	95.41	92.54	1.47	7.46
Early fusion						
LiDAR DDM+RGB						
grid=2	95.99	89.41	95.87	96.11	1.36	3.89
grid=3	95.72	90.44	97.01	94.48	0.96	5.52
grid=4	95.74	89.13	95.56	95.91	1.47	4.09
grid=5	95.11	89.52	95.99	94.24	1.30	5.76
Middle fusion						
LiDAR RV/RGB	93.74	89.90	96.08	91.51	3.92	8.49
LiDAR DDM/RGB						
grid=2	95.61	88.98	95.40	95.81	1.52	4.19
grid=3	95.24	90.12	96.65	93.88	1.07	6.12
grid=4	96.37	90.21	96.75	95.98	1.06	4.02
grid=5	95.64	88.78	95.17	96.11	1.61	3.89
Late fusion						
LiDAR in cam space/RGB	91.91	82.03	87.75	96.94	4.44	3.51
LiDAR DDM/RGB						
grid=3	94.03	88.28	94.69	93.38	1.66	6.62
grid=4	93.08	88.53	94.98	91.26	1.53	8.74

Table 7.6: Comparison between our best performing model in the UU_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark. The state-of-the-art models presented consist of the four top-performing architectures in the MaxF score.

Method	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Our best model						
Middle fusion LiDAR DDM/RGB (grid=4)	96.37	90.21	96.75	95.98	1.06	4.02
State-of-the-art						
SNE-RoadSegV2 [58]	97.08	92.87	96.83	97.34	1.04	2.66
SNE-RoadSeg+ [57]	97.04	92.97	96.84	97.24	1.03	2.76
RD-RoTr [66]	96.85	91.42	96.73	96.96	1.07	3.04
HEAT [66]	96.40	91.84	95.85	96.96	1.37	3.04

In comparison to the state-of-the-art models from the KITTI-ROAD benchmark, our best-performing model achieved competitive results, as presented in Table 7.8. While our model achieved a MaxF of 96.69%, an AP of 91.18%, and demonstrated a balanced trade-off between precision and recall, it falls slightly behind the top-performing state-of-the-art models. However, there is room for improvement to match or surpass the results in the KITTI-ROAD benchmark if we had access to the ground truth labels for the test set. Therefore, we could provide a more fair and accurate comparison with the state-of-the-art models.

7.1.5 Bosch Road

Road estimation in the Bosch dataset was performed using the LiDAR range view representation, along with the middle fusion without calibration. The remaining approaches explored in the KITTI-ROAD dataset could not be employed for the training with this dataset due to the lack of calibration parameters between the camera and the LiDAR, as well as the absence of labels for the RGB images. As we can see from Table 7.9, the network based solely in the LiDAR data achieved the highest MaxF score of 93.80%. Although the middle fusion of the LiDAR in range view with the RGB images achieved an overall performance slightly lower, with a decrease of 0.10% in the MaxF score, it still presented solid results. Regarding the recall and the FNR metrics, the middle fusion obtained better results in comparison with the unimodal network. As such, both approaches presented promising results with a slight trade-off in precision and recall.

7.2 Lane Estimation

7.2.1 Urban Marked (UM_LANE)

Regarding the ego-lane estimation on the KITTI-ROAD dataset, we noted that overall better results were obtained without performing the flipping operation. This observation may be attributed to the

Table 7.7: Road estimation results for the URBAN_ROAD category in KITTI_ROAD, with data augmentation.

Input	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Unimodal						
RGB	95.68	90.16	95.56	95.81	2.53	4.19
LiDAR in cam space	90.34	83.71	88.46	92.31	6.85	7.69
LiDAR RV	95.90	90.05	96.03	95.77	3.97	4.23
LiDAR DDM						
grid=2	95.43	89.95	95.32	95.55	2.67	4.45
grid=3	95.69	89.92	95.29	96.08	2.70	3.92
grid=4	96.10	90.25	95.65	96.56	2.50	3.44
grid=5	95.43	90.84	96.30	94.57	2.06	5.43
Early fusion						
LiDAR DDM+RGB						
grid=2	96.38	90.15	95.55	97.24	2.58	2.76
grid=3	96.11	91.02	96.50	95.72	1.97	4.28
grid=4	96.33	90.41	95.82	96.84	2.40	3.16
grid=5	96.13	90.85	96.30	95.95	2.09	5.76
Middle fusion						
LiDAR RV/RGB	96.14	91.12	95.73	96.54	4.27	3.46
LiDAR DDM/RGB						
grid=2	96.16	89.95	95.32	97.01	2.71	2.99
grid=3	96.17	91.19	96.69	95.65	1.86	4.35
grid=4	96.69	91.18	96.68	96.71	1.89	3.29
grid=5	96.29	90.86	96.32	96.25	2.09	3.75
Late fusion						
LiDAR in cam space/RGB	93.00	84.41	89.22	97.11	6.67	2.89
LiDAR DDM/RGB						
grid=3	95.59	90.17	95.59	95.58	2.48	4.42
grid=4	95.00	89.89	95.28	94.71	2.63	5.29

Table 7.8: Comparison between our best performing model in the URBAN_ROAD category with the state-of-the-art models from the KITTI-ROAD benchmark. The state-of-the-art models presented consist of the four top-performing architectures in the MaxF score.

Method	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Our best model						
Middle fusion LiDAR DDM/RGB (grid=4)	96.69	91.18	96.68	96.71	1.89	3.29
State-of-the-art						
SNE-RoadSegV2 [58]	97.55	93.98	97.57	97.53	1.34	2.47
RD-RoTr [66]	97.53	92.97	97.32	97.74	1.48	2.26
SNE-RoadSeg+ [57]	97.50	93.98	97.41	97.58	1.43	2.42
SPNet [66]	97.21	93.87	97.38	97.04	1.44	2.96

Table 7.9: Road estimation results in the Bosch dataset, without data augmentation.

Input	MaxF (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Unimodal					
LiDAR RV	93.80	93.31	94.29	6.69	5.71
Middle fusion					
LiDAR RV/RGB	93.70	92.22	95.23	7.78	4.77

fact that this data augmentation technique does not accurately reflect the testing conditions. Since the ego-lane is located on the right side of the road of every test image, the model was trained on scenarios that were not verified frequently during testing. On average, the addition of the flipping operation resulted in a 1% lower MaxF value compared to the results obtained without any form of data augmentation. To improve the results with this data augmentation technique, it would be beneficial to include more test samples with the vehicle driving on the left side of the road.

Among the unimodal methods, the LiDAR dense depth map with a grid size of three achieved the largest MaxF value of 89.14%, as shown in Table 7.11. Both unimodal and early fusion techniques demonstrated an improvement in the MaxF score as the grid size increased, reaching its peak at a size of three. However, performance tended to decrease for larger grid sizes, indicating a compromise between resolution and accuracy. The early fusion of the LiDAR DDMs with the RGB images improved the estimation compared to the corresponding unimodal methodologies. These results suggest that the addition of the dense RGB information to the depth values collected by the LiDAR was complimentary, and the network was capable of successfully combining features from both modalities at an early stage.

Regarding the middle fusion approach, the combination of the LiDAR range view and the RGB images without any calibration was the top-performing architecture, with a significantly high MaxF score of 92.51% comparatively with other methods.

Networks utilizing LiDAR dense depth maps presented higher values of false negative rates, indicating a lower detection rate for lane estimation. This might be due to the absence of reflec-

tivity values in these maps. Since the ego-lane is mostly separated from the rest of the road by a white line in the testing images, its reflectivity value could improve the estimation.

As can be seen in Table 7.11, the FNR values for the unimodal LiDAR DDMs reached 25.54% for a grid size of four. With the concatenation of the RGB information for the early fusion, this value decreased to 13.48% showing that depth information alone is insufficient for accurate lane estimation.

Contrary to the road estimation results, late fusion was the second-best-performing fusion technique, achieving a MaxF value of 90.97% for a grid size of three and a considerably large AP value of 83.95%.

When compared to the state-of-the-art methodologies in the KITTI-ROAD benchmark, our best-performing model achieved fourth place in the leadership board, as presented in Table 7.10. While the CyberMELD+PLARD [68], CyberMELD [68] and RoadNet3 [69] models outperformed our approach by 3.02%, 2.14% and 0.05% respectively, our method presented a relatively low FPR of 1.28%. With access to the ground truths for the test set, our approach had the possibility to further improve its ranking on the leadership board. However, since this information is not available, it is difficult to estimate its impact on the overall results. Nonetheless, considering the good performance of our model, there is potential for achieving higher standings if trained and evaluated in the full dataset.

Table 7.10: Comparison between our best performing model in the UM_LANE category with the state-of-the-art models from the KITTI-ROAD benchmark. The state-of-the-art models presented consist of the four top-performing architectures in the MaxF score.

Method	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Our best models						
Middle fusion LiDAR DDM/RGB (grid=3)	91.42	78.95	92.97	89.92	1.28	10.08
State-of-the-art						
CyberMELD +PLARD [68]	94.44	88.59	95.95	92.97	0.69	7.03
CyberMELD [68]	93.56	88.58	95.94	91.30	0.68	8.70
RoadNet3 [69]	91.47	91.01	91.78	91.17	1.44	8.83
RNet [70]	90.54	82.03	94.92	86.56	0.82	13.44

7.2.2 Bosch Lane

The unimodal LiDAR range view network achieved a MaxF score of 64.08%, indicating a moderate level of overall performance. However, its relatively low precision value of 77.04% and recall score of 54.84% indicates that the model faced several challenges in the detection and localization of lanes. The high values of false positive rate (22.96%) and false negative rate (45.15%) highlight the difficulties faced by the network in correctly estimating lane markings. Further exploration is

Table 7.11: Road estimation results for the UM_LANE category in KITTI_ROAD, without data augmentation.

Input	MaxF (%)	AP (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Unimodal						
RGB	82.29	70.81	91.42	74.82	1.32	25.18
LiDAR in cam space	86.24	73.55	86.36	86.11	2.56	13.89
LiDAR RV	88.76	79.80	95.00	96.45	5.00	3.55
LiDAR DDM						
grid=2	86.89	73.38	86.17	87.62	2.65	12.38
grid=3	89.14	78.90	85.20	93.45	3.06	6.55
grid=4	82.52	71.61	92.52	74.46	1.13	25.54
grid=5	87.07	74.10	95.94	79.71	0.64	20.29
Early fusion						
LiDAR DDM+RGB						
grid=2	86.83	79.43	93.55	81.02	1.05	18.98
grid=3	90.02	77.09	90.70	89.36	1.73	10.64
grid=4	89.39	78.53	92.45	86.52	1.33	13.48
grid=5	88.89	78.27	92.14	85.87	1.38	14.13
Middle fusion						
LiDAR RV/RGB	90.51	79.80	93.82	91.23	5.16	3.54
LiDAR DDM/RGB						
grid=2	87.10	72.37	84.93	89.38	2.99	10.62
grid=3	91.42	78.95	92.97	89.92	1.28	10.08
grid=4	78.67	64.14	91.73	68.87	1.17	31.13
grid=5	81.37	67.93	73.14	91.68	6.35	8.32
Late fusion						
LiDAR in cam space/RGB	81.43	73.90	95.66	70.89	0.61	29.11
LiDAR DDM/RGB						
grid=3	90.97	83.95	90.76	91.18	1.75	8.82
grid=4	88.37	76.00	89.37	87.39	1.96	12.61

needed to improve these results. However, this implementation serves as a good base for future improvements in this task.

In contrast, the middle fusion strategy showed a slightly lower value of the MaxF score, dropping to 62.29%. The precision value of 76.90% and the recall score of 52.35% are comparable to the unimodal network, suggesting a similar performance. However, the false positive rate and false negative rate increased by 0.14% and 2.5%, respectively. Although the middle fusion strategy did not demonstrate a substantial improvement compared to the unimodal network, it is important to take into account the potential influence of the desynchronization between the LiDAR point clouds and the camera images in the Bosch dataset. The fusion between the modalities may have struggled to accurately integrate their complimentary aspects, leading to a degradation in performance. It is crucial to consider the desynchronization factor that might have negatively influenced

the performance of the middle fusion network, offering opportunities for future refinement.

Table 7.12: Lane line estimation results in the Bosch dataset, without data augmentation.

Input	MaxF (%)	PRE (%)	REC (%)	FPR (%)	FNR (%)
Unimodal					
LiDAR RV	64.08	77.04	54.85	22.96	45.15
Middle fusion					
LiDAR RV/RGB	62.29	76.90	52.35	23.10	47.65

7.3 Qualitative Analysis

7.3.1 Urban Marked Road (UM_ROAD)

Figure 7.1 contains an example of the predictions of our networks for one testing frame. Since we experimented with several grid sizes and the differences in performance were minimal, we only included the visual results for the best-performing grid size for each fusion technique. Overall, all networks performed well, although the unimodal RGB framework presented several false negative points on the border of the road, which was not found with the LiDAR-based networks. The fusion of the LiDAR information with the camera images solved this situation. The networks that worked in the range view domain presented a low false negative rate.

From the results presented in the example frame in Figure 7.2, it is possible to infer that all networks that performed an estimation in the camera perspective had some difficulty distinguishing the borders of the road. This can be observed by the blue points on both sides of the road, which represent the misclassified non-road points as road. When the border between the road and the sidewalk was not clearly defined, as was the case in Figure 7.2, false positive detections were occasionally noted. Due to the similar colour and texture of the sidewalk and the road, their intersection was ambiguous for the network. Also, their difference in elevation was minimal on the right side of the road and non-existing on the left side, contributing to the false positives count. Further away from the sensors, the unimodal strategies presented worse results compared to the fusion networks. The number of false negatives in the unimodal RGB and LiDAR in camera space is considerably higher when compared to the fusion models.

7.3.2 Urban Multiple Marked Road (UMM_ROAD)

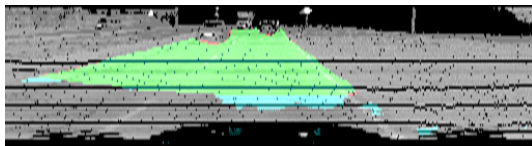
The urban multiple-marked category encompasses several lines of traffic and road markings. Figure 7.3 provides an overview of the prediction results for a test sample. Among the different approaches developed, the projection of the LiDAR points in the camera space without upsampling (Figure 7.3b) achieved the worst results and was characterized by a high rate of false positives and false negatives, indicating a lack of sufficient information for a reliable estimation. Compared to the RGB unimodal result (Figure 7.3a), the fusion methodologies presented an improvement, particularly in reducing the number of false negative points. The inability of the network to identify



(a) Unimodal: RGB



(b) Unimodal: LiDAR in camera space



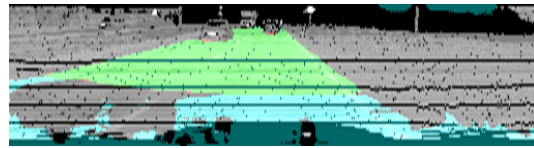
(c) Unimodal: LiDAR range view



(d) Unimodal: LiDAR DDM (grid=4)



(e) Early fusion: LiDAR DDM (grid=2) + RGB



(f) Middle fusion: LiDAR range view + RGB



(g) Middle fusion: LiDAR DDM (grid=4) + RGB



(h) Late fusion: LiDAR in camera space + RGB



(i) Late fusion: LiDAR DDM (grid=3) + RGB

Figure 7.1: Results for the UM_ROAD category in the um_000036 sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

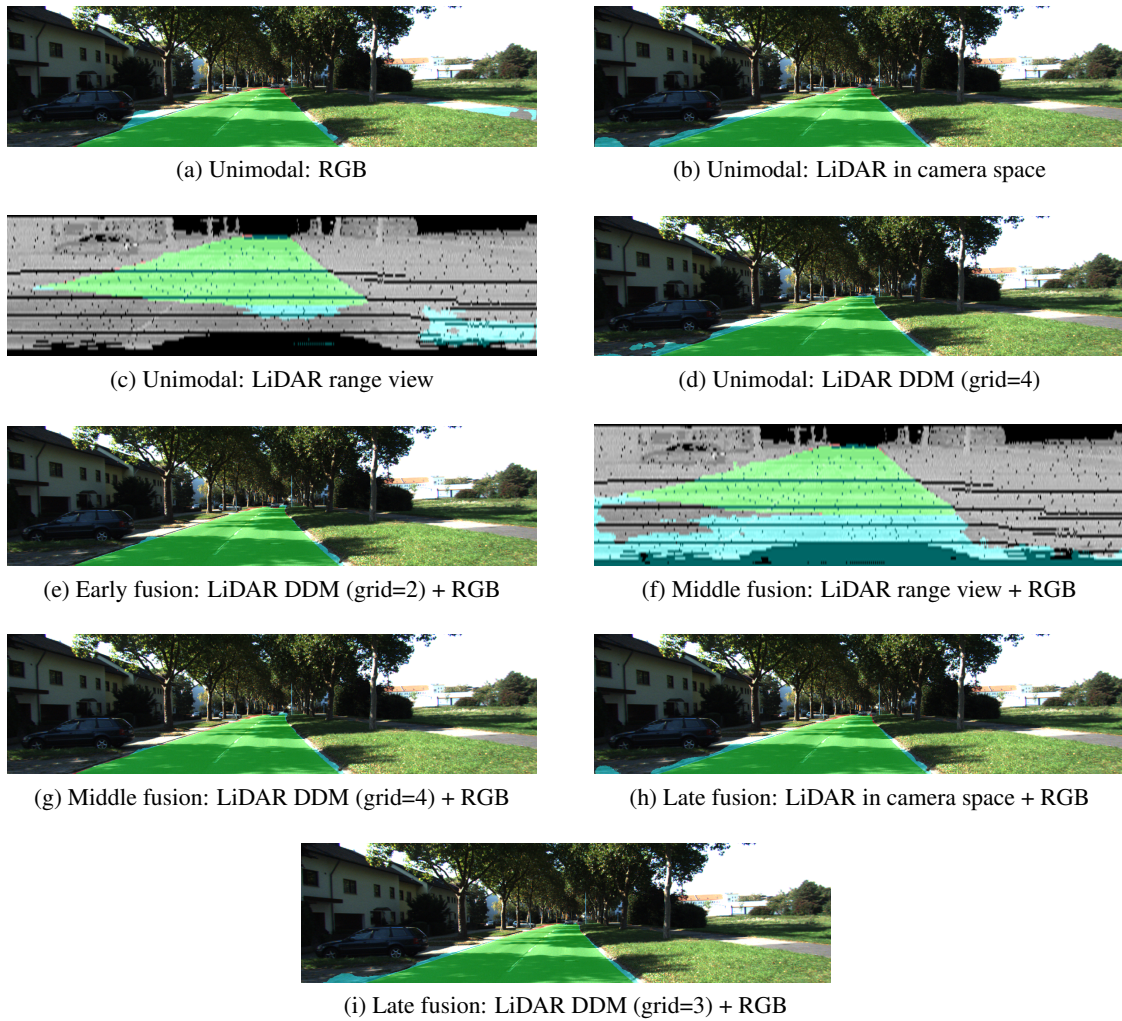


Figure 7.2: Results for the UM_ROAD category in the um_000074 sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

all road pixels based solely on RGB data was likely due to the shade and the position of the road on the right side of the image. The integration with the LiDAR depth values in the early fusion modality led to a significant enhancement in the detection of the road in that area. However, in this scenario, the LiDAR dense depth maps unimodal network provided the necessary information for an accurate estimation on the right side of the road. The middle fusion of the LiDAR DDM with the RGB image (Figure 7.3g) exhibited promising results in terms of the false positive rate.

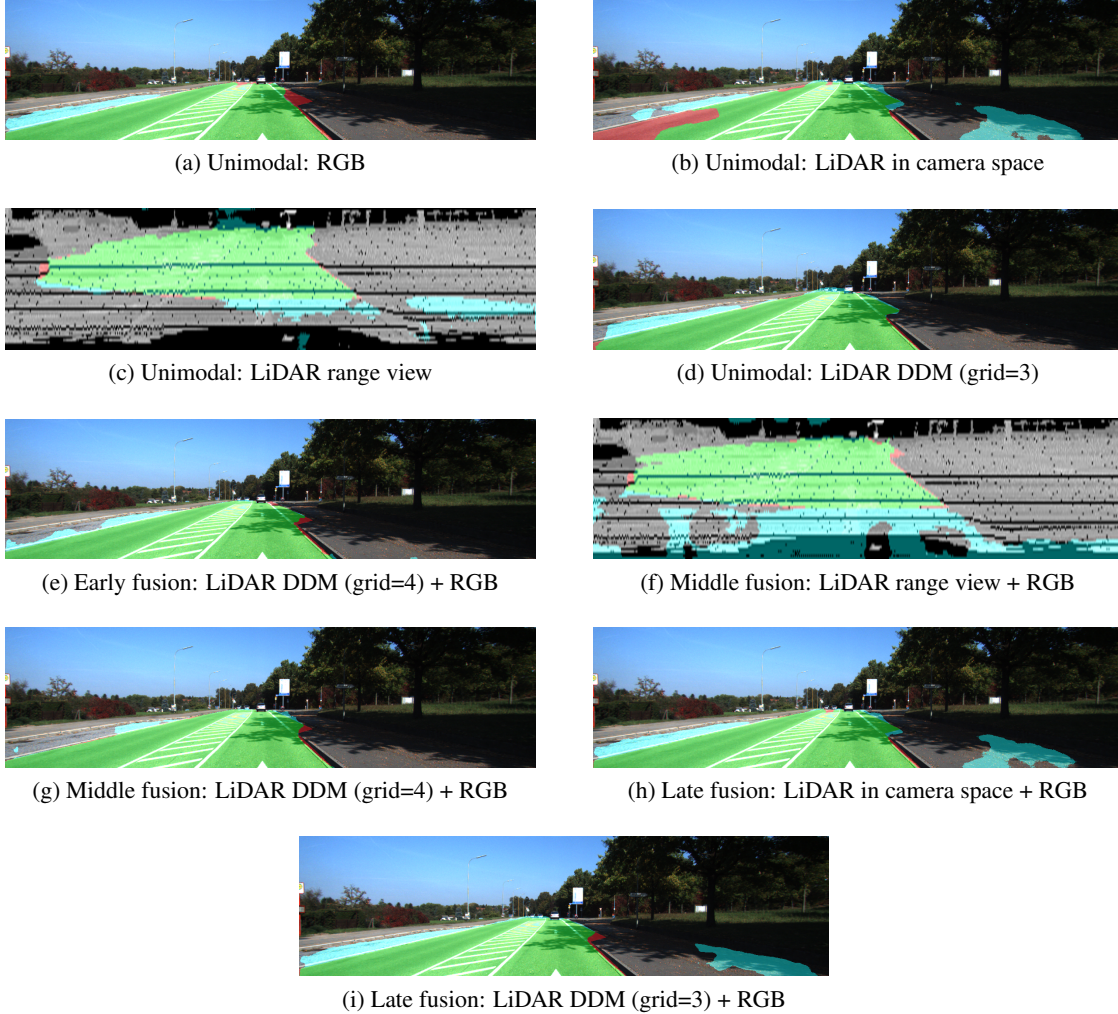
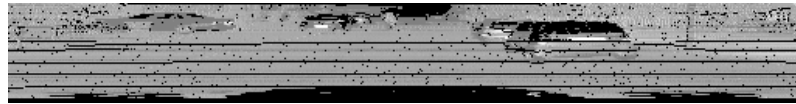


Figure 7.3: Results for the UMM_ROAD category in the umm_000088 sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

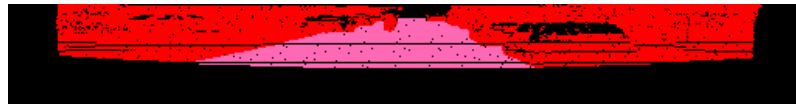
7.3.3 Urban Unmarked Road (UU_ROAD)

The results for the UU_ROAD category presented in Figure 7.5 showcase a crowded street with several parked cars and a cyclist approaching from the opposite direction. Once again, the models had some difficulty distinguishing between the road and the sidewalk on both sides of the street. These boundaries resulted in an almost uniformly red region indicating false negative predictions.

The unimodal LiDAR DDM (grid=4) (Figure 7.5d) network had some trouble correctly classifying the pixels around the cyclist. Since the only information in these maps is the depth values, the addition of the RGB data has improved the predictions in this area for both the middle and early fusions. The middle fusion of the LiDAR range view with the RGB images (Figure 7.5f) had a low rate of false negatives, but the number of false positives was considerably high. This phenomenon was due to labelling limitations of the dataset since only the camera points were annotated and the labels for the LiDAR points were not available. Consequently, the transformation of the label points to the range view representation did not cover the full extension of the data captured by the LiDAR. Therefore, the concatenation of the LiDAR range view samples with the resized RGB images led to most of the falsely classified points actually belonging to the road region. However, due to the lack of ground truth labels for these regions, they were incorrectly identified as false positives. Figure 7.4b depicts the label for the range view image presented in the results. Points near the sensor lack a valid label ('road' or 'non-road'), resulting in their classification as road false positives. However, in reality, those points are indeed part of the road, as evident in Figure 7.4a. Hence, when evaluating the results of the middle fusion in range view, this aspect should be considered.



(a) Reflectivity channel of the range view uu_000029 sample.



(b) Label in range view for the uu_000029 sample. Red points denote the "non-road" class; purple points refer to the "road" pixels; black points are the ones without a label.



(c) Camera image corresponding to the uu_000029 frame, resized to the range view size.

Figure 7.4: LiDAR and RGB data used for the middle fusion in range view. Example for the sample image uu_000029.

7.3.4 Bosch Road

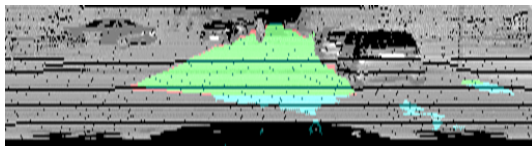
Both the LiDAR range view unimodal network and the middle fusion model demonstrated solid results for the road estimation task. In Figure 7.6, it is possible to observe that the unimodal network faced some challenges in accurately classifying the road pixels near the vegetation on the



(a) Unimodal: RGB



(b) Unimodal: LiDAR in camera space



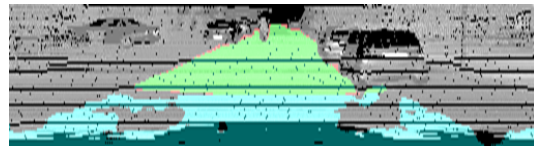
(c) Unimodal: LiDAR range view



(d) Unimodal: LiDAR DDM (grid=4)



(e) Early fusion: LiDAR DDM (grid=2) + RGB



(f) Middle fusion: LiDAR range view + RGB



(g) Middle fusion: LiDAR DDM (grid=4) + RGB



(h) Late fusion: LiDAR in camera space + RGB



(i) Late fusion: LiDAR DDM (grid=3) + RGB

Figure 7.5: Results for the UU_ROAD category in the uu_000029 sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

left side of the image. However, the middle fusion strategy improved this estimation in that area suggesting that the added dense colour information from the RGB images was beneficial in this particular region.

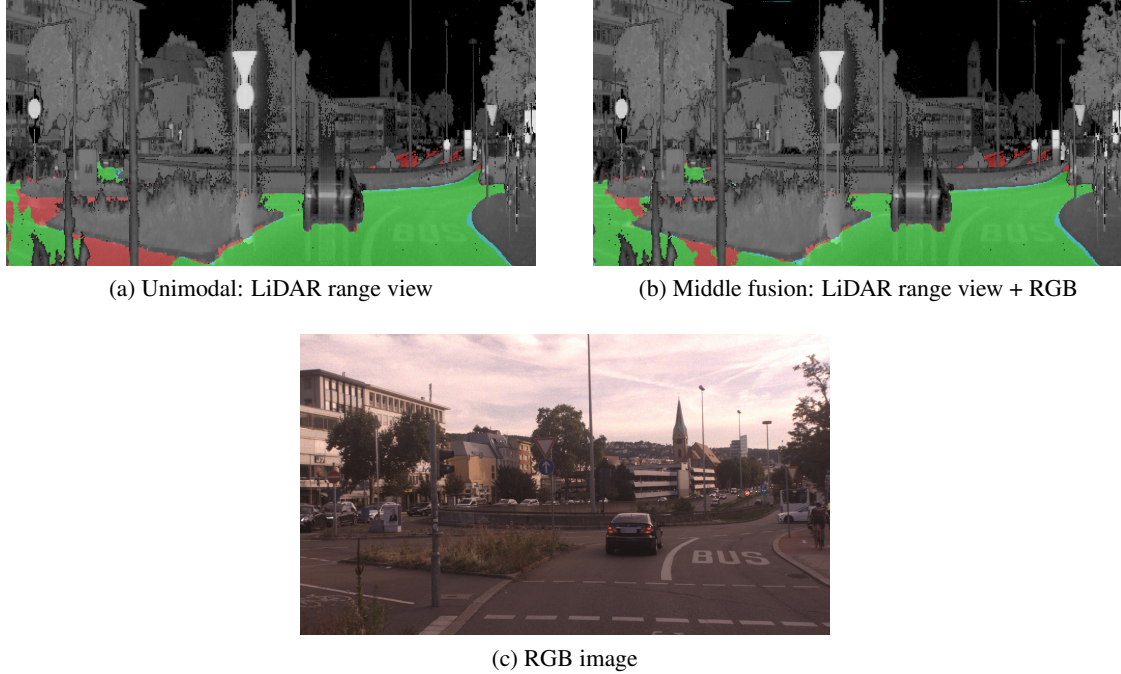


Figure 7.6: Results for road segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

Figure 7.7 showcases an example where the road area is heavily obstructed by pedestrians and other cars. In this situation, both networks encountered several challenges in distinguishing the road from the sidewalk. However, the unimodal network presented a lower rate of false positives, indicating a better understanding of the scene. On the other hand, the middle fusion model demonstrated a lower false negative rate around the cars and pedestrians, indicating that the inclusion of colour information improved the estimation. Similarly, the example in Figure 7.8, which also portrays a situation with many pedestrians, exhibited a high rate of false positives in the sidewalk area.

In Figure 7.9, cross-talk is clearly visible, and can influence the road estimation task in several ways. The presence of cross-talk introduces noise, distortions and artefacts that affect the segmentation. Figure 7.10 showcases two examples with a high level of interference, where the road estimation task was clearly affected by the artefacts introduced.

7.3.5 Urban Marked Ego-lane (UM_LANE)

The network for ego-lane estimation using RGB images solely (Figure 7.11a) presented several struggles, resulting in a high false negative rate. Probably due to the harsh light and shadows, the model was not able to uniformly identify the ego-lane, resulting in a poor classification. On the

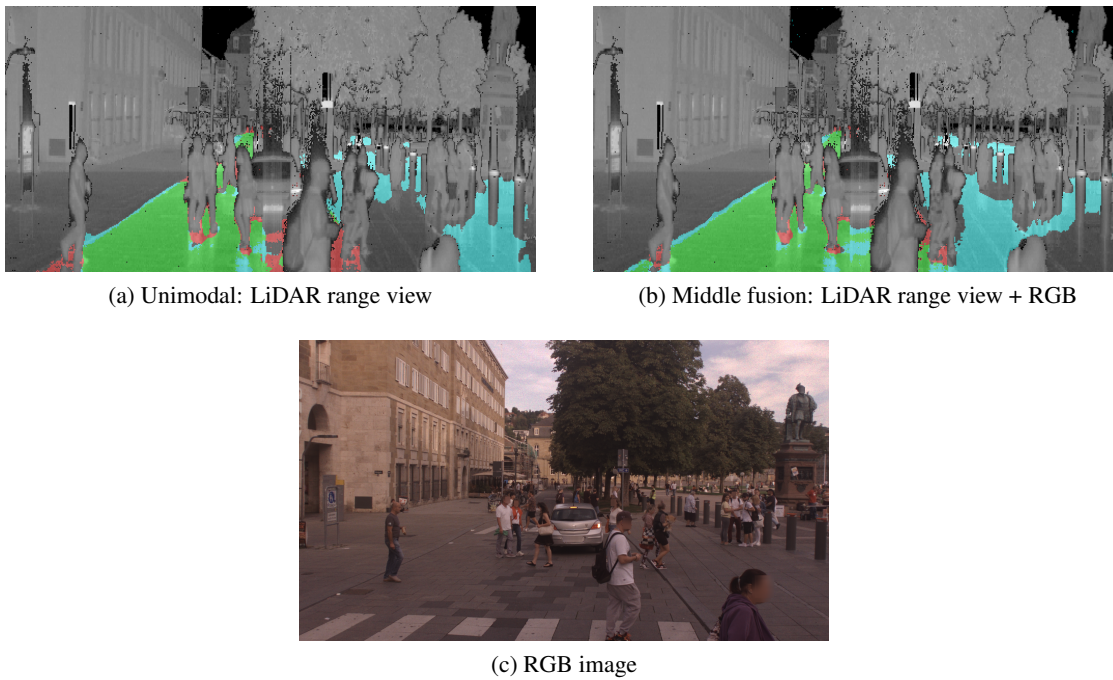


Figure 7.7: Results for road segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

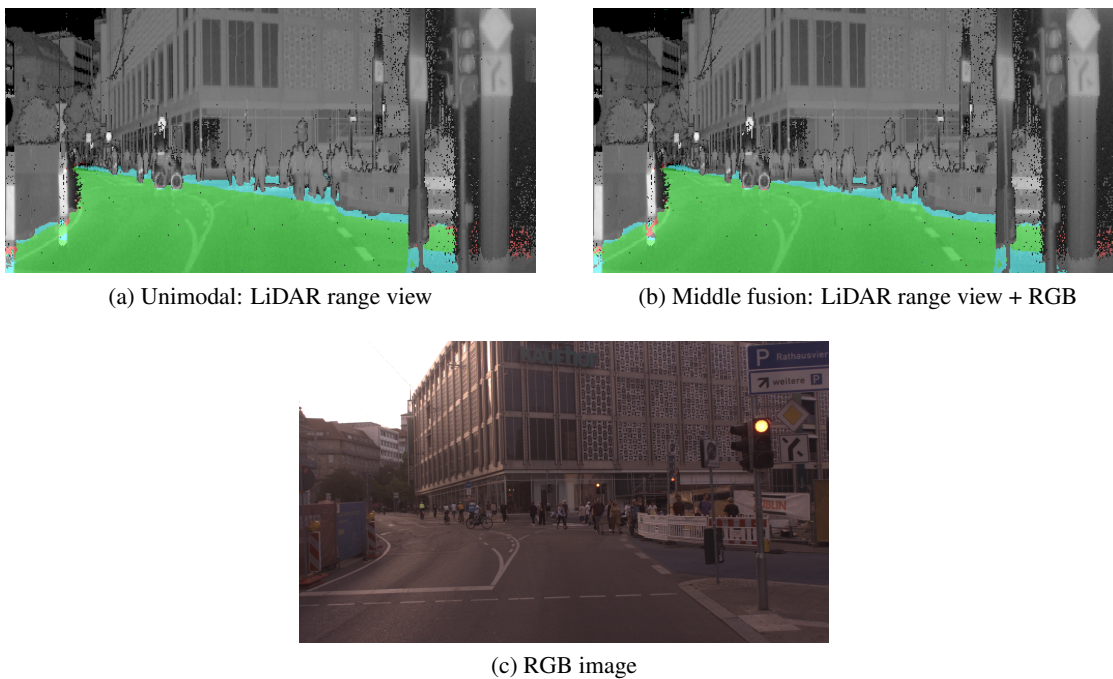


Figure 7.8: Results for road segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

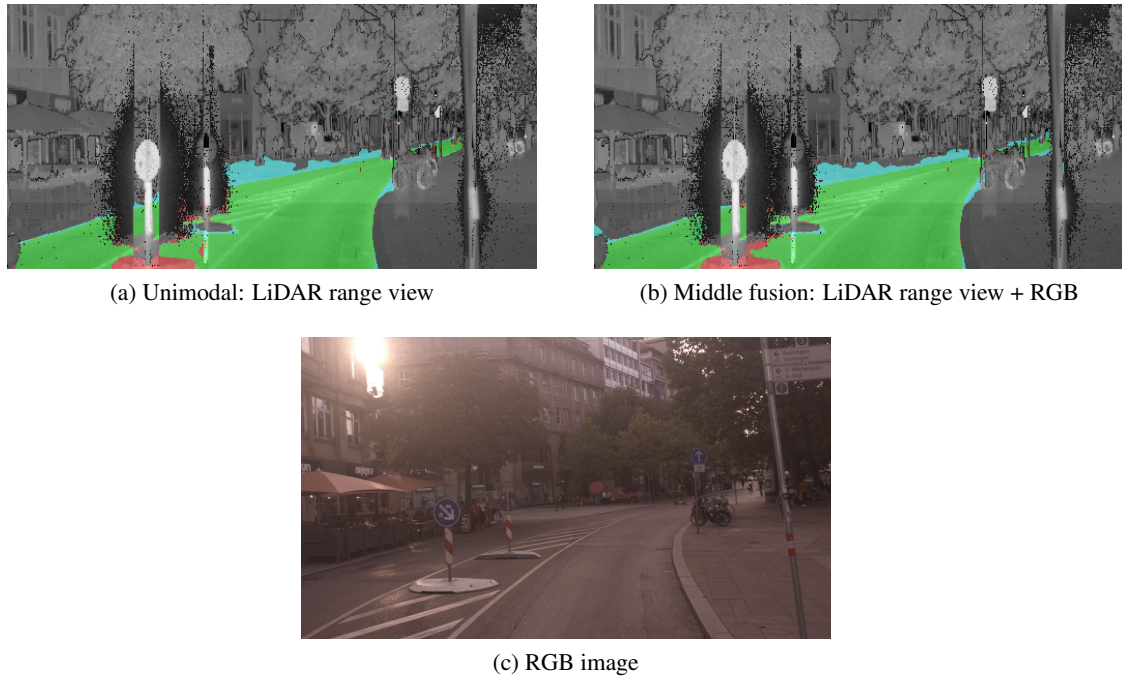


Figure 7.9: Results for road segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

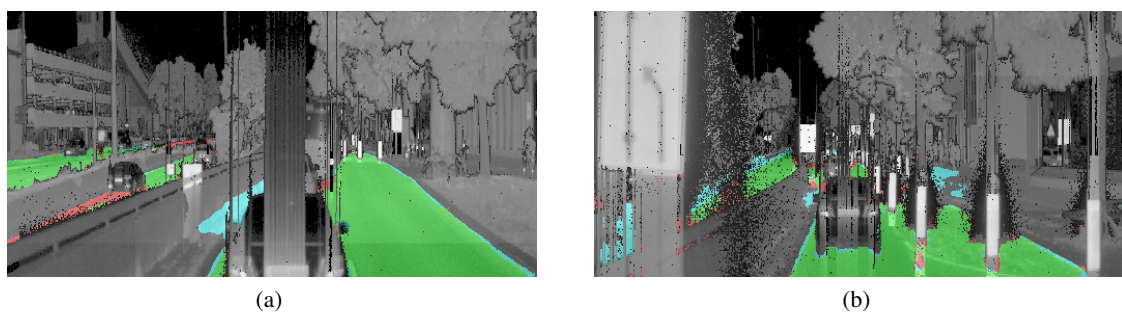


Figure 7.10: Example of samples with considerable cross-talk effects.

other hand, the middle fusion of the RGB images with the LiDAR dense depth maps (Figure 7.11g) exhibited promising results, with only small struggles in differentiating the right side of the road from the sidewalk. The use of LiDAR DDMs alone for the estimation (Figure 7.11d), particularly on the left side of the lane, yielded poorer results. The absence of colour or reflectivity information in these maps hindered the network’s ability to discern the white line that separates the lanes. The late fusion of the LiDAR dense depth map with a grid size of three and the RGB images (Figure 7.11i) also presented solid results. By using the colour information present in the RGB data, the estimation on the left side of the lane was enhanced, making effective use of the white line as a clear delineator. Additionally, the depth information contributed to the correct prediction of the full extent of the lane regardless of the illumination conditions.

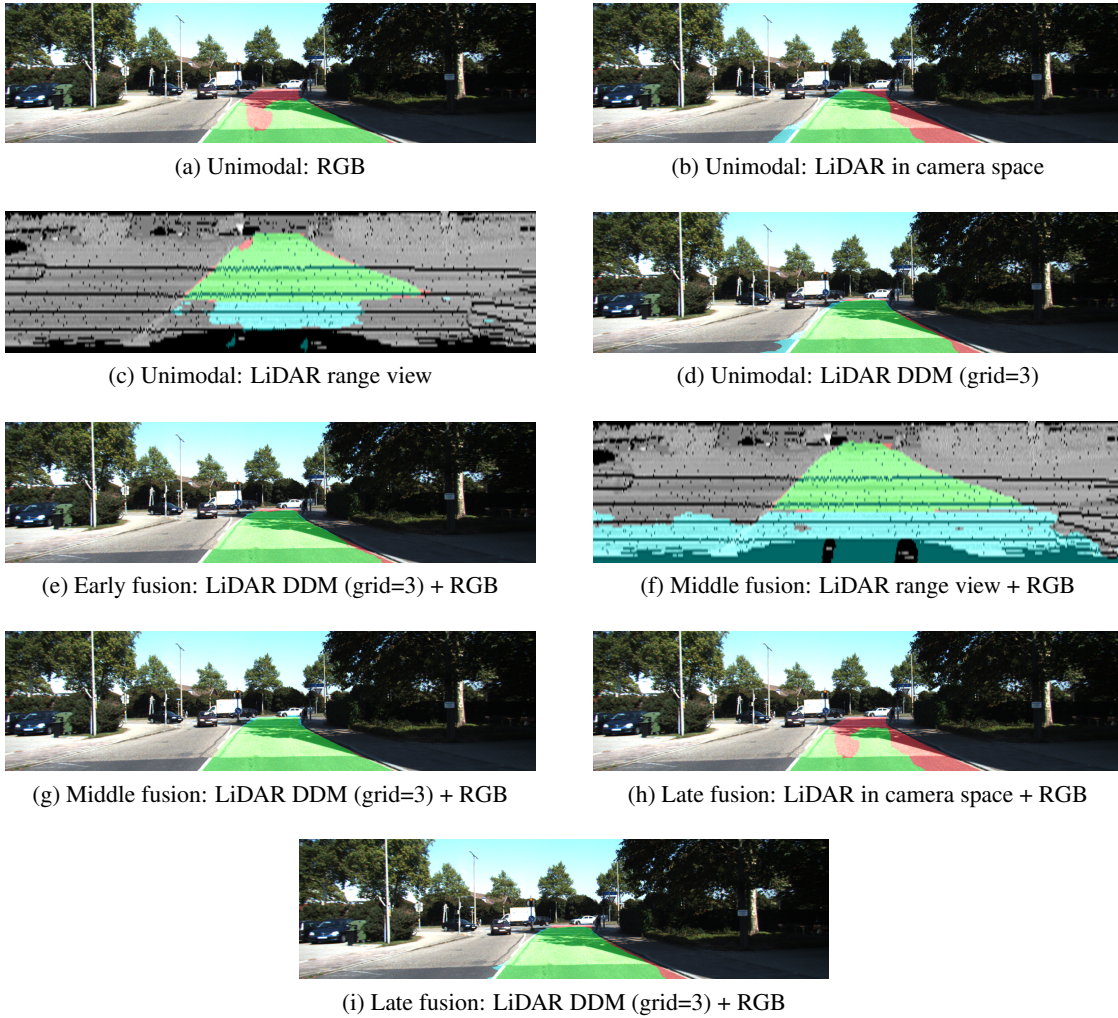


Figure 7.11: Results for the UM_LANE category in the um_000005 sample. Green points are the true positives, red points are the false negatives and blue points are the false positives.

7.3.6 Bosch Lane

The estimation of the lane markings in the Bosch dataset presented greater challenges compared to the road detection task. The slenderness and curvature of the lane lines added complexity to the estimation, introducing difficulties for the networks.

Figure 7.12 presents a situation where the false negative rate was relatively high. Both the unimodal LiDAR network and middle fusion model struggled to accurately classify the majority of the lane pixels due to the dense concentration of lane markings. This scenario portrays a visually complex situation with lane markings in every direction, which may have caused confusion for the network and prevented a solid estimation. It is evident that most of the lane marking points correctly classified were part of longitudinal lines, which could indicate a bias towards vertical lanes. This possible bias is also highlighted in Figure 7.13, where the network failed to detect the lane markings in a horizontal position. This could be attributed to the dominance of longitudinal lane markings in the training set.

Figures 7.14 and 7.15 showcase two situations where the estimation was highly successful. The unimodal network using the LiDAR range view projection effectively distinguished the lane markings from the road, resulting in a low number of false positive and false negative points. The middle fusion network also presented an effective classification capability despite the slightly higher number of false negative points in Figure 7.15b. These situations were both unobstructed, allowing for clear visibility of the lane markings.

In Figure 7.16, our networks demonstrated reliable lane estimation even in the presence of curved markings, highlighting the robustness of our method for non-straight lane lines. However, in regions farther away from the sensor, the false negative rate increased, and the models struggled to identify lane points.

Finally, in Figure 7.17 the sample does not include any points labelled as lane markings, but it is important to highlight the networks' ability to segment the road area without the lane lines as a clear delineator.

7.4 Final Discussion

The early fusion technique combined the LiDAR depth information with RGB images before the model could process it. This fusion strategy required an alignment between the LiDAR points and the RGB pixels and involved low calculation requirements. An improvement in relation to the unimodal networks was definitely noted in this approach, and it is possible to conclude that the network was capable of learning relations between the two types of data from the raw characteristics.

The middle fusion strategy achieved a superior performance across all categories in the KITTI-ROAD dataset. Our work introduced a novel approach to middle fusion without calibration parameters. This methodology did not have a pixel-to-pixel correspondence between the resized RGB images and the LiDAR range view projections, in opposition to the majority of works done so far.

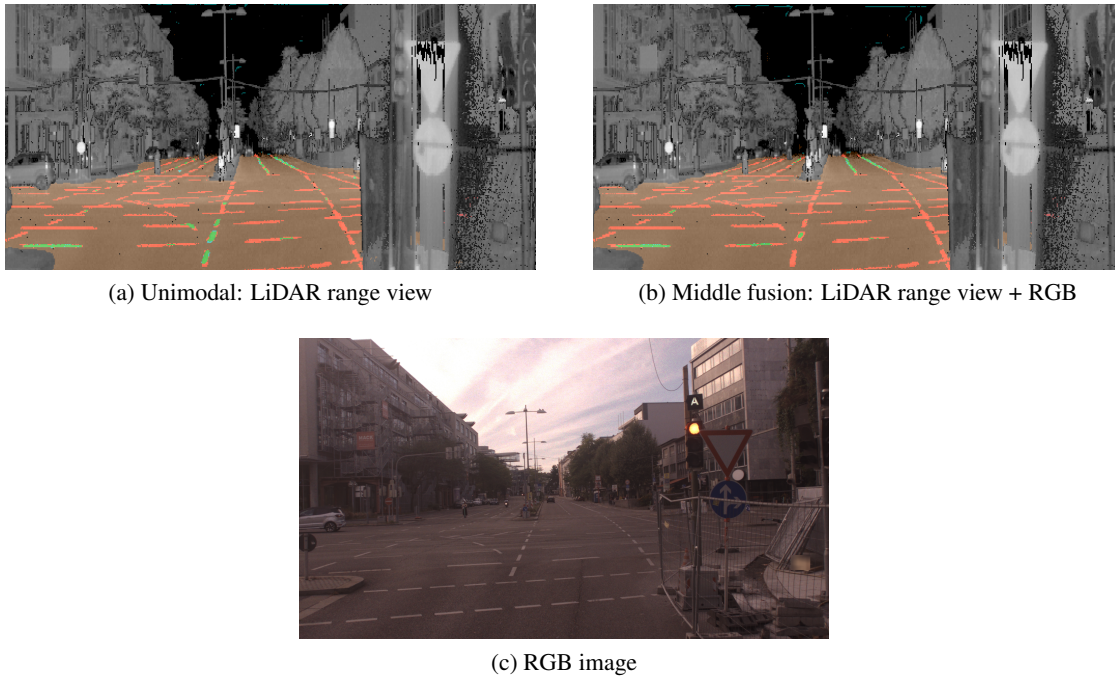


Figure 7.12: Results for lane markings segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives. Dark orange points are the road estimation results.

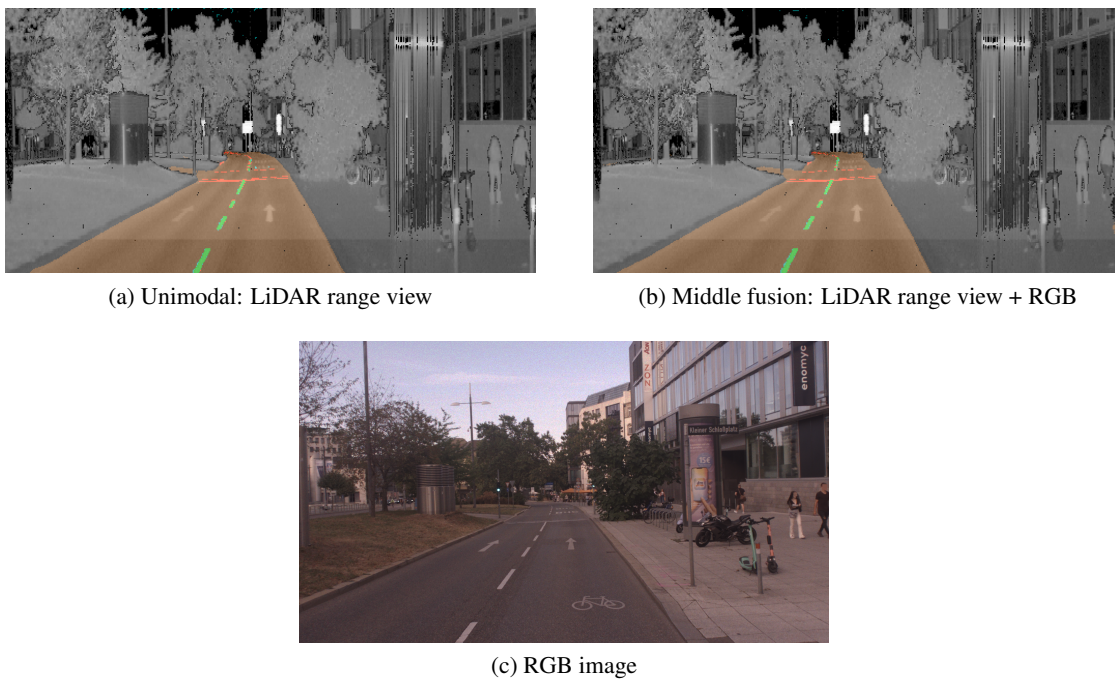


Figure 7.13: Results for lane markings segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives. Dark orange points are the road estimation results.

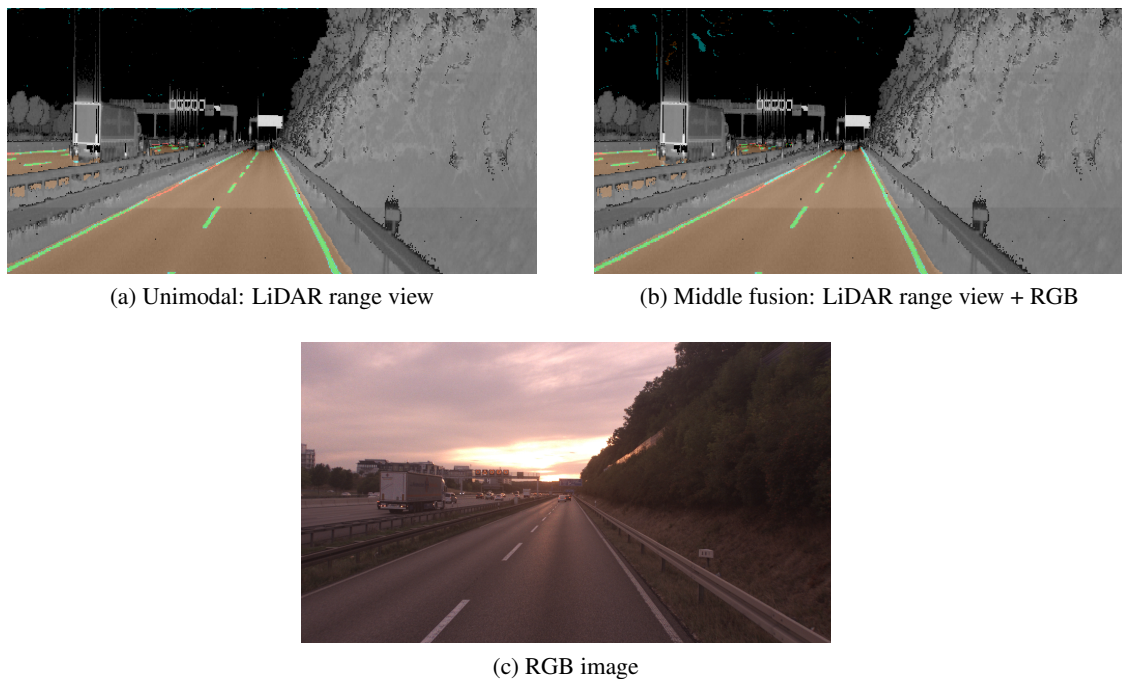


Figure 7.14: Results for lane markings segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives. Dark orange points are the road estimation results.

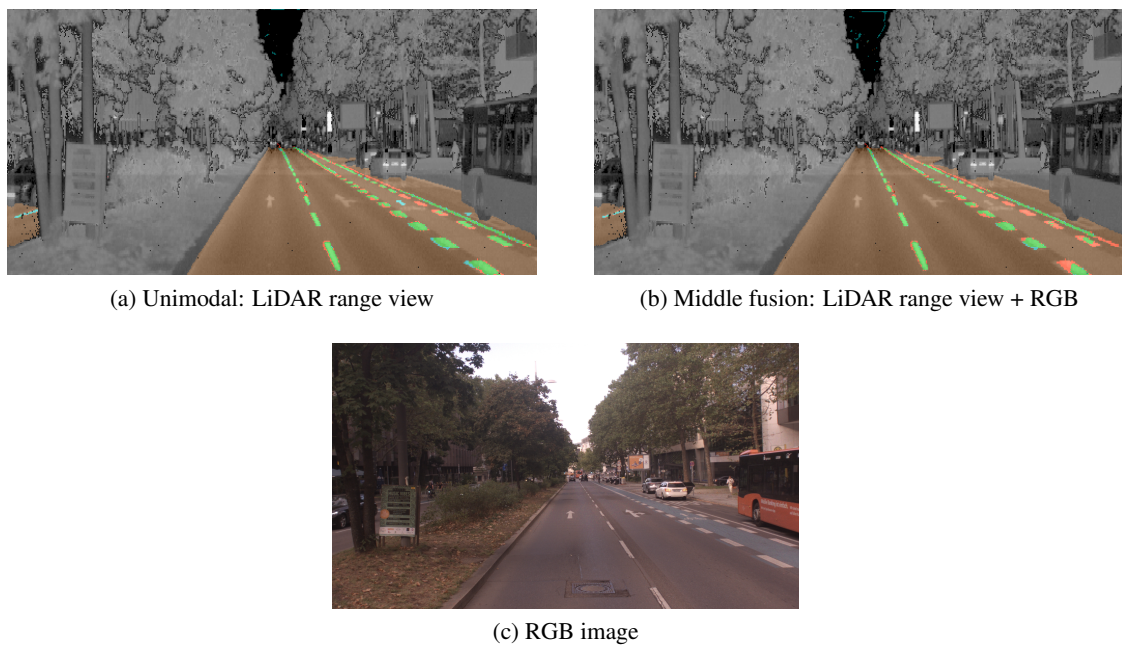


Figure 7.15: Results for lane markings segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives. Dark orange points are the road estimation results.

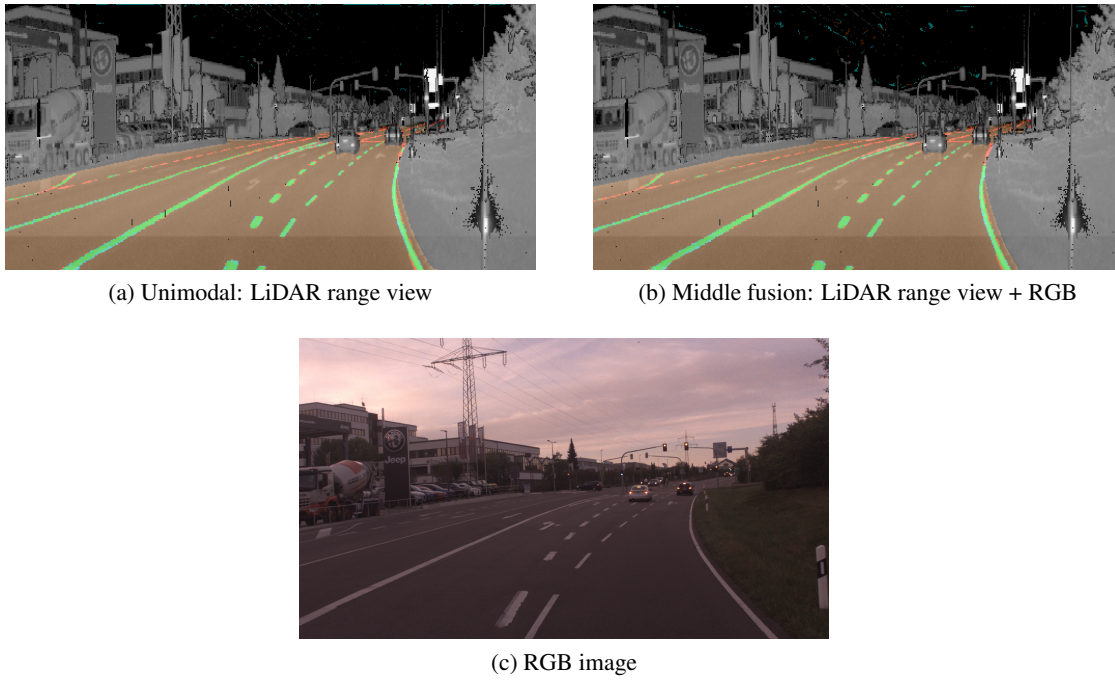


Figure 7.16: Results for lane markings segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives. Dark orange points are the road estimation results.

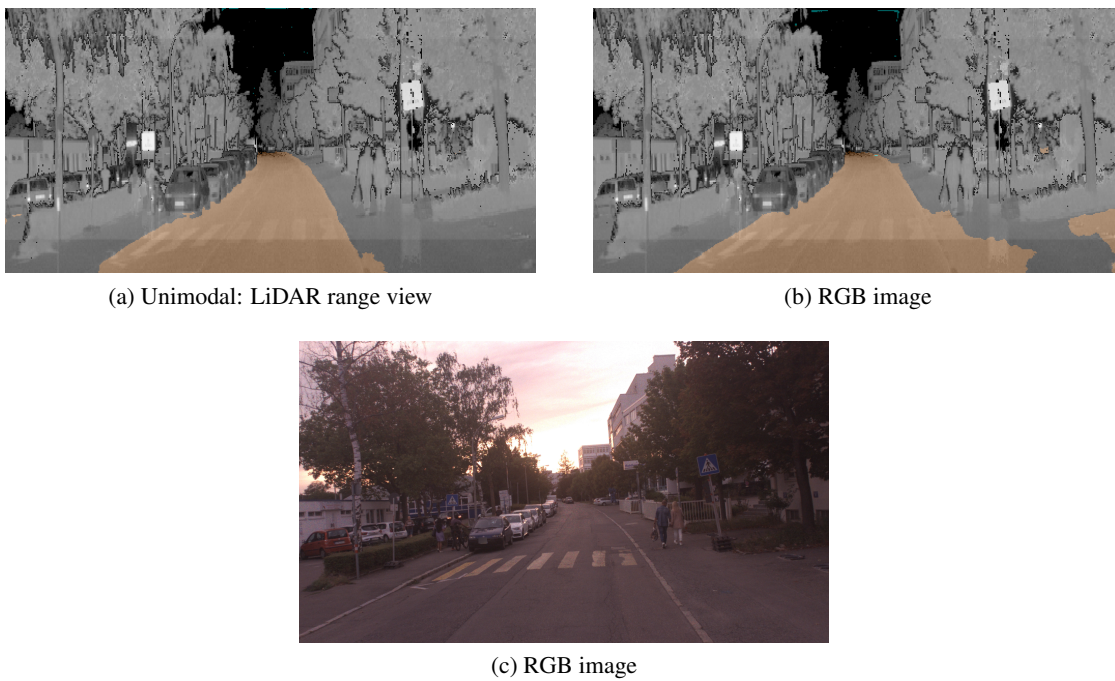


Figure 7.17: Results for lane markings segmentation in the Bosch dataset for an example test sample. Green points are the true positives, red points are the false negatives and blue points are the false positives. Dark orange points are the road estimation results.

The network was capable of learning to align the features extracted from both modalities during training. Although the pixel values of the RGB images did not directly correspond to the points in the range view domain, they still contained valuable semantic information capable of aiding the model in understanding the scene. The context modules for the LiDAR and RGB information were able to capture global information for the respective modality. These cues provided additional information beyond the typical pixel-to-pixel correspondence. This contextual information was leveraged by the network and used to refine the predictions and perform a more accurate road and ego-lane estimation. Despite the lack of explicit calibration parameters in this approach, the network was capable of learning to calibrate itself from the information provided by the context modules. During training, it discovered and adapted the relationships between the LiDAR data and RGB images based on the available information. This strategy also provided a good compromise between performance and inference speed. Beyond KITTI-ROAD, the results on Bosch data support these conclusions about the proposed middle fusion approach. However, beyond sensor calibration, Bosch data also lacks sensor hardware calibration, which may justify why the lane marking segmentation results are slightly inferior to those with just LiDAR. Nevertheless, such a small decrease in performance may just be an acceptable trade-off for many purposes as it allows us, using the proposed middle fusion methodology, to forego entirely the calibration process.

Late fusion performed two separate estimations for each of the modalities. As a result, two networks had to be trained, increasing the computational cost and storage needs compared to the other fusion techniques. This method has the advantage of being flexible and allows for the replacement of the sensing modalities without interfering with the other ones. However, it achieved the overall worst performance of the three. In order to improve the results of this fusion methodology, other networks for combining the results of the predictions could be tested. Our network only had two fully connected layers, and maybe a deeper network would provide better results.

Although multi-sensor configurations have clearly achieved better results, they also introduced more computational complexity due to the different nature of LiDAR and camera data. Upsampling the LiDAR depth data reduced the gap between the sparse LiDAR points and the dense RGB pixels, allowing the fusion of modalities to benefit from their distinct characteristics. However, the computational cost of the upsampling operation was large.

Chapter 8

Conclusion and Future Work

With the advances in Artificial Intelligence and subsequent growth of Deep Learning technologies, the research on accurate and robust perception systems for Autonomous Driving had an increased interest within the scientific community. In this way, road and lane estimation plays a crucial role in the perception of the surrounding environment and constitutes a major target of extensive research. The fusion of RGB and LiDAR information has emerged as a topic of discussion due to their different characteristics and complementarity. While the LiDAR point clouds provide precise depth and reflectivity information independent of lighting conditions, the RGB data contains rich colour and texture information about the surrounding environment. However, determining the best way to fuse this data is not a trivial task.

This study aimed to address this challenge through the use of SalsaNext for semantic segmentation and three fusion strategies in the KITTI-ROAD dataset and a Bosch proprietary dataset. The early fusion methodology combined the raw LiDAR and RGB information before feeding the result to the SalsaNext network; in the late fusion model, the predictions from two separate branches were combined to produce a single output at the decision level; the middle fusion strategy merged the camera and LiDAR data at the context module level.

After the experiments conducted and the detailed analysis of the results, we were able to conclude that the middle fusion strategy is the most promising one. In addition to having achieved the best quantitative results across all categories in the KITTI-ROAD dataset, it also addressed the limitations of the individual modalities, such as the challenges faced by the models relying solely on RGB data in harsh/low light scenarios and the inability of the LiDAR dense depth maps in distinguishing the road from the cars at a long distance. The exploration of the complementarity of these two modalities demonstrated an increased accuracy and robustness in the estimation.

The results in the Bosch dataset demonstrated that both the unimodal LiDAR network and the middle fusion strategy achieved promising results for road estimation. The fusion of RGB information enhanced the estimation accuracy in specific areas, while the LiDAR-only network showcased more accurate detections on the sidewalk region. Additionally, the cross-talk effects on the LiDAR data introduced several challenges for a reliable estimation. Regarding the lane marking estimation, the performance of the networks was considerably lower. Further attention

and optimization are required to enhance the lane estimation results since the increased difficulty of this task posed a significant challenge for a reliable estimation.

Traditional fusion approaches rely on the calibration parameters between the different modalities. However, this calibration can be a challenging and time-consuming task, reducing the capability of the systems to perform in real time. Also, the calculation of the several extrinsic parameters and coordinates transformations between the sensors can only be done with resource to specialized equipment, and expert knowledge. While early and late fusions are dependent on the calibration between the sensors, our middle fusion approach without calibration parameters has alleviated the burden of complex calibrations making the system more accessible. Moreover, without the dependency on calibration data, our model becomes more flexible and can accommodate different types of sensors with little need for modifications.

8.1 Future Work

The work conducted during this dissertation has contributed to the development of multimodal strategies and provided valuable insights for future studies. Nevertheless, there is still plenty of work to be done. For example, the application of domain adaptation strategies could be considered. Given the limited number of samples in the KITTI-ROAD dataset and the lack of challenging scenarios, it does not represent the diversity of real-world driving conditions. By matching the distribution of the training data with the target domain, domain adaptation methods can help close this gap and improve the model's ability to generalize to unseen situations.

Furthermore, other data augmentation techniques could be employed to enhance the robustness of our road and lane estimation systems. We could also test the generalization capabilities of our approaches by evaluating their performance on more datasets from different domains. This could help to assess their adaptability to distinct road conditions and driving styles.

It would also be interesting to analyse the behaviour of our middle fusion network without calibration parameters. The development of a comprehensive evaluation framework to assess the calibration accuracy of the model would provide further insights into the reliability of this mechanism. For the fusion networks that rely on calibration parameters, data augmentation techniques such as sensor misalignments and different sensor setups could improve the generalization capability of the models.

Improving the estimation of the road in the Bosch dataset would require addressing the cross-talk effects. By implementing some pre- or post-processing techniques, the artefacts introduced could be mitigated and significantly improve the results. The lane marking segmentation holds a great potential, and the results could be improved through techniques like artefact removal or noise reduction. Some post-processing techniques could also be employed, and hyperparameter tuning could potentially improve the performance of the networks.

A major challenge encountered during this work was the limited access to datasets with semantic annotations for lanes with both RGB and LiDAR information. Given this limitation, it would have been interesting to explore self-supervised or unsupervised techniques, which could

reduce the reliance on labelled data and present alternative solutions to address this issue. Another approach would be the use of synthetic data for the estimation.

References

- [1] P. Thomas, A. Morris, R. Talbot, and H. Fagerlind, “Identifying the causes of road crashes in Europe,” *Annals of Advances in Automotive Medicine*, vol. 57, pp. 13–22, Sep. 2013, ISSN: 1943-2461.
- [2] D. Feng, C. Haase-Schuetz, L. Rosenbaum, *et al.*, “Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 1341–1360, 2019.
- [3] R. Harrington, C. Senatore, J. Scanlon, and R. Yee, “The Role of Infrastructure in an Automated Vehicle Future,” *Bridge*, vol. 40, pp. 48–55, Jun. 2018.
- [4] J. Tang, S. Li, and P. Liu, “A review of lane detection methods based on deep learning,” *Pattern Recognition*, vol. 111, p. 107 623, Mar. 2021, ISSN: 00313203. DOI: [10.1016/j.patcog.2020.107623](https://doi.org/10.1016/j.patcog.2020.107623).
- [5] H. Abualsaud, S. Liu, D. B. Lu, K. Situ, A. Rangesh, and M. M. Trivedi, “LaneAF: Robust Multi-Lane Detection With Affinity Fields,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7477–7484, Oct. 2021, ISSN: 2377-3766. DOI: [10.1109/LRA.2021.3098066](https://doi.org/10.1109/LRA.2021.3098066).
- [6] D.-H. Paek, S.-H. Kong, and K. T. Wijaya, “K-Lane: Lidar Lane Dataset and Benchmark for Urban Roads and Highways,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 4449–4458, ISBN: 978-1-66548-739-9. DOI: [10.1109/CVPRW56347.2022.00491](https://doi.org/10.1109/CVPRW56347.2022.00491).
- [7] Y.-T. Cheng, A. Patel, C. Wen, D. Bullock, and A. Habib, “Intensity Thresholding and Deep Learning Based Lane Marking Extraction and Lane Width Estimation from Mobile Light Detection and Ranging (LiDAR) Point Clouds,” *Remote Sensing*, vol. 12, no. 9, p. 1379, Jan. 2020, ISSN: 2072-4292. DOI: [10.3390/rs12091379](https://doi.org/10.3390/rs12091379).
- [8] F. Yu, H. Chen, X. Wang, *et al.*, “Bdd100k: A diverse driving dataset for heterogeneous multitask learning,” Jun. 2020, pp. 2633–2642. DOI: [10.1109/CVPR42600.2020.00271](https://doi.org/10.1109/CVPR42600.2020.00271).
- [9] M. Bai, G. Mattyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun, “Deep multi-sensor lane detection,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain: IEEE Press, 2018, pp. 3102–3109. DOI: [10.1109/IROS.2018.8594388](https://doi.org/10.1109/IROS.2018.8594388). [Online]. Available: <https://doi.org/10.1109/IROS.2018.8594388>.

- [10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–44, May 2015. DOI: [10.1038/nature14539](https://doi.org/10.1038/nature14539).
- [11] S. J. Russell, P. Norvig, and E. Davis, *Artificial Intelligence: A Modern Approach* (Prentice Hall Series in Artificial Intelligence), 3rd ed. Upper Saddle River: Prentice Hall, 2010, ISBN: 978-0-13-604259-4.
- [12] *Dark side of neural networks explained [2023]*, <https://research.aimultiple.com/how-neural-networks-work/>.
- [13] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning*. Dec. 2020.
- [14] A. Olgac and B. Karlik, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," *International Journal of Artificial Intelligence And Expert Systems*, vol. 1, pp. 111–122, Feb. 2011.
- [15] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," p. 8,
- [16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, ISSN: 1476-4687. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0).
- [17] *Deep Learning*, <https://www.deeplearningbook.org/contents/optimization.html>.
- [18] A. Mustapha, L. Mohamed, and K. Ali, "An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field," in *Smart Applications and Data Analysis*, M. Hamlich, L. Bellatreche, A. Mondal, and C. Ordonez, Eds., Cham: Springer International Publishing, 2020, pp. 349–359, ISBN: 978-3-030-45183-7.
- [19] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,"
- [20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1412.6980>.
- [21] K. O'Shea and R. Nash, *An Introduction to Convolutional Neural Networks*, Dec. 2015. arXiv: [1511.08458](https://arxiv.org/abs/1511.08458) [cs].
- [22] *CS231n Convolutional Neural Networks for Visual Recognition*. [Online]. Available: <https://cs231n.github.io/convolutional-networks/>.
- [23] *Deep Learning*, <https://www.deeplearningbook.org/contents/regularization.html>.
- [24] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jul. 2020, pp. 237–242. DOI: [10.1109/IWSSIP48289.2020.9145130](https://doi.org/10.1109/IWSSIP48289.2020.9145130).

- [25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010, ISSN: 0920-5691, 1573-1405. DOI: [10.1007/s11263-009-0275-4](https://doi.org/10.1007/s11263-009-0275-4).
- [26] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. Wellington, *3D Point Cloud Processing and Learning for Autonomous Driving*, Mar. 2020. arXiv: [2003.00601](https://arxiv.org/abs/2003.00601) [cs, eess].
- [27] *Papers with Code - KITTI Road Dataset*, <https://paperswithcode.com/dataset/kitti-road>. [Online]. Available: https://www.cvlibs.net/datasets/kitti/eval_road.php.
- [28] *TuSimple Competitions for CVPR2017*, TuSimple, Nov. 2022.
- [29] *Papers with Code - CULane Dataset*, <https://paperswithcode.com/dataset/culane>.
- [30] H. Xu, S. Wang, X. Cai, W. Zhang, X. Liang, and Z. Li, “CurveLane-NAS: Unifying Lane-Sensitive Architecture Search and Adaptive Point Blending,” in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., vol. 12360, Cham: Springer International Publishing, 2020, pp. 689–704, ISBN: 978-3-030-58554-9 978-3-030-58555-6. DOI: [10.1007/978-3-030-58555-6_41](https://doi.org/10.1007/978-3-030-58555-6_41).
- [31] K. Behrendt and R. Soussan, “Unsupervised Labeled Lane Markers Using Maps,” in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 832–839, ISBN: 978-1-72815-023-9. DOI: [10.1109/ICCVW.2019.00111](https://doi.org/10.1109/ICCVW.2019.00111).
- [32] X. Zhang, Z. Li, X. Gao, D. Jin, and J. Li, “Channel Attention in LiDAR-camera Fusion for Lane Line Segmentation,” *Pattern Recognition*, vol. 118, p. 108 020, Oct. 2021, ISSN: 0031-3203. DOI: [10.1016/j.patcog.2021.108020](https://doi.org/10.1016/j.patcog.2021.108020).
- [33] R. Yin, Y. Cheng, H. Wu, Y. Song, B. Yu, and R. Niu, “FusionLane: Multi-Sensor Fusion for Lane Marking Semantic Segmentation Using Deep Neural Networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1543–1553, Feb. 2022, ISSN: 1558-0016. DOI: [10.1109/TITS.2020.3030767](https://doi.org/10.1109/TITS.2020.3030767).
- [34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788. DOI: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [35] M. Najibi, M. Rastegari, and L. S. Davis, “G-CNN: An Iterative Grid Based Object Detector,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2369–2377. DOI: [10.1109/CVPR.2016.260](https://doi.org/10.1109/CVPR.2016.260).
- [36] R. Girshick, “Fast R-CNN,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1440–1448. DOI: [10.1109/ICCV.2015.169](https://doi.org/10.1109/ICCV.2015.169).
- [37] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask R-CNN*, Jan. 2018. arXiv: [1703.06870](https://arxiv.org/abs/1703.06870) [cs].

- [38] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.
- [39] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, “Spatial as Deep: Spatial CNN for Traffic Scene Understanding,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, ISSN: 2374-3468. DOI: [10.1609/aaai.v32i1.12301](https://doi.org/10.1609/aaai.v32i1.12301).
- [40] D. K. Dewangan, S. P. Sahu, B. Sairam, and A. Agrawal, “VLDNet: Vision-based lane region detection network for intelligent vehicle system using semantic segmentation,” *Computing*, vol. 103, no. 12, pp. 2867–2892, Dec. 2021, ISSN: 0010-485X, 1436-5057. DOI: [10.1007/s00607-021-00974-2](https://doi.org/10.1007/s00607-021-00974-2).
- [41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [42] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2015, pp. 234–241, ISBN: 978-3-319-24574-4. DOI: [10.1007/978-3-319-24574-4_28](https://doi.org/10.1007/978-3-319-24574-4_28).
- [43] Y. Lyu, L. Bai, and X. Huang, “Real-Time Road Segmentation Using LiDAR Data Processing on an FPGA,” in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5. DOI: [10.1109/ISCAS.2018.8351244](https://doi.org/10.1109/ISCAS.2018.8351244).
- [44] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, *SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving*, Jul. 2020. arXiv: [2003.03653 \[cs\]](https://arxiv.org/abs/2003.03653).
- [45] E. E. Aksoy, S. Baci, and S. Cavdar, “SalsaNet: Fast Road and Vehicle Segmentation in LiDAR Point Clouds for Autonomous Driving,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*, Oct. 2020, pp. 926–932. DOI: [10.1109/IV47402.2020.9304694](https://doi.org/10.1109/IV47402.2020.9304694).
- [46] D. Neven, B. Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, “Towards end-to-end lane detection: An instance segmentation approach,” Jun. 2018, pp. 286–291. DOI: [10.1109/IVS.2018.8500547](https://doi.org/10.1109/IVS.2018.8500547).
- [47] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, “Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA: IEEE, Jun. 2021, pp. 294–302, ISBN: 978-1-66544-509-2. DOI: [10.1109/CVPR46437.2021.00036](https://doi.org/10.1109/CVPR46437.2021.00036).

- [48] L. Liu, X. Chen, S. Zhu, and P. Tan, “CondLaneNet: A Top-to-down Lane Detection Framework Based on Conditional Convolution,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada: IEEE, Oct. 2021, pp. 3753–3762, ISBN: 978-1-66542-812-5. DOI: [10.1109/ICCV48922.2021.00375](https://doi.org/10.1109/ICCV48922.2021.00375).
- [49] Z. Qin, H. Wang, and X. Li, *Ultra Fast Structure-aware Deep Lane Detection*, Aug. 2020. arXiv: [2004.11757 \[cs\]](https://arxiv.org/abs/2004.11757).
- [50] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. D. Souza, and T. Oliveira-Santos, “Polylanenet: Lane estimation via deep polynomial regression,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, Los Alamitos, CA, USA: IEEE Computer Society, Jan. 2021, pp. 6150–6156. DOI: [10.1109/ICPR48806.2021.9412265](https://doi.org/10.1109/ICPR48806.2021.9412265). [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/ICPR48806.2021.9412265>.
- [51] R. Liu, Z. Yuan, T. Liu, and Z. Xiong, “End-to-end Lane Shape Prediction with Transformers,” in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA: IEEE, Jan. 2021, pp. 3693–3701, ISBN: 978-1-66540-477-8. DOI: [10.1109/WACV48630.2021.00374](https://doi.org/10.1109/WACV48630.2021.00374).
- [52] A. Xiao, J. Huang, D. Guan, K. Cui, S. Lu, and L. Shao, “Polarmix: A general data augmentation technique for lidar point clouds,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 11 035–11 048.
- [53] T. Zheng, Y. Huang, Y. Liu, *et al.*, “CLRNet: Cross Layer Refinement Network for Lane Detection,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 888–897, ISBN: 978-1-66546-946-3. DOI: [10.1109/CVPR52688.2022.00097](https://doi.org/10.1109/CVPR52688.2022.00097).
- [54] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 936–944, ISBN: 978-1-5386-0457-1. DOI: [10.1109/CVPR.2017.106](https://doi.org/10.1109/CVPR.2017.106).
- [55] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, “Deep Layer Aggregation,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 2403–2412, ISBN: 978-1-5386-6420-9. DOI: [10.1109/CVPR.2018.00255](https://doi.org/10.1109/CVPR.2018.00255).
- [56] J. Fritsch, T. Kuhn, and A. Geiger, “A new performance measure and evaluation benchmark for road detection algorithms,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, The Hague, Netherlands: IEEE, Oct. 2013, pp. 1693–1700, ISBN: 978-1-4799-2914-6. DOI: [10.1109/ITSC.2013.6728473](https://doi.org/10.1109/ITSC.2013.6728473).

- [57] H. Wang, R. Fan, P. Cai, and M. Liu, “SNE-RoadSeg+: Rethinking Depth-Normal Translation and Deep Supervision for Freespace Detection,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 1140–1145. DOI: [10.1109/IROS51168.2021.9636723](https://doi.org/10.1109/IROS51168.2021.9636723).
- [58] R. Fan, H. Wang, P. Cai, and M. Liu, “SNE-RoadSeg: Incorporating Surface Normal Information into Semantic Segmentation for Accurate Freespace Detection,”
- [59] R. Fan and M. Liu, “Road Damage Detection Based on Unsupervised Disparity Map Segmentation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4906–4911, Nov. 2020, ISSN: 1524-9050, 1558-0016. DOI: [10.1109/TITS.2019.2947206](https://doi.org/10.1109/TITS.2019.2947206).
- [60] Z. Chen, J. Zhang, and D. Tao, “Progressive lidar adaptation for road detection,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, pp. 693–702, May 2019. DOI: [10.1109/JAS.2019.1911459](https://doi.org/10.1109/JAS.2019.1911459).
- [61] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, “A²-Nets: Double Attention Networks,” in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018.
- [62] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, “CCNet: Criss-Cross Attention for Semantic Segmentation,”
- [63] W. Shi, J. Caballero, F. Huszar, *et al.*, “Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 1874–1883, ISBN: 978-1-4673-8851-1. DOI: [10.1109/CVPR.2016.207](https://doi.org/10.1109/CVPR.2016.207).
- [64] M. Berman, A. R. Triki, and M. B. Blaschko, “The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 4413–4421, ISBN: 978-1-5386-6420-9. DOI: [10.1109/CVPR.2018.00464](https://doi.org/10.1109/CVPR.2018.00464).
- [65] J. Fritsch, T. Kuhn, and A. Geiger, “A new performance measure and evaluation benchmark for road detection algorithms,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, The Hague, Netherlands: IEEE, Oct. 2013, pp. 1693–1700, ISBN: 978-1-4799-2914-6. DOI: [10.1109/ITSC.2013.6728473](https://doi.org/10.1109/ITSC.2013.6728473).
- [66] *The KITTI Vision Benchmark Suite*. [Online]. Available: https://www.cvlibs.net/datasets/kitti/eval%5C_road.php.
- [67] A. A. Khan, J. Shao, Y. Rao, L. She, and H. T. Shen, “LRDNet: Lightweight LiDAR Aided Cascaded Feature Pools for Free Road Space Detection,” *IEEE Transactions on Multimedia*, pp. 1–13, 2022, ISSN: 1941-0077. DOI: [10.1109/TMM.2022.3230330](https://doi.org/10.1109/TMM.2022.3230330).
- [68] X. Wang, Y. Qian, C. Wang, and M. Yang, “Map-Enhanced Ego-Lane Detection in the Missing Feature Scenarios,” *IEEE Access*, vol. 8, pp. 107 958–107 968, 2020, ISSN: 2169-3536. DOI: [10.1109/ACCESS.2020.3000777](https://doi.org/10.1109/ACCESS.2020.3000777).

- [69] Y. Lyu, L. Bai, and X. Huang, “Road Segmentation using CNN and Distributed LSTM,” in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5. DOI: [10.1109/ISCAS.2019.8702174](https://doi.org/10.1109/ISCAS.2019.8702174).
- [70] Z. Chen and Z. Chen, “RBNet: A deep neural network for unified road and road boundary detection,” in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds., Cham: Springer International Publishing, 2017, pp. 677–687, ISBN: 978-3-319-70087-8.