UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Navigability estimation for autonomous vehicles using machine learning

Caio César Teodoro Mendes

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Caio César Teodoro Mendes

Navigability estimation for autonomous vehicles using machine learning

Doctoral dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Denis Fernando Wolf

USP – São Carlos
August 2017

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

T538n

Teodoro Mendes, Caio César
Navigability estimation for autonomous vehicles using machine learning / Caio César Teodoro Mendes; orientador Denis Fernando Wolf. -- São Carlos, 2017.
94 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2017.

1. Road detection. 2. Obstacle detection. 3. Machine learning. 4. Deep learning. 5. Stereo vision. I. Wolf, Denis Fernando, orient. II. Título.

Caio César Teodoro Mendes

Estimação de navegabilidade para veículos autônomos usando aprendizado de máquina

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Denis Fernando Wolf

USP – São Carlos
Agosto de 2017

ACKNOWLEDGEMENTS

AGRADECIMENTOS

Gostaria de agradecer ao meu orientador, Denis Fernando Wolf, pelo conhecimentos, motivação e principalmente pela paciência que teve comigo. Também ao demais alunos, professores e frequentadores do laboratório de robótica móvel (LRM) pelas eventuais contribuições, trabalhos em conjunto e ideias. Ao professor Vincent Fremont por me aceitar e receber para um ano de estágio na França. À SEMCON e meus colegas que lá trabalham pelo bom entendimento do tempo que tive que dedicar a escrita da tese. Agradeço ainda a Ariadne pela ajuda e finalmente à FAPESP (proc. 2011/21483-4) e ao CNPq pelas bolsas concedidas.

ABSTRACT

MENDES, C. C. T. **Navigability estimation for autonomous vehicles using machine learning**. 2017. 94 p. Doctoral dissertation (Doctorate Candidate Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Autonomous navigation in outdoor, unstructured environments is one of the major challenges presents in the robotics field. One of its applications, intelligent autonomous vehicles, has the potential to decrease the number of accidents on roads and highways, increase the efficiency of traffic on major cities and contribute to the mobility of the disabled and elderly. For a robot/vehicle to safely navigate, accurate detection of navigable areas is essential. In this work, we address the task of visual road detection where, given an image, the objective is to classify its pixels into road or non-road. Instead of trying to manually derive an analytical solution for the task, we have used machine learning (ML) to learn it from a set of manually created samples. We have applied both traditional (shallow) and deep ML models to the task. Our main contribution regarding traditional ML models is an efficient and versatile way to aggregate spatially distant features, effectively providing a spatial context to such models. As for deep learning models, we have proposed a new neural network architecture focused on processing time and a new neural network layer called the semi-global layer, which efficiently provides a global context for the model. All the proposed methodology has been evaluated in the Karlsruhe Institute of Technology (KIT) road detection benchmark, achieving, in all cases, competitive results.

Keywords: Road detection, Obstacle detection, Machine learning, Deep learning, Stereo vision.

RESUMO

MENDES, C. C. T. **Estimação de navegabilidade para veículos autônomos usando aprendizado de máquina**. 2017. 94 p. Doctoral dissertation (Doctorate Candidate Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

A navegação autônoma em ambientes externos não estruturados é um dos maiores desafios no campo da robótica. Uma das suas aplicações, os veículos inteligentes autônomos, tem o potencial de diminuir o número de acidentes nas estradas e rodovias, aumentar a eficiência do tráfego nas grandes cidades e contribuir para melhoria da mobilidade de deficientes e idosos. Para que um robô/veículo navegue com segurança, uma detecção precisa de áreas navegáveis é essencial. Neste trabalho, abordamos a tarefa de detecção visual de ruas onde, dada uma imagem, o objetivo é classificar cada um de seus pixels em rua ou não-rua. Ao invés de tentar derivar manualmente uma solução analítica para a tarefa, usamos aprendizado de máquina (AM) para aprendê-la a partir de um conjunto de amostras criadas manualmente. Nós utilizamos tanto modelos tradicionais (superficiais) quanto modelos profundos para a tarefa. A nossa principal contribuição em relação aos modelos tradicionais é uma forma eficiente e versátil de agregar características espacialmente distantes, fornecendo efetivamente um contexto espacial para esses modelos. Quanto aos modelos de aprendizagem profunda, propusemos uma nova arquitetura de rede neural focada no tempo de processamento e uma nova camada de rede neural, chamada camada semi-global, que fornece eficientemente um contexto global ao modelo. Toda a metodologia proposta foi avaliada no benchmark de detecção de ruas do Instituto de Tecnologia de Karlsruhe, alcançando, em todos os casos, resultados competitivos.

Palavras-chave: Estimação de rua, Detecção de obstáculos, Aprendizado de máquina, Aprendizagem profunda, Visão estéreo.

CONTENTS

CHAPTER

1

INTRODUCTION

Autonomous vehicles can serve many practical purposes. They can be cars that drive themselves carrying passengers and goods or agricultural machines that free individuals from tedious and dangerous tasks. Such vehicles and, more generally, Advanced Driver Assistance Systems (ADAS) come with the promises of increasing traffic safety, bringing mobility to the disabled, improving fuel efficiency and overall transforming the transport landscape. One of these promises, increasing traffic safety, is especially appealing. According to Vos *et al.* (2015), in 2013 53 million people suffered from road injuries and 1.4 million died as result of such injuries. The most common cause of accidents is driver error, they are present in about 93% of all accidents (LUM; REAGAN, 1985). Autonomous vehicles and ADAS may significantly reduce the number of accidents by minimizing or even eliminating the human role in driving.

To accomplish such a useful feat, vehicles must sense the environment through sensors, detect transversable regions and move through them precisely and efficiently. However, this is a notoriously difficult task as it involves many technical challenges. These challenges range from dealing with intrinsically noisy sensors to correctly identifying road signs. Nevertheless, automotive companies are heavily investing in this technology and partial-automated driving features are offered by many of them. For instance, Mercedez-Benz (a division of the company Daimler AG) offers automatic emergency braking (AEB), lane keeping assistance (LKA) and adaptive cruise control (ACC) in the Mercedes-Benz E and S-Class models. Since October 2015 Tesla Motors offers the *Tesla Autopilot* in their Model S vehicle, this feature allows hands-off automated driving in highways (NELSON, 2015). Moreover, the investment of the industry is expected to rise in the following years. Ambroggi and Kona (2014) predicts that the vehicle-related semiconductor industry revenue will reach half a billion dollars in 2020.

A crucial task of an autonomous navigation system or an ADAS is road detection. That is, detecting the free road surface ahead of the ego-vehicle. It also constitutes an

important preprocessing step for other tasks such as vehicle and pedestrian detection. Once the free road surface is detected, the search space for those objects is reduced, thus contributing to a lower processing time and false detection reduction. Road detection is a challenging task due to the dynamic nature of outdoor scenarios and vehicle motion. Therefore a road detection system must deal with a continuously changing background, the presence of different objects (vehicles, pedestrians) and variable ambient conditions (illumination, weather). Moreover, algorithms must deal with high intra-class variability since there are different road types (color, shape, size).

Most autonomous vehicles prototypes rely partially or fully on active sensors (BUEHLER *et al.*, 2010), such as LIDARs (Light Detection and Ranging) sensors, for road detection. Cameras, on the other hand, tend to play a secondary role. The reason for this is twofold: (I) LIDARs have a centimetric precision even over long distances (e.g. 80 meters), and (II) obstacles and road are usually defined in terms of their geometric properties and, since LIDARs provide precise geometric information, the data interpretation is direct. Nevertheless, there are limitations of solutions employing LIDARs. Those include its monetary cost, which prevents many commercial applications; its relatively low resolution, which makes recognizing distant objects difficult; and its poor color sensitivity, which makes it unable to perform tasks such as traffic light state recognition. A possible alternative is the use of digital cameras.

Cameras are affordable sensors, costing a fraction of the cost of a LIDAR sensor. They provide a rich description of the environment in terms of color and texture, and due to their reduced weight and size, they provide flexibility in handling and assembling. Using a pair of cameras it is possible to create a stereo system or stereo camera. With the use of such camera, together with a stereo method, it is possible to obtain geometrical information about the environment. The challenge in the use of cameras is the interpretation of data: how to determine if an object represents an obstacle in terms of properties such as color or texture? Or, in the case of stereo cameras, how to deal with the high level of noise present in the geometrical information? Moreover, there are still issues related to lighting that can compromise the performance of the sensor or interfere with the consistency of certain information, such as the color of an object.

In this thesis, we explore the use of digital cameras (stereo and monocular) in the task of road detection. We treat road detection as a supervised machine learning task, that is, we try to learn the task by using a set of images (examples) in which the road was correctly detected. The hope is that the learned model generalizes well beyond the examples and works as a general road detector. We employ two types of machine learning models to this end, classical (or shallow) models and deep learning models. The main difference between the two is that classical models are unable to work directly with high dimensional data (e.g. images) and require a manually engineered pre-processing step called feature

Figure 1 – Vehicular platforms of the CaRINA project

(a) CaRINA I (b) CaRINA II

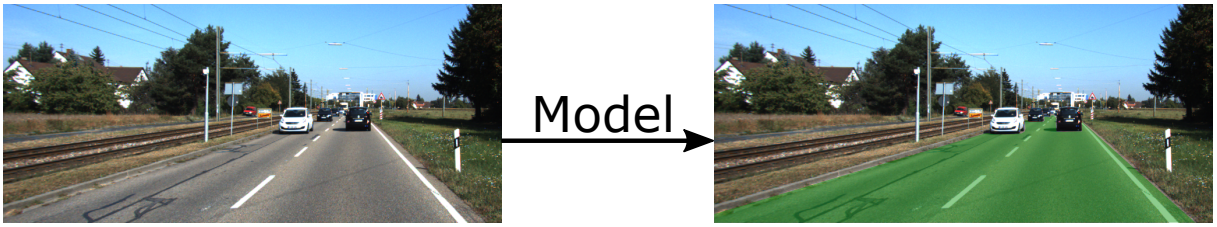
Source: Elaborated by the author.

extraction. The objective of this step is to create a more relevant and low dimensional representation of the input data. Naturally, while using classic models, we focused on devising efficient ways to create such representations. By taking advantage of the geometric information provided by a stereo camera we have also created an obstacle detector and used its output as features to classical models. While using deep learning models, we switched our focus from devising features to the models themselves. Deep learning models are able to create such low dimensional representations (features) automatically, as part of the learning process. Hence we focused on creating deep architectures and "layers" specifically for the task of road detection.

It is important to note that this work is part of larger project called *Carro Robótico Inteligente para Navegação Autônoma* (CaRINA) from the Portuguese, Intelligent Robotic Car for Autonomous Navigation, which is a Brazilian autonomous vehicle research project. The project is hosted by the *Laboratório de Robótica Móvel* (Mobile Robotics Laboratory)[1] in the University of São Paulo in São Carlos, and it receives funding from the agencies CNPq, CAPES, and FAPESP. The objective of the project is to research and develop technologies related to autonomous vehicles, including the development of autonomous vehicles prototypes. The project started in April 2010, with the acquisition of an electric vehicle, named CaRINA I (Figure 1a). Modifications were made to allow its electronic control and in October of the same year, the vehicle performed its first autonomous navigation test. A few months later, using a camera to detect the road, the vehicle was

[1] http://www.lrm.icmc.usp.br/

Figure 2 – The visual road detection task: given an image (or pair of images), classify every of its pixels into road or non-road



Source: Elaborated by the author.

able to transverse one kilometer without human interference. In 2011, CaRINA I was superseded by the CaRINA II vehicle, a FIAT Palio Weekend (Figure 1b). This vehicle also went through modifications so it could be electronically controlled and, in 2012, it drove 15 kilometers autonomously throughout the public roads of São Carlos (RODRIGUEZ; COSENTINO, 2014). Further details from both platforms are available in the work of Fernandes *et al.* (2014). A number of research topics have been addressed as part of the CaRINA project, including path planning (MATIAS *et al.*, 2015), control theory (MASSERA *et al.*, 2016), obstacle detection (SHINZATO *et al.*, 2014), localization (HATA; WOLF, 2015) and pedestrian detection (ALENCAR *et al.*, 2014). This work fits in the perception branch of the CaRINA project and it can be used to aid the vehicles prototypes to safely navigate in urban environments.

1.1 Problem Statement

The objective of road detection is to segment a given sensor input into road and non-road. In the specific case of visual road detection, an image (or pair of images in the case of a stereo camera) is the input. Therefore the objective of visual road detection is to classify each pixel of the image into road or non-road as illustrated by Figure 2. More formally, we would like to learn a transformation $\mathbf{RD} : \mathbf{X} \rightarrow \mathbf{Y}$ which maps the input image $\mathbf{X}$ to the target segmentation $\mathbf{Y}$. There are a number of ways to address the visual road detection task. In this thesis, we will focus on machine learning methods where, instead of manually creating the mapping $\mathbf{RD}$, we will *learn* it from data.

The term "road" is generic and may be used in distinct contexts. For instance, it might mean a dirty, unpaved course, a highway or an urban street. In this work, we use the term to refer to unobstructed urban streets and highways. And, even though most people have o good intuition of what is an urban street or a highway, pinpointing those structures in an image is challenging. For instance, are lane markings part of the highway or a distinct element in the scene? What about speed bumps and potholes? These sort of ambiguous situations starts to come up as soon as you try to manually classify every

pixel of an image into road and non-road. For that reason, we use a more restricting road definition and assume the road to be the artificially made navigable space surrounding the ego-vehicle excluding, for example, sidewalks. Moreover, as we employ the KITTI dataset (FRITSCH *et al.*, 2013) to learn the task and evaluate the obtained results, we assume their road definition to be the ground-truth.

Cameras, both traditional and stereo, need light to operate hence we restricted the scope of our efforts to sunlight illuminated scenes. In addition, we only consider good weather situations, ignoring any weather condition that may negatively affect the camera visibility or function (e.g. snow, rain, fog, etc.). Such conditions, especially snow, might require specialized solutions that are out of the scope of this thesis. It is also important to distinguish road detection from other similar tasks. The ego-lane detection detects only the current lane in which the vehicle is located while road detection does not make distinctions between lanes and segments all the visible road. The task commonly know as "free-space estimation" aims the detection of any unobstructed space regardless of the underlying surface and it may, for instance, detect short grass as navigable. In road detection, the underlying surface is a defining characteristic of the road. Road detection is also intended to be more directly applicable for urban autonomous driving, detecting only legally drivable surfaces.

1.2 Contributions

The main contributions presented in this thesis are as follows:

- a method to efficiently aggregate spatially distant image features. This approach is relevant in tasks where the spatial context is important and the aggregated images features are used in conjunction with a classic/shallow ML model;
- a deep neural network architecture for road detection, which combines elements of two architectures presents in the literature and focus on real-time performance;
- a new deep neural network layer named semi-global layer, which efficiently aggregates spatially distant information and was specially designed to work with two-dimensional data;
- a high-performance multi-core and Graphics Processing Unit (GPU) implementation of a generic sparsely connected neural network layer.

1.3 Thesis Outline

The remanding of the thesis is divided into 6 chapters. In Chapter 2 we provide the relevant technical background to the reader, introducing the concepts and methods that

will be referred throughout this thesis. In Chapter 3 we introduce relevant related works. In Chapter 4 we present our approach to solve the road detection task using classical or shallow machine learning models and its results, by 'classical' we refer just to the model and not to the whole approach. Our main contribution in this chapter is an efficient way to feed contextual information to the model. The Chapter 5 presents our efforts to perform road detection using deep learning. We propose a deep neural network architecture focusing on fast inference time and also a new neural network layer. In Chapter 6 we present a discussion related to the main results achieved in this thesis. Finally, in Chapter 7 we draw the conclusions and suggest future works.

CHAPTER

2

TECHNICAL BACKGROUND

In this chapter, we provide the relevant technical background for the reader, covering theoretical concepts, terminology, and methods that were used in this thesis. We start by providing an overview of machine learning, neural networks and deep learning in Section 2.1. Stereo vision and associated concepts are covered in Section 2.3. The reader may skip this chapter if he or she is already familiar with the mentioned concepts and methods.

2.1 Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence that deals with the research and development of algorithms and methods that are able to improve their performance based on previous experience (MITCHELL, 1997). Machine learning algorithms are designed to identify patterns or extract rules from a set of samples (i.e. training data) that refers to the target task. This learning enables the estimation of the desired solution for a new instance of the task. Machine learning methods are useful in cases where a direct analytic solution is difficult to attain and relevant data is available or can be created. Given their ability to learn, these algorithms can also be adaptive, that is, they can adapt themselves to variations in the scope of the problem without the need to develop an algorithm for each situation (ALPAYDIN, 2004).

The learning regime of ML methods can be divided into three categories: supervised, unsupervised and reinforcement learning. Supervised learning refers to methods that learn from a set of correctly done instances of the target task. For instance, given the task of classifying emails as spam or non-spam, supervised methods require a set of correctly classified emails, which would be the training data. Unsupervised learning, on the other hand, does not require such supervision and its purpose is to find patterns and/or similarities within the input data. It could be used to group emails by their similarities for instance.

Reinforcement learning is a middle term between the two, where the supervision is sparse or time delayed. An example of a task where reinforcement learning is applicable is the game of chess, where the supervision or correct action is usually not available for every move but only at the end of each game (i.e. whether the machine won or lost the game).

Machine learning models can be divided into parametric and instance-based models. Parametric models are, as the name suggests, defined by their parameters. The learning process for such models consists in learning the best parameters values for a given task based on the training data. An example of a classic parametric model is the logistic regression, which is a linear binary classifier (i.e. it creates a linear mapping between the input and output, where the output can be either true or false). Instance-based models also tend to have parameters, but they are mostly defined by the training samples (instances) themselves. These models stores a subset or the whole training set as result of the learning process and perform predictions based on the stored samples. A popular instance-based model is the Support Vector Machine (SVM). An SVM model is defined by a subset of weighted training samples, called support vectors in this context. SVM models are linear models, that can be used for both classification (predict a value from a discrete set) or regression (predict a real-valued output). SVMs can also be used for nonlinear tasks by using the "kernel trick" (AIZERMAN *et al.*, 1964), which consists of transforming the input data to a higher dimensional space using a kernel function.

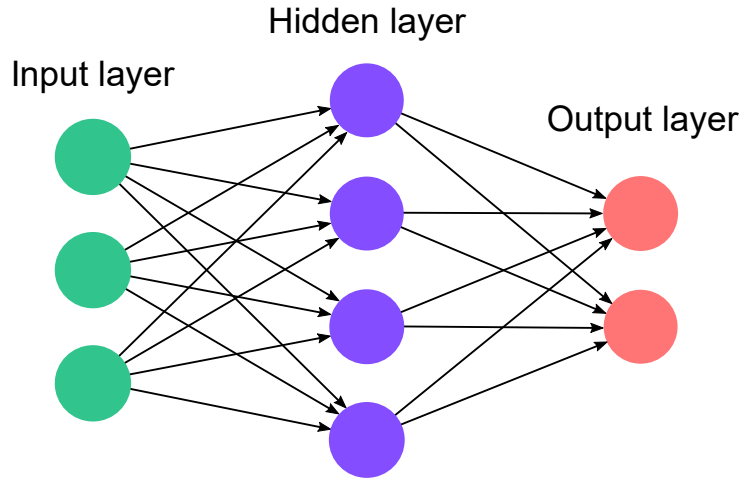
In this thesis, we will be using supervised machine learning to solve the task of road detection. The specific set of models/methods chosen to learn the task are grouped under the name of artificial neural networks, which will be detailed next.

2.1.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are a set of mathematical models and learning methods inspired by the way the biological nervous system works. The defining characteristic of these models is the presence of processing elements called (artificial) neurons which, as in a biological nervous system, are interconnected to form a network. ANNs may vary in the specific neuron implementation, network connections, and learning process and, as a consequence, they are versatile models that can be used to tackle a diverse set of tasks. They have been used under all learning regimes (i.e. supervised, unsupervised and reinforcement learning) and were successful in a number of applications, including speech recognition, image classification, game-playing, decision making, medical diagnosis and natural language processing (HAYKIN, 1998; GOODFELLOW *et al.*, 2016).

The artificial neuron, originally designed in 1943 (MCCULLOCH; PITTS, 1943), works by summing its weighted inputs and passing the result through an activation or

Figure 3 – The usual artificial neural network Multi-Layer Perception (MLP) architecture with only one hidden layer



Source: Elaborated by the author.

transfer function. Mathematically, a neuron with n inputs can be described by equation

$$y = \sigma\left(\sum_{i=1}^{n} x_i w_i + b\right) = \sigma(\mathbf{x} \cdot \mathbf{w}^\intercal + b), \tag{2.1}$$

where $x_1, x_2 \ldots x_n$ are the inputs, $w_1, w_2 \ldots w_n$ are the weights, b is the bias and σ the activation function. The specific activation function vary according to the task and overall structure of the network, originally a simple step function was used. The bias can be seen as a weight independent of the input, functioning as the name suggests, as a bias. Geometrically, a neuron creates an hyperplane where the bias term is the hyperplane origin off-set or y-intercept.

Probably the most well-known and simpler ANN architecture is the Multi-layer Perceptron (MLP) (CHURCHLAND; SEJNOWSKI, 1994). MLPs learning is supervised and they can be used for both regression and classification. In this architecture, the neurons are divided into layers and arranged in sequence as shown in Figure 3. Usually, an MLP model has three layers: the input layer, which receives the input data, the hidden layer, and the output layer, which provides the results. Neurons are only present in the hidden and output layer, the input layer merely illustrates the input data. Every layer in an MLP is a fully connected layer, meaning that every neuron of a given layer is connected to all neurons in the previous layer. If we denote $\mathbf{a}_{l-1}$ as the output of the layer $l-1$ arranged in a N dimensional vector, $\mathbf{W}_l$ to be the matrix of weights of layer l, where $w^l_{i,j}$ connects the neuron j of the layer $l-1$ with the neuron i of layer l, $\mathbf{b}_l$ as a vector with the biases of the neurons in layer l and σ_l to be an activation function used in the layer l. Then, the output vector $\mathbf{a}_l$ from the layer l is given by

$$\mathbf{a}_l = \sigma_l\left(\mathbf{W}_l \cdot \mathbf{a}_{l-1} + \mathbf{b}_l\right), \tag{2.2}$$

where $\cdot$ represents the matrix-vector product. To calculate the output of the network, this operation should be performed for every layer, starting with the first hidden layer which takes a sample $\mathbf{x}$ arranged in a vector as input. Common activation functions for hidden layers of MLPs include the sigmoid function

$$\text{sig}(x) = \frac{1}{1 + e^x}, \tag{2.3}$$

which is a "s" shaped function that squashes the output in a value between 0 and 1 and the hyperbolic tangent function

$$\text{tahn}(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{2.4}$$

which achieves a similar effect squashing the output between -1 and 1.

A more recent activation function is the Rectified Linear Unit (ReLU) function, defined as:

$$\text{ReLU}(x) = \max(0, x). \tag{2.5}$$

This function adresses the saturation problem (i.e. there is no differentiation between large values) that can happen with the previously described activation functions. This problem is more evident in deep architetures (which we shall see next) and hence the ReLU fuction is widely used in such architectures.

The activation function of the last layer depends on the specific task. For instance, the sigmoid function could be used for binary classification. For multi-class classification, where the possible number of outputs is larger than two and classes are mutually exclusive, the Softmax function is a common choice and can be formalized as

$$\text{sm}(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \tag{2.6}$$

where $\text{sm}(x_j)$ is the output of neuron j of the output layer. The Softmax function enforces that the sum of the outputs of all its neurons adds up to one, hence it creates a probability distribution over the possible classes. For regression, where the output is real-valued, there is no need for an activation function in the output layer.

Training an ANN is done by minimizing an error function, also known as loss function. In the supervised case, this function compares the correct output $\hat{\mathbf{y}}_\mathbf{s}$ for a sample s with the output from the network $\mathbf{y}_\mathbf{s}$ for the same sample and returns an estimated error or mismatch between the two. The objective of the training is to adjust the network

parameters Θ to minimize the loss function over all samples and can be generally defined as:

$$\min_{\Theta} \quad \frac{1}{S}\sum_{s=1}^{S} \ell(\hat{\mathbf{y}}_\mathbf{s}, \mathbf{y}_\mathbf{s}; \Theta), \tag{2.7}$$

where S is the number of samples used for training. An example of loss function commonly used for regression is the mean squared loss, defined as

$$\ell(\hat{\mathbf{y}}_\mathbf{s}, \mathbf{y}_\mathbf{s}; \Theta) = (\hat{\mathbf{y}}_\mathbf{s} - \mathbf{y}_\mathbf{s})^2. \tag{2.8}$$

Regularization techniques are an important part of training an ML model. They are used to improve the generalization of the model or, in other words, they try to avoid that the model simply memorizes the training samples (i.e. overfit the training data) instead of learning from it. A common regularization technique used in neural networks consists in impeding large weights in the network and it is known as weight decay. One way to imped large weights is to add the weights magnitude to the loss function, therefore the minimization process would not only try to learn from the training samples but also keep the weights magnitude low. An example of loss function with l2 weight decay is:

$$\ell(\hat{\mathbf{y}}_\mathbf{s}, \mathbf{y}_\mathbf{s}; \Theta) = (\hat{\mathbf{y}}_\mathbf{s} - \mathbf{y}_\mathbf{s})^2 + \gamma\Theta^2, \tag{2.9}$$

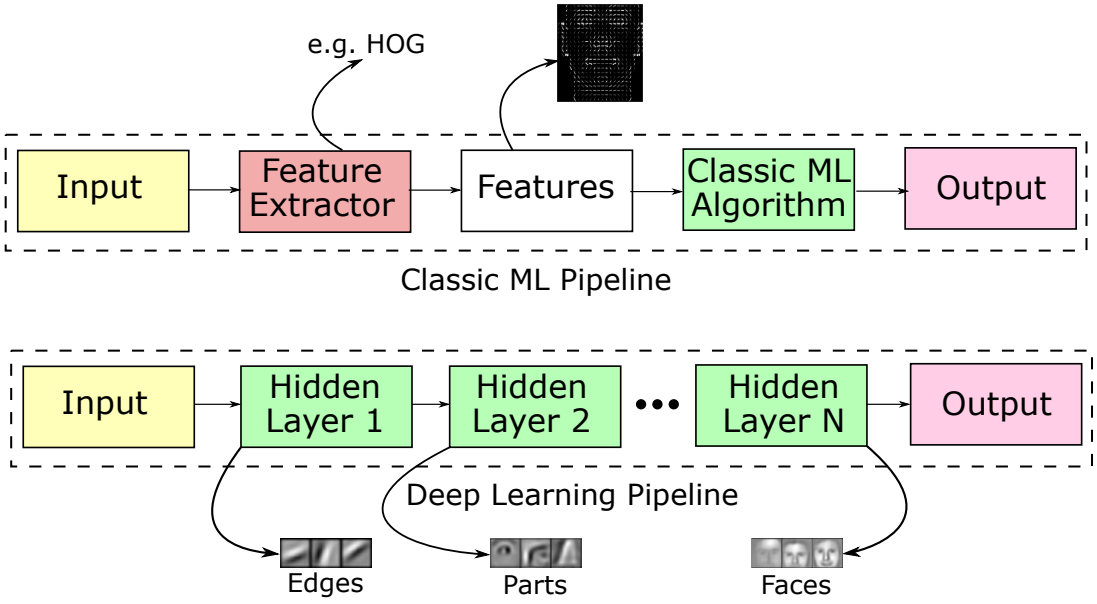
where γ controls the amount of regularization.

To minimize the loss function over all samples the Stochastic Gradient Descent (SGD) optimization method is the most common choice. Given a differentiable loss function ℓ, the SGD updates the parameters of the network according to

$$\Theta \leftarrow \Theta - \frac{\alpha}{m}\sum_{i=1}^{m} \frac{\partial \ell(\hat{\mathbf{y}}_\mathbf{i}, \mathbf{y}_\mathbf{i}; \Theta)}{\partial \Theta}, \tag{2.10}$$

where α is the learning rate, m is a number between 1 and S (the total number of samples), also known as mini-batch size. The SGD method updates the parameters according to a subset m of training samples, effectively performing an approximation (hence stochastic) of the original minimization problem, which is defined over all samples (see Equation 2.7). In this setup, a training cycle is usually defined in *epochs*, i.e. the number of times all samples were used to train the network. The number of parameters updates in an epoch is given by S/m, that is, the total number of samples divided by the mini-batch size. Using the SGD requires the partial derivatives of the network output in relation to its parameters, they can be calculated using the chain-rule, also know as back-propagation in this context (RUMELHART *et al.*, 1986).

Figure 4 – Comparison between a face recognition system using the classic machine learning pipeline and the deep learning one, where HOG refers to the Histogram of Oriented Gradients feaature extractor and blocks shown in green are learned from data



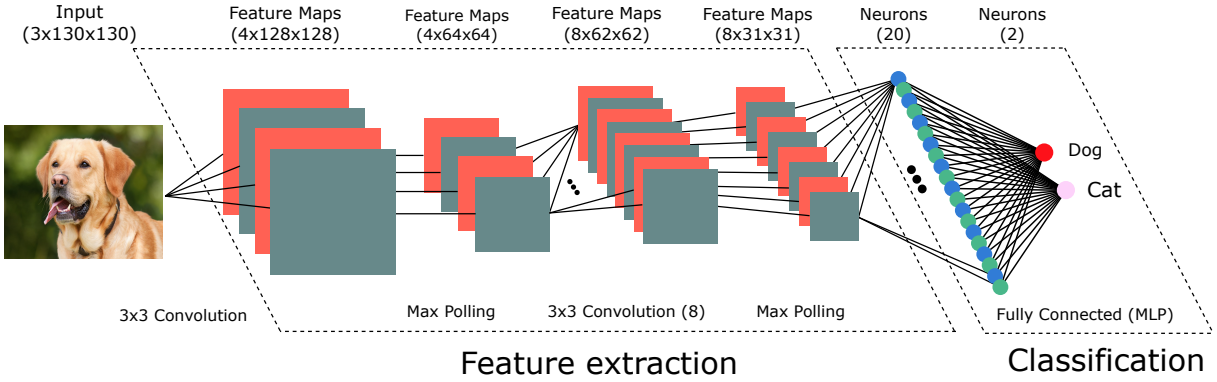
Source: Elaborated by the author.

2.2 Deep Learning

Deep learning is a branch of machine learning that is characterized by making use of models composed of multiple nonlinear layers of transformations (LECUN *et al.*, 2015), hence the name "deep". Each layer transforms the representation (or features) of the previous layer into a higher, more abstract representation. An illustration of this process is shown in Figure 4. There are two key aspects of this process: (I) it is able to start from raw data (e.g. pixels); (II) these transformations are learned from data, using a general learning algorithm. In a classification task, the classifier and the transformations can be joined in a single model and learned jointly, where higher layers tend to learn discriminative features while suppressing irrelevant noise.

In a visual task, traditional or shallow machine learning methods have a limited role. First, the raw pixels values need to be transformed into a vector of "features", such transformation in made using a hand-crafted feature extractor. This transformation attempts to create a more relevant low dimensional representation of the input. Feature extractors are usually task-specific and take advantage of the human ingenuity and eventually known priors. They also may be specific to the type of classifier (e.g. linear or non-linear) and, as such, are hardly useful in other contexts. In this traditional pipeline, the performance is mostly dependent on the feature extractor, and the classifier accuracy contingent on it. Deep learning techniques have substituted this labor-intensive task of developing feature extractors in many tasks and have recently advanced the state-of-the-art

Figure 5 – Typical CNN architecture. It has a series of convolutions and polling layer, referred by some as the feature extraction phase. The second part, called of classification here, consists of a series of fully connected layers, an MLP basically.



Source: Elaborated by the author.

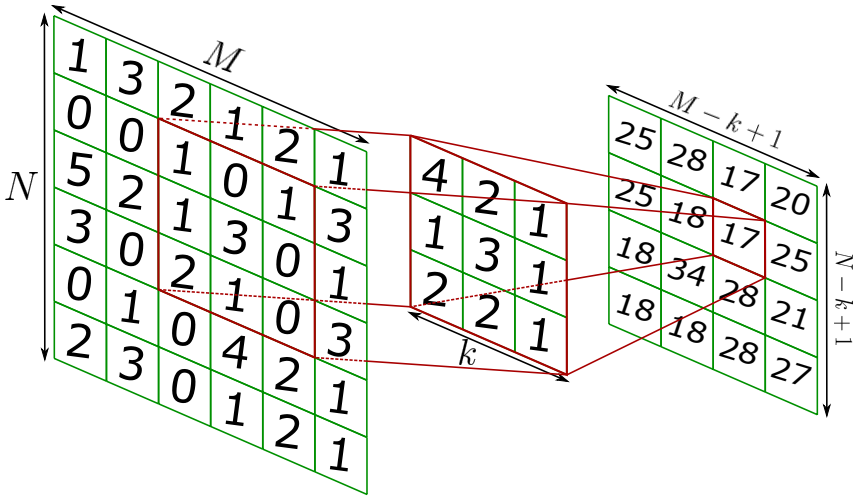
in image recognition (FARABET *et al.*, 2013; SZEGEDY *et al.*, 2014), speech recognition (HINTON *et al.*, 2012; MIKOLOV *et al.*, 2011) and others (XIONG *et al.*, 2015; MA *et al.*, 2015). While most deep models required an unsupervised pre-training stage, one particular type of deep model called Convolutional Neural Network (CNN) can be trained in a fully supervised manner and its recent successes made its use widespread in the computer vision community. As this project addresses one particular application of image segmentation, we will be focusing on CNNs as they are the deep model of choice for image applications.

2.2.1 Convolutinal Neural Networks

Convolutional Neural Networks (CNNs) are feedforward neural networks that make use of convolutional layers. Although its initial theory dates back to 1980 (FUKUSHIMA, 1980) its practical success started in 1998 with LeCun's improved design (LECUN *et al.*, 1998) and consolidated in 2012 with the results obtained in the ImageNet competition (KRIZHEVSKY *et al.*, 2012). The later success popularized these networks and was partially due to the efficient use of Graphics Processing Units (GPUs) for training and a new regularization technique called dropout (SRIVASTAVA *et al.*, 2014). Nowadays CNNs are a sensible choice for many visual tasks including object recognition, detection and image segmentation.

CNNs can be viewed as an enhancement of the standard Multi-Layer Perceptrons (MLPs), where the main difference is the addition of convolutional layers. In the two-dimensional case, such layers perform an operation similar to a standard image convolution but instead of using hard-coded kernels, the network learns the kernels' parameters (weights) through back-propagation. Such layers allow the training of much larger (deep) networks and are able to learn directly from images (or raw input in general) avoiding, therefore, the need for task-specific hand-crafted features. As Figure 5 shows, a typical

Figure 6 – Illustration of a two-dimensional convolution with a stride of one. The kernel scans the input matrix, multiplying and summing matching values. The kernel values, also known as weights or parameters, are learned during the training of a convolutional layer



Source: Elaborated by the author.

CNN architecture consists of a repeated sequence of convolutional and polling layers, referred by some authors as the features extraction phase, followed by a number of fully connected layers. The intermediary matrices between convolutions are known as "feature maps", but one can view them as images or as neurons arranged in a three-dimensional grid. The training is performed using stochastic gradient descent and the gradients are computed according to the chain rule, exactly as in a standard MLP.

Convolutional Layer

The convolutional layer is the defining layer of a CNN. In the simplest two-dimensional case, given a $k \times k$ kernel $\mathbf{k}$, a stride of one and a $M \times N$ input matrix (it could be an image) $\mathbf{X}$, it produces a $(M - k + 1) \times (N - k + 1)$ matrix (feature map) $\mathbf{Y}$ as output. The underlying operation can be mathematically described by:

$$\mathbf{Y}_{i,j} = \sum_{i'=1}^{k} \sum_{j'=1}^{k} \mathbf{k}_{(i',j')} \mathbf{X}_{\left(i - \frac{k+1}{2} + i', j - \frac{k+1}{2} + j'\right)}, \tag{2.11}$$

where the subscripts are used for indexing. Throughout this thesis we may also represent a generic convolution using the $*$ symbol as in:

$$\mathbf{Y} = \text{conv}(\mathbf{X}; \mathbf{k}) = \mathbf{k} * \mathbf{X}. \tag{2.12}$$

In a convolution, the kernel basically scans the input matrix, multiplying and summing its values with the matching values of the input matrix. A visual illustration of

this operation can be seen in Figure 6. A convolutional layer may also use an activation function. A popular one for this kind of layer is the ReLU function (see Equation 2.5).

More generically, a convolutional layer can be defined by the number of kernels/filters K, the kernels' size k and their stride s, i.e. by how much the kernel moves in relation to the input matrix. In this generic case, the convolutional layer takes a $C \times M \times N$ three-dimensional tensor (it could be a many-channel image) as input and produces a $Y_C \times Y_M \times Y_N$ three-dimensional tensor as output. The size of the output can be calculated according to:

$$Y_C = K; \quad Y_M = \frac{M-k}{s} + 1; \quad Y_N = \frac{N-k}{s} + 1. \tag{2.13}$$

For example, in the case of the first convolution of Figure 5, we have as input an RGB input image, that can be thought of as a 3D tensor with size $3 \times 130 \times 130$. In this layer we have 4 kernels ($K = 4$), all sized 3×3 ($k = 3$) and the traditional stride of 1 ($s = 1$). With this information we can calculate the size of the output:

$$Y_C = 4; \quad Y_M = \frac{130-3}{1} + 1 = 128; \quad Y_N = \frac{130-3}{1} + 1 = 128. \tag{2.14}$$

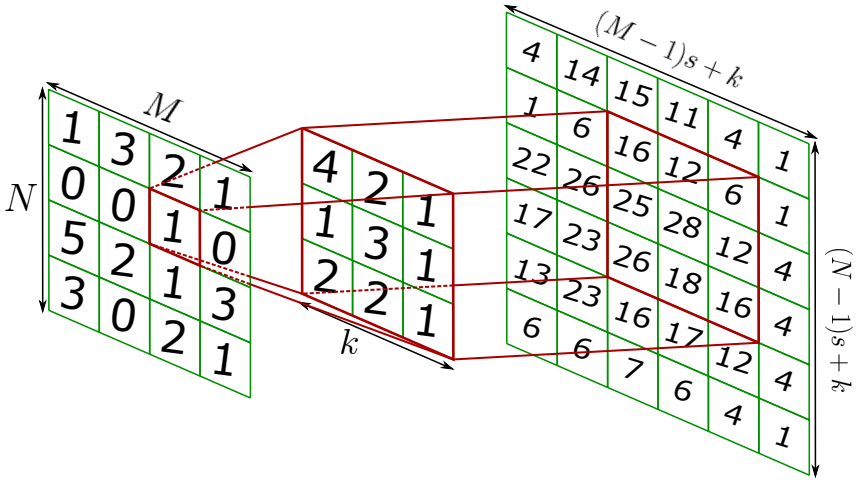
The number of learnable parameters of a convolutional layer is $C \times K \times k \times k + B$, where C is the number of channels of input and the final B refers to the optional bias parameter for each filter. In the case of the previous example, we would have $3 \times 4 \times 3 \times 3 + 4 = 112$ parameters. It should be noted that the effective operation in cases where $C > 1$ is actually a 3D convolution using a 3D filter. Most deep learning libraries handle up to five-dimensional convolutions, these additional dimensions could be used for a time dimension (e.g. when the input is a video), and/or to speed up training by processing multiple samples at once. As convolutional layers reduce the size of the input (even using a unitary stride), it is common practice to pad the input matrix with zeroes so that the output has the same size as the input.

Besides convolutional and fully connected layers, CNNs architectures may include a wide range of other layers. In fact, any differentiable operation can by used as a layer. We will now proceed to describe the most popular ones: deconvolutional, max polling, unpolling and the batch normalization layer.

Deconvolutional Layer

Within the deep learning community, the deconvolutional layer, also known as "fractionally-strided convolution", "backwards convolution","full convolution", or "upconvolution", is widely used for end-to-end image segmentation tasks. In these tasks, the input is the target image and the output the segmentation. One may notice however that

Figure 7 – Illustration of the operation performed by a two-dimensional deconvolutional layer with a stride of one. Each element of the input matrix is element-wise multiplied by the kernel and added to the output matrix



Source: Elaborated by the author.

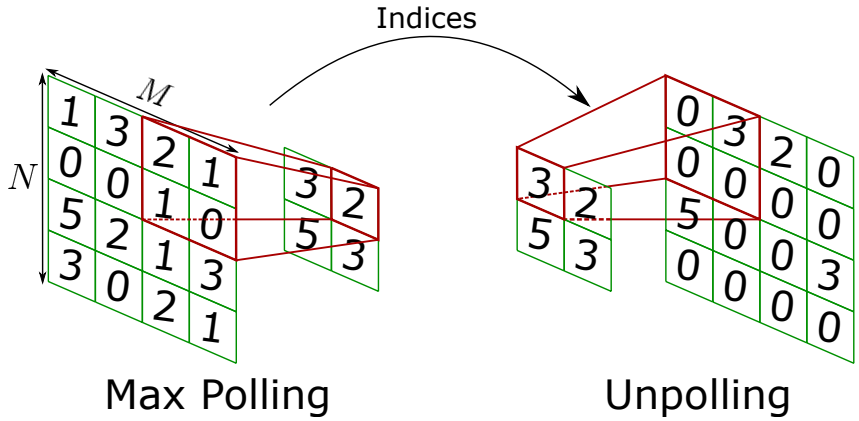
a series of convolutions and polling layers significantly reduces the input size (i.e. the feature map resolution). If no operation is performed to counteract this reduction the only output possible would be a course (low resolution) segmentation of the input. A series of deconvolutional layers could be used as such operation, counteracting the effects of the convolutional and polling layers.

The operation performed by a deconvolutional layer is not a deconvolution as known by the signal processing community. Rather, it is a backwards convolution and could be mathematically defined as the partial derivative of a convolution with respect to its input:

$$\text{deconv}(\mathbf{X};\mathbf{k}) = \frac{\partial \text{conv}(\mathbf{X};\mathbf{k})}{\partial \mathbf{X}}. \tag{2.15}$$

In its simplest two-dimensional case, given a $M \times N$ input matrix $\mathbf{X}$, an output matrix $\mathbf{Y}$ full of zeros, a $k \times k$ kernel and a stride s the operation performed by the deconvolution layer could be explained as follows: for every value of the input, an element-wise multiplication is made between the value and the kernel. The resulting $k \times k$ matrix is element-wise added to a $k \times k$ region of the output matrix. The exact region in the output matrix where this addition happens is defined by the position of the element in the input matrix and by the stride of the deconvolutional layer. An illustration of this operation when the stride is one is given in Figure 7. Instead of scanning the input matrix with a stride s the deconvolutional layer scans the output matrix with a stride of s. Hence, if we increase the stride we could upsample the input. The size of the output $Y_C \times Y_M \times Y_N$ for

Figure 8 – Illustration of the two-dimensional max polling and unpolling operations. In its simplest case, the max polling operation performs a non-overlapping scan of the input matrix with a 2×2 window or kernel and copies the maximum value present in the region to the output. Given the set of selections (indices) made by a polling operation, the unpolling operation simply copies the input to output, placing zeros in the non-selected places



Source: Elaborated by the author.

the generic case where we have K kernels can calculated according to:

$$Y_C = K; \quad Y_M = (M-1)s + k; \quad Y_N = (N-1)s + k. \tag{2.16}$$

Polling and Unpolling Layer

Convolutional layers tend to be followed by polling layers. Such layers reduce the dimensionality of the input by performing non-parametric operations (e.g. average, max) in local regions of the input. Polling layers tend to aggregate similar regions, enforce translation invariance and avoid overfitting. The most popular operation is the max, which takes the maximum of a given region of the input and copies to the output. Concretely, given an $M \times N$ input matrix, a kernel size k and a stride s, the max polling layer scans the input matrix with the $k \times k$ kernel taking the maximum of the matching region and copying it to the output. An illustration of the commonly used case of a stride 2 and a 2×2 kernel is given in Figure 8, where the input is reduced by half. The output size $Y_C \times Y_M \times Y_N$ of a generic three-dimensional max polling layer for a $C \times M \times N$ tensor input is:

$$Y_C = C; \quad Y_M = \frac{M-k}{s} + 1; \quad Y_N = \frac{M-k}{s} + 1. \tag{2.17}$$

Given the indices of a selection-based polling operation such as the max polling, the operation can be performed backwards by an unpolling layer as illustrated in Figure 8. Unpolling layers, similarly to deconvolutional layers, are used to increase the resolution

of the internal representation of the network. They are commonly used together with deconvolutional layers, both counteracting the loss of resolution that has been done by convolutional and polling layers.

Batch Normalization Layer

Training deep neural networks in practice can be quite challenging. One of the reasons is that the parameters of a given layer affect the input of every layer that comes after. During training, when the parameters are constantly being updated, this constant variation makes it hard to select the best parameters for each layer. This is known as *internal covariate shift* and is especially concerning when training deep architectures where the effect is amplified by each layer. A solution to this problem was proposed by Ioffe and Szegedy (2015) in the form of a layer, the Batch Normalization (BN) layer.

When training a neural network with SGD, each parameter update is done based on a subset, called mini-batch, of the training samples (see Equation 2.10). Despite being called a normalization layer, the BN layer works by performing a batch-wise linear transforming of its input. The parameters of such transformation are learned as the parameters of any other layer. The BN layer is usually used after each parametric layer (e.g. fully connect and convolutional) of the network.

Given an input $\mathbf{X} = \{x_1, x_2 \dots x_N\}$, where N is the mini-batch size, parameters γ and β, the batch normalization layer can be formalized as performing the transformation:

$$\mathrm{BN}_{\gamma,\beta} : \{x_1, x_2 \dots x_N\} \rightarrow \{y_1, y_2 \dots y_N\}, \tag{2.18}$$

where $\mathbf{Y} = \{y_1, y_2 \dots y_N\}$ is the output, that is, the transformed input. To perform such transformation, first the layer calculates the mean μ_X and variance σ_X^2 of the batch/input $\mathbf{X}$:

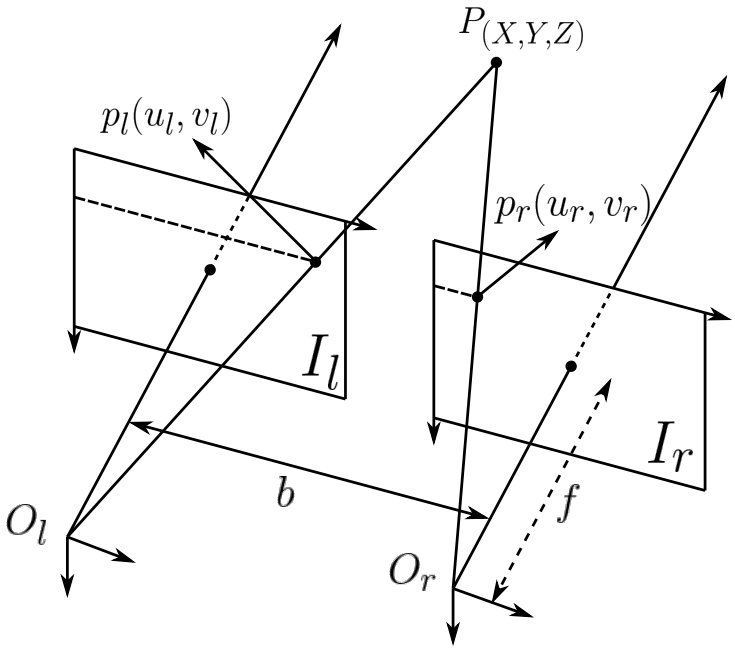
$$\mu_X = \frac{1}{N}\sum_{i=1}^{N} x_i; \qquad \sigma_X^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_X)^2. \tag{2.19}$$

Using the calculated values, the input is now normalized to have a mean of zero and a variance of one:

$$\hat{x}_i = \frac{x_i - \mu_X}{\sqrt{\sigma_X^2 + \epsilon}}, \tag{2.20}$$

where ϵ is a small constant value (usually 0.00001) used to avoid overflows and divisions by zero. Finally, using this normalized input $\hat{\mathbf{X}} = \{\hat{x}_1, \hat{x}_2 \dots \hat{x}_N\}$ the actual linear

Figure 9 – Illustration of the canonical stereo system with two cameras forming two image planes (I_l and I_r) with a baseline b and a focal length f.



Source: Elaborated by the author.

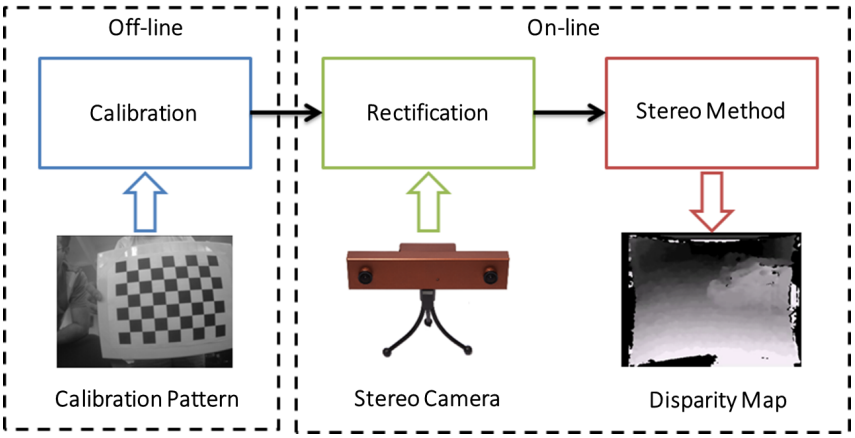
transformation is performed:

$$y_i = \gamma \hat{x}_i + \beta. \tag{2.21}$$

2.3 Stereo Vision

A stereo camera is a combination of two traditional cameras and works similarly to the human vision system: two spatially displaced cameras capture two similar images. An object seen by both cameras would appear in a different location for each camera, which would be reflected in the captured images. The difference in position, from the image of one camera to the image of the other is proportional to the depth of such object. By knowing this difference, the object's depth could be estimated. This process could be done for every point or region seen by both cameras, resulting in a depth map. The most challenging part of estimating depth using a stereo camera is finding corresponding regions between the two images. Given two images, the depth estimation system must scan one image and for each region (pixel) of this image, find the corresponding one in the other image. That is challenging for many reasons, for instance, one region of one image might not be present in the other image due to occlusion. In addition, the appearance of a region in one image may be different from the other because of different cameras parameters (e.g. exposure and focus), illumination and the slightly different angle that the cameras observe the region.

Figure 10 – Flowchart related to the utilization of a stereo camera



Source: Elaborated by the author.

Figure 9 shows the geometry of a canonical stereo system with two cameras. By knowing the focal length (i.e. distance from the optical center to the image plane), the baseline, u_l and u_r, it is possible to know the distance of the point P from the camera using triangulation. Equation 2.22 shows the calculation, being Z the distance, b the baseline, f the focal length and d the difference in the u-coordinate of the corresponding points p_l and p_r, commonly referred as disparity.

$$Z = \frac{b \cdot f}{u_l - u_r} = \frac{b \cdot f}{d} \tag{2.22}$$

To obtain the disparity, that is, the difference on the u-axis of the corresponding points p_l and p_r, it is necessary to use a matching algorithm or stereo method, whose purpose is to find corresponding regions (pixels) between the pair of stereo images. This kind of search has a high computational cost and therefore the possible search area for corresponding points should be minimized. In a perfect canonical system, the search could be constrained horizontally but in practice cameras present distortions and can not be perfectly aligned. To correct for imperfections and hence constrain horizontally the search for corresponding points it is necessary to perform a calibration process. This process infers the distortion and misalignment of the camera pair from a set of pictures containing a calibration pattern, usually a chess board. With this information, it is possible to virtually align the images and remove the distortion in a process called rectification. Figure 10 shows a flowchart regarding the use of a stereo camera.

The stereo method performs, by far, the most challenging and important part of the process of estimating pixels' depth. When searching for a corresponding pixel, the information of the pixel itself is not enough, any noise would cause false correspondences. For this reason, stereo methods always use a support region to help finding the corresponding pixel. As a result, stereo methods are divided according to their use of the support

region into three categories: local, semi-global and global. Local methods use as support region the surrounding region of the target pixel. The most popular local stereo method is the block matching method, which uses a square region around the target pixel as the support region. Therefore it compares two-dimensional regions rather than single pixels to find each pixel's correspondence. Global stereo methods use the whole image as a support region, that is, they consider every pixel's match at once and tend to be formulated as an optimization problem over all correspondences. These methods are inherently computationally demanding and hardly useful for real-time applications as it is often the case in robotics. Semi-global methods are a midterm between local and global methods. As global methods, they are formulated as an optimization problem, but such optimization considers only part of the image. For instance, a popular semi-global method is called Dynamic Programming (DP) (SCHARSTEIN; SZELISKI, 2002). This method's optimization formulation considers all the pixels in the same line as the target pixel, that is, it considers and solves all the correspondences of a line at once. It is called DP because it uses the DP optimization technique to solve the optimization problem. The most popular semi-global method, however, is mostly referred by as simply Semi-Global Method (SGM) and was proposed by Hirschmuller (2005). This method takes into consideration not only the horizontal line in which the target pixel is located but also the vertical and diagonal lines. By taking into consideration these extra lines, it improves its depth estimation accuracy and ensures vertical smoothness in the depth map.

There are some inherent limitations in using a stereo camera for estimating depth. For instance, when an object is too close to the stereo camera only one of its cameras will observe it, making its triangulation impossible. The depth estimation accuracy also tends to degrade with distance, as the differences between the images became increasingly small. In practice, however, the most common and severe source of error come from the stereo method in the form of wrong correspondences. These correspondence errors produce large deviations in the depth map and, more critically, are impossible to avoid and hard to detect. That being the case, stereo camera users must be prepared to deal with such errors. In addition, a stereo camera suffers from the same limitations that affect an ordinary camera as, for instance, the need for lighting.

2.4 Final Considerations

In this chapter, we have provided a quick overview of the main concepts and methods that will be used throughout the thesis. We presented artificial neural networks and gave examples of both shallow (MLP) and deep models (CNN). The MLP model will be used in our classic approach (Chapter 4), while deep architectures, specifically convolutional neural networks, will be used in our deep approaches (Chapter 5). We recommend to the reader that want to dive deeper into this topic the recently published

book (GOODFELLOW *et al.*, 2016). In addition, we have provided a quick overview of stereo vision, which will be used in the classic approach. Stereo methods are also mentioned in the stereo vision section and one of them, the semi-global stereo method, will be used as an inspiration to propose a new neural network layer in Chapter 5.

CHAPTER

3

RELATED WORK

Autonomous vehicles research dates back to the 1920s and 30s ('PHANTOM..., 1926), but the first self-sufficient prototype emerged only in the 1980s with the Navlab (NAVLAB..., 1984) and AVL (WALLACE, 1985) projects in 1984 and the Eureka PROMETHEUS project (EUREKA..., 1987) in 1987. These early prototypes operation was mostly limited to highways and controlled environments as the hardware available at the time was not ideal for the task. Nevertheless, as these projects evolved over time, they began to achieve some significant milestones. In 1994, VAMP and VITA-2, two vehicles of the PROMETHEUS project, drove one thousand kilometers on Paris highways and reached speeds up to 130 km/h. In 1995, members of the PROMETHEUS project working together with Daimler-AG modified an S-Class Mercedes-Benz and went on a semi-autonomous trip of 1600 km, from Munich to Copenhagen. They claimed that, during this trip, the longest section without human intervention was of 156 km which was impressive for the time.

In 2004, a major competition, named Grand Challenge, was put in place by the American Defense Advanced Research Projects Agency (DARPA) to promote research in the area of autonomous vehicles. It consisted of a race of driverless vehicles through a 241 kilometers off-road course with a prize of one million dollars for the winner (WALTON, 2004). The competition proved to be more difficult than expected as most vehicles were unable to reach the second kilometer and none finished the race. Carnegie Mellon University's Red Team with its vehicle nicknamed Sandstorm traveled the farthest distance and completed 11.7 kilometers of the course before getting stuck on a rock. A second Grand Challenge took place in 2005 (THRUN *et al.*, 2007), with an updated prize of two million dollars and a new course of 212 km. This time 5 competitors were able to finish the race and Stanford Racing Team's vehicle (nicknamed Stanley) was the winner with a total time of 6 hours and 54 minutes. Figure 11 shows Stanley crossing the finish line. In 2007, a new Grand Challenge, also know as "DARPA Urban Challenge" took place, this time in an urban setting and a 92 kilometers course (URMSON *et al.*, 2009). Although

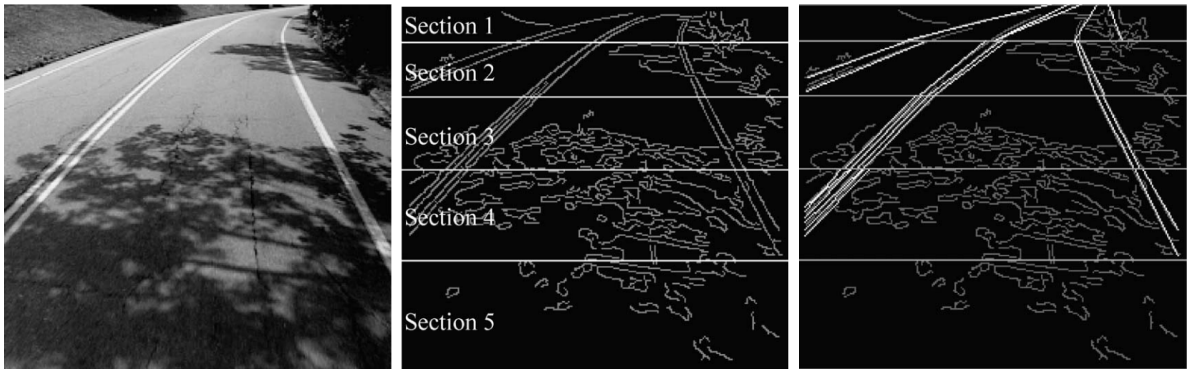
Figure 11 – Stanford Racing Team's vehicle Stanley winning the 2005 Grand Challenge

Source: Thrun *et al.* (2007).

this competition was less physically demanding on the vehicles it was far more challenging in terms of "intelligence". Participating vehicles had to obey California's traffic regulations and negotiate the traffic with other autonomous vehicles and human-controlled vehicles. Team Tartan Racing from Carnegie Mellon University finished the race in first with a total time of 4 hours and 10 minutes. Stanford's team, the previous Grand Challenge winner, finished in second with their vehicle (a Volkswagen Passat Wagon) nicknamed Junior. In third came the VictorTango team, from Virginia Tech. In total, 6 teams finished the race but only 4 within the maximum allowed time (6 hours).

The European counterpart of the Grand Challenge is called European Land-Robot Trial (ELROB) (SCHNEIDER, 2017) but rather than a competition it is a demonstration of the tasks land-robots can perform. The event takes place every year alternating between civilian and military scenarios. The first event happened in 2006 with military scenarios concentrated on navigation in both indoor and outdoor environments. These events, particularly the Grand Challenge, fomented the academic research and the public interest in autonomous vehicles. Today, a number of companies have presented autonomous prototypes. Google probably owns the most advanced one as they hired key members of the 2005 and 2007 Grand Challenge winners, for instance, Sebastian Thrun the team leader of Stanford Racing and Chris Urmson, Tartan Racing' technology leader. According to Google, by March of 2016, their fleet of driverless cars have driven 2.4 million kilometers in autonomous mode and only one, out the 13 accidents involving these vehicles, was their system's fault. Other companies that have shown prototypes include Ford, Mercedez-Benz, Uber, Nissan, Delphi and Bosch (KANE, 2016; INWOOD, 2016; DAVIES, 2016; DAVIES,

Figure 12 – Road detection method employed by Wang *et al.* (2004)



(a) Input image (b) Canny edge detection (c) Line estimation

Source: Wang *et al.* (2004).

2017; ETHERINGTON, 2017; MUOIO, 2016).

Early attempts to perform road detection, this thesis main theme, relied on detecting the boundaries of the road using cameras. For instance, in the work of Wallace (1985), the authors use a number of different edge detection methods, such as applying convolutions with the Difference of Gaussians (DoG) and the Roberts filter. After detecting the edges in the image, they estimate the road boundaries by fitting lines with the Hough transform (BALLARD, 1981). A more modern version of this approach can be seen in the work of Wang *et al.* (2004). They also start by finding edges in the image but after doing so the image is divided into horizontal sections and lines are detected in each section independently, as shown by Figure 12. Finally, the authors use a B-spline to model the road boundaries and hence they are able to model curved roads. A related approach is presented by Rasmussen (2004) and Kong *et al.* (2010), where they convolve the image with a bank of Gabor wavelet filters (LEE, 1996). These filters vary in orientation and size and by analyzing their responses it is possible to determine the main orientation θ of each pixel $\mathbf{p}$ and hence every pixel $\mathbf{p}$ defines a line. Now a voting scheme takes place, where each pixel $\mathbf{p}$ casts votes for every point along its line. The point with the most votes (i.e. the point where most of the lines cross) is the vanishing point of the road and the pixels that cast votes for the winning vanishing point are part of the road boundary. Although approaches that try to find the road boundaries can be useful in some situations, they assume that there is a clear delineation between the road and its surroundings which is not always the case. Nevertheless, a significant number of works rely on this assumption (ALY, 2014; PARAJULI *et al.*, 2013; BOTTAZZI *et al.*, 2013).

Many works, including this one, rely on machine learning techniques to detect the road. Despite the need of manually creating examples, machine learning-based approaches tend to produce better, more robust, results than the ones that try to directly detect the boundaries of the road. In these works the objective is to *learn* a mapping between

Figure 13 – Road detection results obtained by Alvarez and Lopez (2011) where an illuminant invariant color space was employed

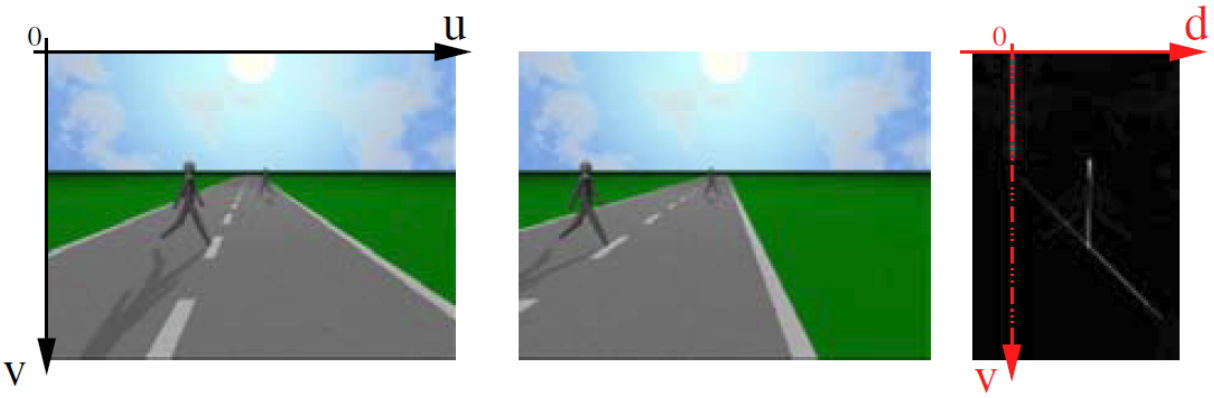
Source: Alvarez and Lopez (2011).

pixels (or image regions) and their class, i.e. road or non-road. An example of a work that uses a shallow model (MLP) is given by Shinzato *et al.* (2010), which has the following steps: (I) divide the input image into square regions (a grid); (II) extract statical features (e.g. mean, standard deviation and entropy) from such regions; (III) train a classifier to learn a mapping between the features and the correct class. After training, a new image can be classified by classifying each square region independently. In Zhou *et al.* (2010), the authors use the kernel-SVM model and the classification is made per pixel instead of regions. As the results of a per-pixel classification tend to be noisy, the authors use a set of morphological operators to improve the final result.

An important decision for those using shallow models is what features to use since the performance of such models is highly dependent on the features. Some works (ZHOU *et al.*, 2010) directly use the Red, Green, and Blue (RGB) color space as features while others (RAMSTROM; CHRISTENSEN, 2005; ROTARU *et al.*, 2008) try to use a color space that is less susceptible to variations in illumination. Adequately dealing with illumination is one of the main challenges of approaches that use a camera. For instance, the work from Sotelo *et al.* (2004) applied constraints to the road shape to be able to correctly classify shaded areas. Other works (ALVAREZ; LOPEZ, 2011; KIM *et al.*, 2011) aim to create a color space that is invariant to illumination changes. Among these, it is worth mentioning the work of Alvarez and Lopez (2011) that achieved significant qualitative results as can be seen in Figure 13. Texture-related features are used by Kong *et al.* (2010) and Lombardi *et al.* (2005) whereas the work of Yun *et al.* (2007) uses a combination of color and texture. Some papers use features estimated automatically through unsupervised learning methods. For example, the work of Kuhnl *et al.* (2011) uses the Slow Feature Analysis (SFA) technique to estimate lighting invariant features.

Several works use a stereo camera to detect obstacles and estimate the navigable region. These works take advantage of the geometric information provided by the combina-

Figure 14 – Stereo image pair and its respective “V” disparity map



Source: Caraffi *et al.* (2007).

tion of a stereo camera and a stereo method. The selection of which stereo method to use is an important one since they can compute a sparse or dense depth map with more or less accuracy. Also, due to its high computational cost, only around the year of 1997 dense stereo methods began to be used for robotic navigation. The work of Murray and Jennings (1997) uses the dense block matching stereo method to estimated a dense disparity map and uses this information for mapping and navigation. However, the robot's performance in such tasks was severely limited by the throughput of the stereo method, which was only two hertz with an image resolution of 180x120. As the work of Rankin *et al.* (2005) shows, most works using a stereo camera look for obstacles by searching for discontinuities and elevations in the depth map. Some works have as its first task the estimation of the surface in which the vehicle is situated. For this, the technique of generating the “V” disparity map is widely used (LABAYRADE *et al.*, 2002; BROGGI *et al.*, 2005; CARAFFI *et al.*, 2007). The “V” disparity map is essentially a lateral projection of the original depth map, its creation is done by transforming the original map into a new map so that the columns represent the disparity values in ascending order, the lines are kept in the same order and the intensity represents the amount of identical disparity values. Figure 14 shows a stereo image pair and its respective “V” map where the navigable surface can be seen as a diagonal line. Using simple heuristics it is also possible to estimate the location of obstacles, as was done in the work of Lima and Pereira (2010). One limitation of methods based on the “V” map is that, while the longitudinal profile of the navigable surface can be curved, the transversal profile must be linear. In addition, a roll angle between the sensor and the surface blurs the map, compromising the accuracy of the obstacle detection.

A slightly different approach to detect navigable regions using a stereo camera can be seem in the work of Konolige *et al.* (2008) and Happold and Ollis (2006). These works assume that the vehicle is standing on a geometric plane and use the RANdom SAmple Consensus (RANSAC) (FISCHLER; BOLLES, 1981) method to estimate the parameters of such plane. With the estimated plane it is possible to calculate the distance of any

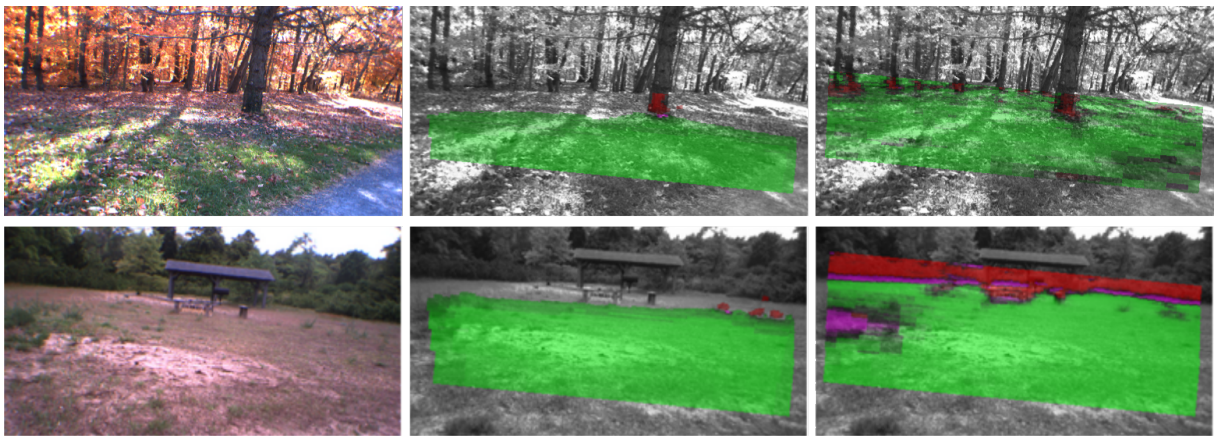
Figure 15 – Sample of the obstacle detection results obtained by Broggi *et al.* (2011) during the VisLab Intercontinental Autonomous Challenge (VIAC)

Source: Broggi *et al.* (2011).

point to the plane and, given a threshold, decide which ones are obstacles. Although this method can deal with a roll angle between the camera and the surface it is not suitable to represent curved or rugged regions. Hadsell *et al.* (2009) address this issue by using several planes (instead of just one) to approximate the terrain surface. In 2002, Talukder *et al.* (2002) proposed an obstacle definition together with a detection algorithm. The definition states that for two three-dimensional points be considered obstacles, there must be a significant difference in height and slope between them. This method has some advantages over those previously mentioned. It provides and applies a clear definition of obstacle and, in practice, it is able to deal with rough terrain and it was successfully used in the VisLab Intercontinental Autonomous Challenge (BROGGI *et al.*, 2011), as can be seen in Figure 15. But the method also has its disadvantages, for instance, it has a high computational cost and researchers usually employ approximations to use it in real time (BROGGI *et al.*, 2011; MARK *et al.*, 2007; SANTANA *et al.*, 2008). Moreover, it consists of a strictly binary detection making it unfit for further processing or as features to a machine learning model.

A common extension to works using a stereo camera is to use, not just the geometric information, but also the appearance one. In a stereo system, the appearance (color and texture) information is already available and associated with the geometric one. Naturally, some works try to take advantage of that and use the two sources of information complementarily. For instance, the geometric information provided by a stereo camera tend to degrade with distance, while the appearance does not suffer as much. Based on this observation, Happold and Ollis (2006) and Erkan *et al.* (2007) used a technique called *near-to-far* which has the following steps: (I) classify the lower (nearest) part of the image using geometric information; (II) use this classification as ground-truth to train an appearance-based classifier; (III) use the appearance-based classifier to classify far away regions where the geometric information is unreliable. Although a valuable strategy, the near-to-far technique has its flaws. First, it relies on very accurate geometry-based classification to train the appearance-based classifier, which is hard to attain in practice.

Figure 16 – Some qualitative results from the stereo-based near-to-far system proposed by Hadsell *et al.* (2009). Left: input image; middle: geometry-based classification; right: appearance-based classification using a convolutional neural network, which was trained using the geometry-based classification as samples. Green is traversable, red is obstacle, and pink is footline

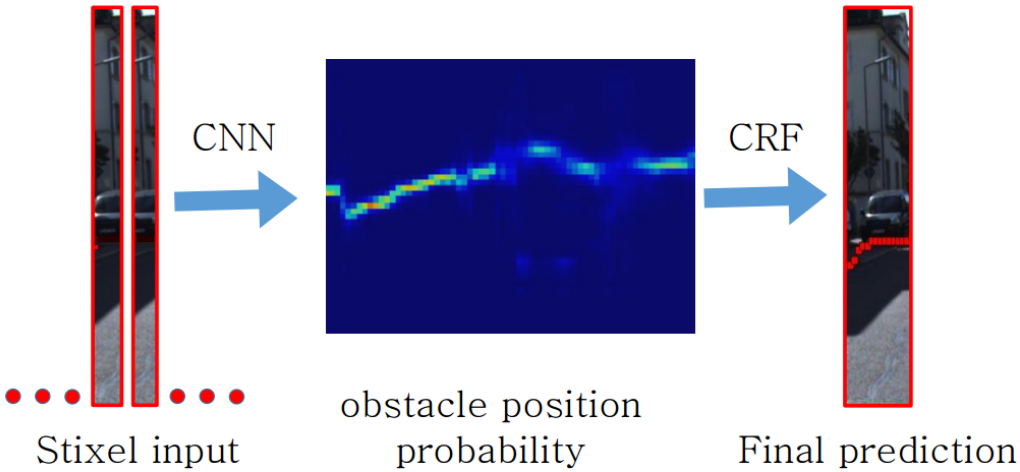


Source: Hadsell *et al.* (2009).

Furthermore, its final classification is purely based on appearance, effectively discarding part of the geometric information.

The work of Manso *et al.* (2010) uses a more elaborated methodology to combine geometry and appearance information. It creates two independent methods to detect navigable areas; one based on geometry and another based on appearance. The geometry-based method assumes that the navigable area is flat and calculates a transformation, the planar homography, between the pixels of one image of the stereo pair, to the other, as if they were all part of a navigable area. The pixels of the resulting image that match the ones of the image for which the transformation was calculated, belong to the navigable surface. Likewise, unmatched pixels (given a threshold) can be considered obstacles, moreover, the authors employ heuristic filters to remove false positives. The appearance-based detection uses supervised training to model the navigable region as color histograms. Finally, both methods are combined using a set of manually derived rules. The authors also provide qualitative evidence of the importance of using both sources of information, showing cases where one of them may fail. In the work of Guo *et al.* (2011), instead of creating two independent classifiers, the authors create a joint model using both sources of information. Specifically, they model the task using a Markov Random Field (MRF), which is a probabilistic graphical model defined by a set of random variables whose dependencies are encoded in an undirected graph. Their model also contemplates spatial and temporal dependencies, that is, it enforces smoothness between nearby regions and temporal coherence. In this instance, each random variable was related to a homogeneous group of pixels (a "superpixel") and its value to the estimated navigability of the corresponding region.

Figure 17 – Flowchart of the approach taken by Levi *et al.* (2015), where the input to the Convolutional Neural Network (CNN) is a vertical strip (called stixel) and the output is the position of the road/non-road boundary. Finally a one-dimensional Conditional Random Field (CRF) is used to smooth the results



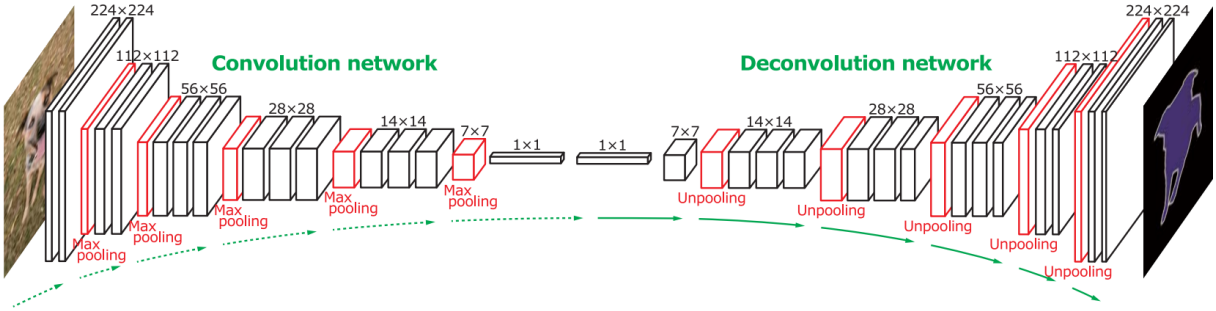
Source: Levi *et al.* (2015).

Since 2012, deep learning techniques have dominated visual tasks, including road detection. They tend to perform better, in terms of accuracy, and faster than their shallow counterparts. In fact, the first 5 entries in the KITTI road detection benchmark[1] use deep learning techniques, specifically, Convolutional Neural Networks (CNNs). Probably the earliest attempt to use CNNs for robot navigation happened in 2005 in a work from LeCun *et al.* (2005), where the first author, Yann LeCun, is founding father of CNNs. In this work the authors build an *end-to-end* vision-based obstacle avoidance system, where the input is a stereo image pair and the output is forward, left or right. After training the CNN with manually labeled data, a small robot was used to test the system and show the feasibility of such *end-to-end* system. Although an interesting technical result, this work has little practical application since the output of the network (left, right or forward) is unlike to precisely guide a robot or a vehicle. In 2009, again with the participation of Yann LeCun, a new work on robot navigation using deep learning was published (HADSELL *et al.*, 2009). This time it consisted of a near-to-far system, where the stereo geometric information is used to classify the nearby region and this classification is used as training samples to a CNN. Figure 16 shows some classification results from such work.

Since LeCun's initial works on robot navigation, CNNs have become commonplace in visual robotic navigation and a number of works using CNNs have been published (ALVAREZ *et al.*, 2012; MOHAN, 2014; BRUST *et al.*, 2015; LEVI *et al.*, 2015; OLIVEIRA *et al.*, 2016). These works differ in the specifics of the employed CNN architecture and also in the way the chosen architecture is used to solve the problem of road detection. For instance, in the work of Levi *et al.* (2015), the authors divide the image into vertical strips

[1] <http://www.cvlibs.net/datasets/kitti/eval_road.php>

Figure 18 – Convolutinal autoencoder architecture proposed by Noh *et al.* (2015). This architecture in divided into a convolutinal (encoder) part, which creates a more relevant low dimensional representation of the input and a deconvolutinal one, which upsamples this representation back to the original resolution and produces the segmentation result.



Source: Noh *et al.* (2015).

(called stixels) and try to detect the boundary between the navigable and non-navigable parts in each one of the strips, as can be seen in Figure 17. Given a vertical strip, the CNN task is to return the place (height) in which the boundary is present. Given that the output of the network is a position and the multimodal nature of the formulation, the authors model the probability of the boundary position as a piecewise-linear probability distribution and train the network to minimize its log-probability loss. Finally, a one-dimensional Conditional Random Field (CRF) is used to smooth the boundaries throughout the whole image.

A more direct approach can be seen in the work of Mohan (2014), where the input to the CNN is the whole image and the output its full road/non-road segmentation. As the classic convolution-pooling CNN architecture stage reduces the input size, in this work, the authors train deconvolution layers to upsample the resulting segmentation. In addition, they train a different CNN for each image region to account for the spatial prior of road regions (i.e. some regions of the image are more likely to be the road). Despite achieving good results this work has the inconvenient of requiring three separate training stages, one of each type of layer (convolutional, fully connected and deconvolutional). In this sense, the work from Oliveira *et al.* (2016) is more elegant. In it, an architecture commonly know as convolutional autoencoder (see Figure 18), first described in Noh *et al.* (2015), was used for the task of road detection. This architecture is similar to the one used by Mohan (2014), having convolutional (encoder) and deconvolutional (decoder) parts, but all layers are trained jointly in a single training procedure. More importantly, its results are more accurate than the ones from Mohan (2014). A shared characteristic of recent CNN-based road detection works is that none of them use stereo/geometric information. Possible reasons include: (I) the high computational cost of stereo methods; (II) small improvements in accuracy by doing so and (III) the already good-enough results obtained

by using CNNs.

3.1 Final Considerations

In this chapter, we have presented a brief history of the development of autonomous vehicles prototypes and a set of related works about road detection that we found to be the most relevant. Regarding the history of autonomous vehicles, we must highlight the DARPA Grand Challenge competitions that have put autonomous vehicles in evidence and significantly contributed to the development of the field. The related works started with the initial attempts to perform road detection using a monocular camera. Theses initial works employed a manually created set of rules to detect the road boundaries in the image. Machine learning-based methods come after and were most likely an advancement in terms of accuracy and robustness. We have also presented works that use stereo vision, these works tend to look for discontinuities and elevations in the depth map to detect obstacles or try to model the navigable surface using a geometric model (e.g. plane). Finally, we have presented works that use deep learning and are currently the state-of-the-art in the task. Some works were highly influential in this thesis: we use the same RANSAC method as Konolige *et al.* (2008) to estimate plane parameters in our stereo-based road detector and the deep architecture proposed by Noh *et al.* (2015) to test our proposed neural network layer.

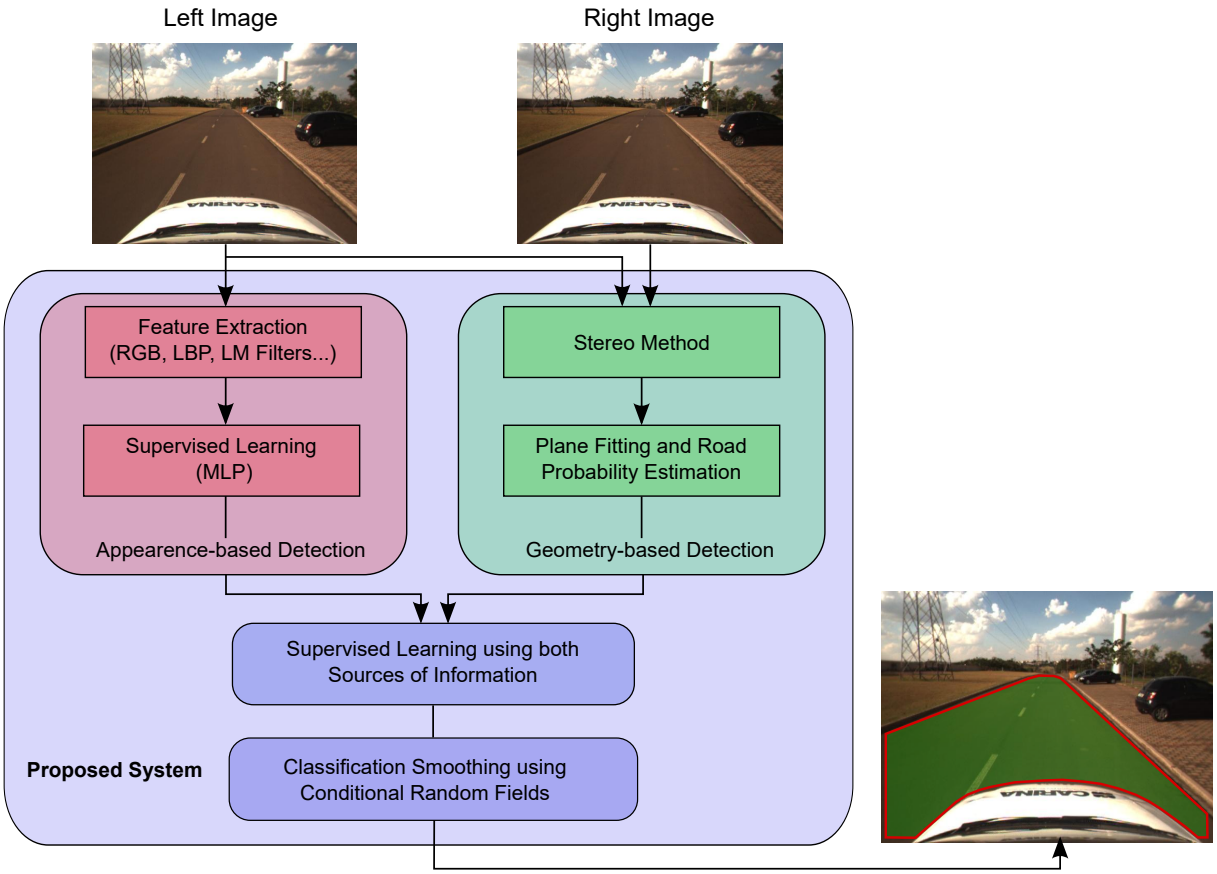
CHAPTER

4

CLASSICAL APPROACH

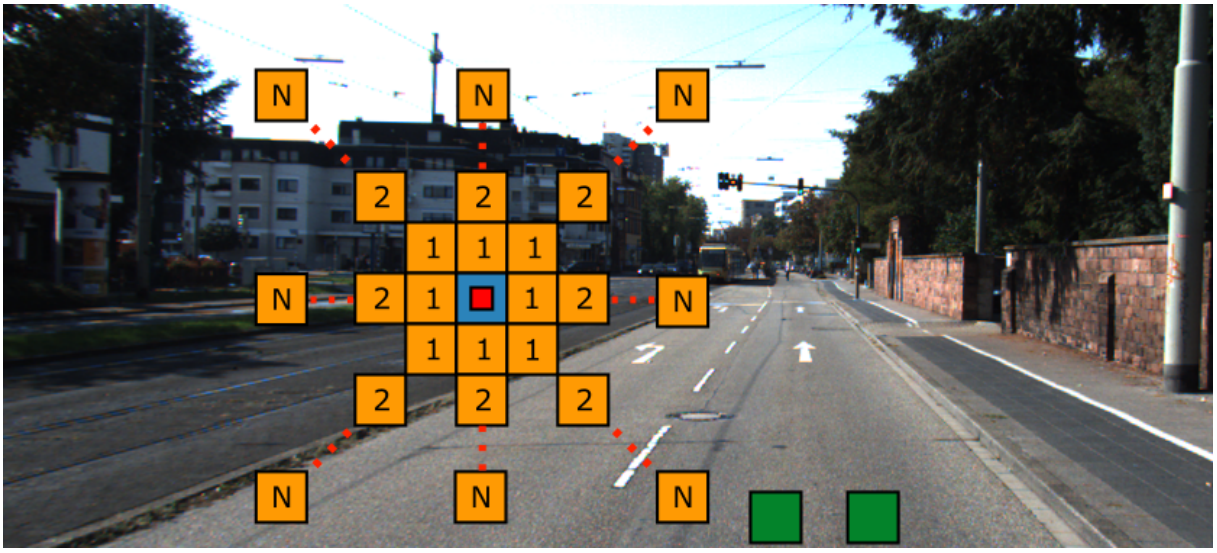
In this chapter, we describe a road detection system that uses a stereo camera as its source of information. In the core of this detection system are traditional/shallow machine learning models, we hence have chosen to call it a "classical" system or approach.

Figure 19 – Block diagram of the proposed system, where the input is a pair of rectified stereo images and the output is an image segmented into road/non-road areas



Source: Elaborated by the author.

Figure 20 – Illustration of the proposed block scheme; the classification block is show in red, the contextual blocks in orange, the possible support block in blue and the road blocks in green



Source: Elaborated by the author.

4.1 Overview

A simplified flowchart of the proposed system can be seen in Figure 19. The input consists of a pair of rectified stereo images; these are provided by a stereo camera. The reference image (left) is used by the appearance-based detection module. The geometry module uses both images so it can perform stereo matching and hence extract geometric information. The appearance-based detection module uses a trained model to independently classify small regions or patches of the image into road or non-road, where the model's inputs are features extracted from the patch. The model's training is supervised; hence a manually created dataset of classified images is required for training. The geometry-based detection module starts by performing pixel-level correspondence between the left and right images (stereo matching), resulting in geometrical information about the environment. We then model the navigable area with a plane, using the RANSAC method to robustly estimate the plane parameters. By calculating the distance from every point to the plane, we have an estimation of whether a point is part of the road or not and, as a result, each image pixel of the reference image is given a probability of belonging to the road area. The results of the geometry-based module are used as features to augment the appearance-based detection classifier. Finally, the segmentation is globally smoothed using a Conditional Random Field (CRF) model. The entire methodology is explained and justified in detail in the following Sections.

Algorithm 1 – Feature Concatenation

$\mathbf{v}_{final} \leftarrow \varnothing$ ▷ Empty vector
$\mathbf{v}_{final} \leftarrow \mathbf{v}_{final} \oplus \mathbf{v}_{class}$ ▷ $\oplus$: Concatenation
for $i \leftarrow 1$ **to** $radius \times 8$ **do**
    $\mathbf{v}_{final} \leftarrow \mathbf{v}_{final} \oplus \mathbf{v}^{i}_{context}$
end for
if $size(class) \neq size(context)$ **then**
    $\mathbf{v}_{final} \leftarrow \mathbf{v}_{final} \oplus \mathbf{v}_{support}$
end if
$\mathbf{v}_{final} \leftarrow \mathbf{v}_{final} \oplus (\mathbf{v}^{1}_{road} - \mathbf{v}_{class})$
$\mathbf{v}_{final} \leftarrow \mathbf{v}_{final} \oplus (\mathbf{v}^{2}_{road} - \mathbf{v}_{class})$

4.2 Appearance-based Detection

This module extensively use rectangular block/patches as shown in Figure 20. These blocks are divided into three categories: classification blocks, contextual blocks and road blocks. Classification blocks are the ones whose pixels are classified, while contextual and road blocks are auxiliary and delimit regions from which features are extracted. To classify a single image region, or classification block, one should extract features from the block itself, from its respective contextual blocks and from the road blocks. All these features are pre-processed and concatenated into a single final vector $\mathbf{v}_{final} \in \mathbb{R}^N$ that is fed to the classifier. The output of the classifier, road or non-road, is attributed to every pixel of the classification block. To classify an entire image, this task should be repeated for every image region, or classification block, and, since we classify every pixel in a classification block, its vertical and horizontal stride is always equal to its vertical and horizontal size.

The features employed in this work do not provide spatial information, i.e. they do not make distinction between pixels positions within a block, hence we use what we call "contextual blocks" to provide information about the surroundings of the classification blocks. The first contextual blocks are positioned in the direct neighborhood of a reference block according to the eight connected scheme and further blocks are aligned in a "star" shape pattern. The reference block is the classification block itself if the classification block and the contextual blocks have the same size. Otherwise it consists of an additional block, called support block, centered on the classification block and with the same size as the contextual blocks. The number of contextual blocks for a classification one is given by their "radius". For instance, a radius of one yields 8 contextual blocks while a radius of 2 yields 16. The feature vector $\mathbf{v}^{i}_{context}$ of each one of the contextual blocks, and the possible support one $\mathbf{v}_{support}$, are concatenated into the final feature vector $\mathbf{v}_{final}$.

Finally, road blocks are positioned in the center bottom part of the image and they provide a frame relative notion of the road appearance. The feature vector of each road

Table 1 – Image features selection

Feature		Dim.
RGB	Mean and std. dev. of each channel	6
Grayscale	Mean and std. dev.	2
Entropy	Mean and std. dev.	2
LBP	Normalized LBP histogram (4-connected)	16
LM Filters 1	Mean and std. dev. of filter responses	30
LM Filters 2	Normalized histogram of the max. responses	15
	Total	71

Source: Elaborated by the author.

block $\mathbf{v}^i_{road}$ is subtracted from the classification block feature vector $\mathbf{v}_{class}$ and concatenated into $\mathbf{v}_{final}$. The subtraction is made to directly provide the classifier a similarity notion of the block being classified and a supposed road region. We opted for using two small road blocks instead of a larger one, as it is usually done, to minimize the effect of lane markings in the road blocks features. The feature concatenation is summarized in Algorithm 1.

4.2.1 Image Features

We decided to use some simple color and texture/structure features. We gave preference for fast (low computational cost) and low dimensional features. As small image regions are being classified, there is no need for complex features, such as those employed for object recognition (e.g. Histogram of Oriented Gradients). Furthermore, as we will be using a parametric classifier, a low-dimensional feature vector is desirable since it can improve generalization.

Table 1 presents the selected image features. Entropy, Local Binary Patterns (LBP) and Leung-Malik (LM) (LEUNG; MALIK, 2001) filters responses features are generated based on the grayscale image. The entropy is calculated using a circular support region with a radius of 5 pixels. For the LBP descriptor, we chose to use four neighbors instead of the usual eight reducing its histogram dimensionally from 256 to 16. We employed a subset of the original LM filter bank consisting of 6 edge, 6 bar, 1 Gaussian and 2 Laplacian of Gaussian filters, with a 19×19 pixel support, $\sqrt{2}$ scale for oriented and blob filters and 6 orientations.

A spatial prior, in the form of the position of the classification block, is also included in our final feature vector. Preliminary tests suggested that it is preferred to input it encoded as a one-hot bit vector instead of a floating point. Intuitively, this encoding may facilitate the learning of strong priors in parametric models. Concretely we normalized each classification block coordinate, discretize it in 11 parts and represent each discretized

coordinate as an 11 bins one-hot bit vector. Therefore the dimensionality of the spatial prior feature is $dim(\mathbf{v}_{spatial}) = 22$, 11 for each coordinate. The exact number of bins should make a small difference in performance as long as it is not too small (e.g. < 5), compromising its discriminative power, or too large (e.g. > 100), significantly increasing the model complexity in parametric models.

If we assume the use of the additional support block, the dimensionality of the final feature vector is given by:

$$dim(\mathbf{v}_{final}) = dim(\mathbf{v}_{class}) + dim(\mathbf{v}_{support}) + radius \times 8 \times dim(\mathbf{v}_{context}) + 2 \times dim(\mathbf{v}_{road}) + dim(\mathbf{v}_{spatial}), \quad (4.1)$$

where the function dim returns the dimensionality of the input vector. It should be noted that, for this work $dim(\mathbf{v}_{class}) = dim(\mathbf{v}_{context}) = dim(\mathbf{v}_{road})$ and if we consider all features, they are all equal to 71 (Table 1).

4.2.2 Classifier

We chose to use a standard Multilayer Perceptron (MLP) neural network, which is a parametric non-linear model. MLPs present a reasonable classification performance in a wide range of tasks and are easily parallelizable to exploit the processing power of Graphics Processing Units (GPUs) and multi-core systems and, as it is a parametric model, its prediction computational cost does not depend on the training procedure (unlike SVMs, for instance).

Our model consists of one hidden layer with Rectified Linear (ReLU) activations functions and an output layer with the sigmoid activation function. We used the cross entropy loss function, therefore only one output neuron is used for the binary classification task. Formally, given that the feature vector $\mathbf{v}_{final}$ is a column vector the prediction is given by:

$$g(\mathbf{v}_{final}) = \sigma(\mathbf{W}_o \cdot \psi(\mathbf{W}_h \cdot \mathbf{v}_{final})), \quad (4.2)$$

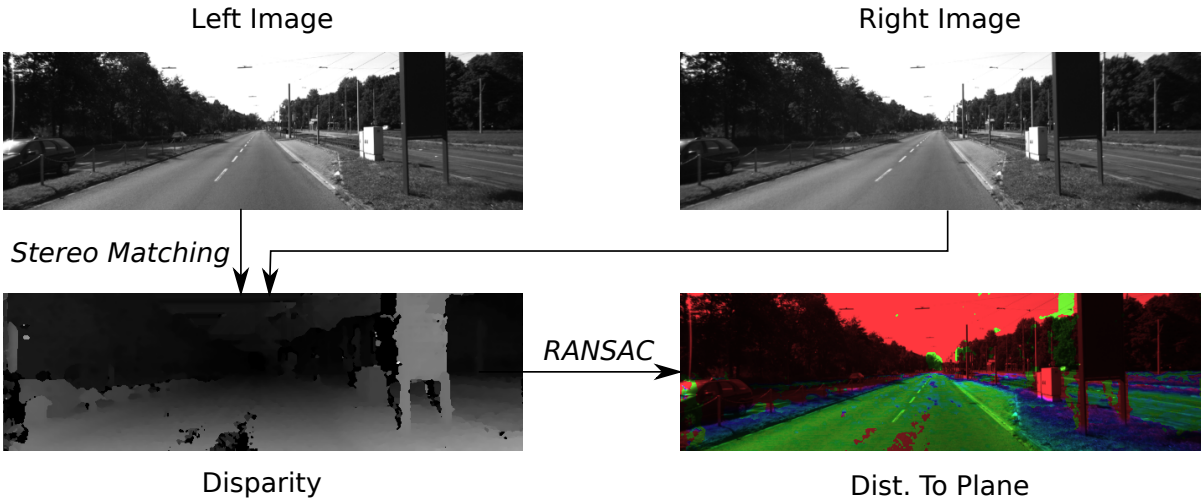
where $\mathbf{W}_h$ and $\mathbf{W}_o$ are the weight matrices of the hidden and output layer respectively (each row stores the weights of a neuron), ψ is the ReLU function and σ is the sigmoid function. Finally, the output of the model is thresholded according to:

$$L = \begin{cases} \text{Road} & \text{if} \quad g(\mathbf{v}_{final}) > 0.5 \\ \text{Non-road} & \text{if} \quad g(\mathbf{v}_{final}) \leq 0.5 \end{cases}, \quad (4.3)$$

where L is the label of the referent classification block.

For regularization, we limit the Euclidean norm of the MLP weights (parameters), the maximum value is chosen per layer and it is applied individually to the weights

Figure 21 – Illustration of the initial steps of the geometry-based detection. The image in the bottom right represents, with color, the distance of every point to the calculated plane. From green, meaning near to the plane, to red, meaning far from the plane.



Source: Elaborated by the author.

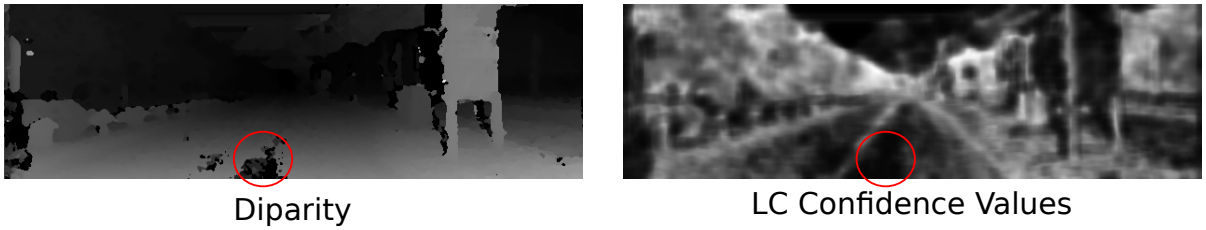
corresponding to a single neuron (output dimension of the layer). When the norm exceeds the limit, it is scaled down to have exactly the limit value. The training is done using mini-batch stochastic gradient descent. The training is finished after a number, here called of "patience", of epochs without any improvement in the accuracy of the validation set.

One drawback of MLPs is their large number of hyperparameters. To tackle this issue, before every training, we use a small subset of the training and validation sets to perform a hyperparameter optimization. For this optimization procedure we use the particle swarm optimization (PSO) algorithm and optimize the following parameters: number of neurons, learning rate, hidden layer maximum norm and output layer maximum norm.

4.3 Geometry-based Road Detection

Given a pair of rectified images provided by a stereo camera, we use the semi-global stereo method proposed by Hirschmuller (2005) to obtain a disparity map. The disparity map is then transformed into a point cloud using the intrinsics parameters of the camera. We proceed by fitting a plane in the point cloud using the robust fitting algorithm RANSAC (FISCHLER; BOLLES, 1981). The idea of fitting a plane is motivated by the fact that usually most points belong to the road, and the road is approximatively plane. Therefore, the fitted plane tends to represent the road. Now, with a plane fitted, it is possible to calculate the distance of every point in the point cloud to the plane. Figure 21 shows an image with color-coded distances along with the other initial steps necessary to produce such results.

Figure 22 – Disparity map and its respective matching confidence values. The red circles show how the confidence is lower (darker) in regions containing matching errors.



Source: Elaborated by the author.

A visible problem of this approach is that there are stereo matching errors, that is, errors in the disparity map. Such errors affect the calculated distances, making the resulting detection inviable to use in practice. To remedy this problem, we modified the used stereo matching implementation to produce a "stereo confidence value" for every pixel of the disparity map. We calculate this confidence value using the local curve (LC) (WEDEL *et al.*, 2009) metric:

$$LC = \frac{\max\left(C_+, C_-\right) - C_{min}}{\gamma}, \tag{4.4}$$

where C_+ and C_- are the adjacent values to C_{min}, the minimal cost location, and γ is a normalization factor. Figure 22 shows how the confidence values correlate with disparity errors.

We then convert the confidence values to a calibrated probability using the Bayes theorem:

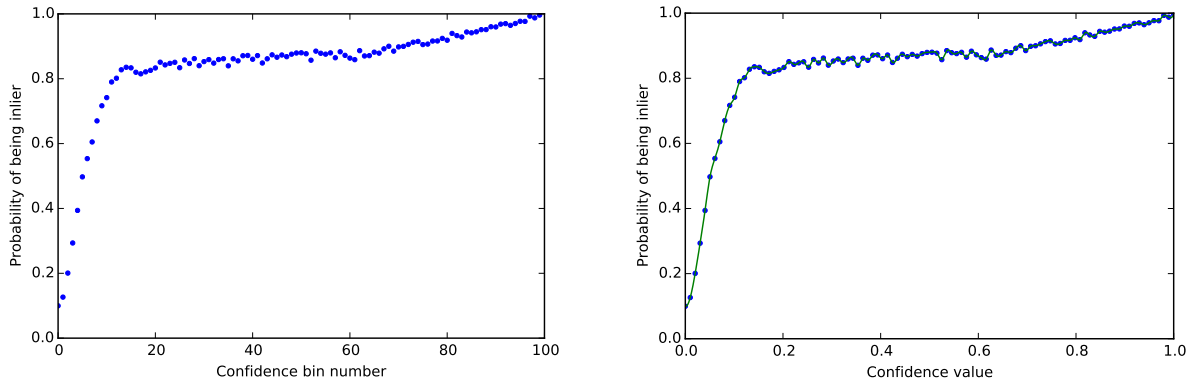
$$p(i \mid c) = \frac{p(c \mid i) \cdot p(i)}{p(c \mid i) \cdot p(i) + p(c \mid \neg i) \cdot p(\neg i)}, \tag{4.5}$$

where $p(i \mid c)$ is the probability of an inlier, i.e. a true disparity, given a confidence value c. To calculate all the values of the formula, we discretized the confidence values into 100 parts or bins, and used the stereo ground-truth from the KITTI vision benchmark suite (FRITSCH *et al.*, 2013). In this benchmark, the true disparity map is provided (using a LIDAR sensor), we hence can check whether or not a disparity value is an inlier and its confidence value. After calculating $p(i \mid c)$ for each one of the 100 confidence values, we interpolate those using a B-spline so that we can extract continuous probabilities. Figure 23 shows both the discretized and interpolated values for $p(i \mid c)$.

We now assume that the probability of a point being part of the road is inversely proportional to its distance to the plane, given that the point is an inlier. More precisely:

$$p(r \mid i) = \tau \frac{1}{d}, \tag{4.6}$$

Figure 23 – An illustration of the calculated $p(i \mid c)$ graph

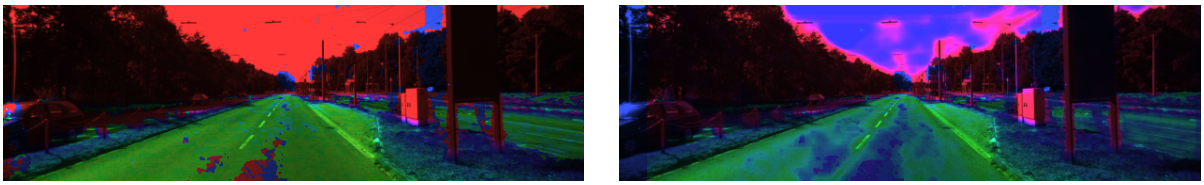


(a) Discrete version, used for calculating the values

(b) Continuous interpolated counterpart, used in practice

Source: Research data.

Figure 24 – Sample images of using the confidence values in the final results. While the image (a) assumes that every point is an inlier, the image (b) uses the confidences values, therefore, correcting the bottom part of the image. Both images range from green, high probability of being road, to red, low probability of being road.



(a) $p(r \mid i)$

(b) $p(r)$

Source: Elaborated by the author.

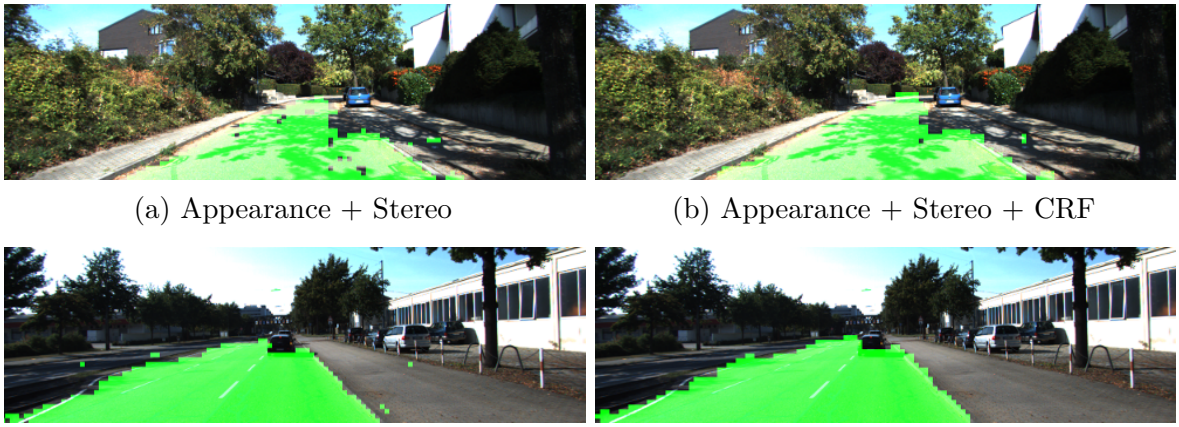
where $p(r \mid i)$ is the probability of a point being road given that it is an inlier, d is the distance to the plane and τ is a scaling parameter. We have chosen to use $\tau = 0.05$ in our experiments. Now that we have the probability of a point being road given that it is an inlier $p(r \mid i)$ and also the probability of a point being an inlier given a confidence value $p(i \mid c)$, we finally can calculate the probability of a point being a road, that is $p(r) = p(r \mid i) \cdot p(i \mid c)$. The final results can be seen in Figure 24.

The final results $p(r)$ are used as features to the same classifier of the appearence-based road detection module. The mean of $p(r)$ is added as a feature to each one of the blocks employed.

4.4 Classification Smoothing

So far, regions are classified independently, but as the environment is continuous, there are spatial dependencies between such regions. To model such dependencies a suitable

Figure 25 – Classification results with and without using the CRF model



(a) Appearance + Stereo (b) Appearance + Stereo + CRF

(c) Appearance + Stereo (d) Appearance + Stereo + CRF

Source: Elaborated by the author.

mathematical model is required, two models are popular in the literature: Markov Random Field (MRF) and Conditional Random Field (CRF). The first is a generative model, that is, it models the joint probability distribution of the classes and features. The CRF model, in contrast, models the probability of the classes given features directly, such model is called discriminative. As the work of (LAIBLE *et al.*, 2013) shows, the Conditional Random Field model is more appropriate for road classification.

Being $\mathbf{y}$ the setting (classification) of the whole image and $\mathbf{x}$ the corresponding features. The classification can be defined as the search by the setting $\mathbf{y}^*$ that maximizes the probability $p(\mathbf{y}|\mathbf{x})$ given the observed features $\mathbf{x}$. Since the regions from the previous module were classified independently, we have:

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{M} p(y_i|x_i), \tag{4.7}$$

where M is the number of regions and the probabilities $p(y_i|x_i)$ are given by the MLP classifier. Alternatively we use the CRF model:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(-\lambda \sum_{i=1}^{M} \phi_i(y_i, x_i) - \sum_{(i,j)\in\mathcal{N}} \psi_{i,j}(y_i, y_j, x_i, x_j)\right) \tag{4.8}$$

$$\phi_i(y_i, x_i) = -\log(p(y_i|x_i)) \tag{4.9}$$

$$\psi_{i,j}(y_i, y_j, x_i, x_j) = \mathbf{1}_{\{y_i \neq y_j\}} \exp\left(-\beta(x_i - x_j)^2\right), \tag{4.10}$$

where Z is the normalization function (or partition function), λ controls the influence of each term and $\mathcal{N}$ is the set of neighbors. The factor $\phi_i(y_i, x_i)$ models the feature dependent component (also called data term) where $p(y_i|x_i)$ is the classifier output, and $\psi_{i,j}(y_i, y_j, x_i, x_j)$ models the class distribution (also called smooth term). The idea behind

equation (4.10) is that, the probability of two neighbor regions belonging to the same class is high, but if they belong to different classes their feature vector (x) must also differ. The feature vector used here is the same as the previous step. Exact inference in the proposed model is computationally intractable, we hence perform it in an approximate manner using Gibbs sampling. Figure 25 show samples of the results obtained using a CRF. It is possible to see that small errors are smoothed out by using the CRF model.

4.5 Tests and Evaluation

4.5.1 Dataset and Setup

To evaluate our approach, we made use of the KITTI Vision Benchmark Suite (FRITSCH *et al.*, 2013). Specifically, we use the road detection benchmark, which provides 289 annotated images for training and 290 test images. Both sets are divided into three categories: urban unmarked (UU), urban marked (UM) and urban multiple marked lanes (UMM). Methods are ranked according to their pixel-wise maximum F-measure on the Bird's-eye view (BEV) space. The benchmark further provides laser points (Velodyne data), stereo images and GPS data. In our work, only the monocular and/or stereo images are used and we do not make distinction between the three road categories.

To evaluate each component of our system and to select adequate hyperparameters, we divide the 289 annotated images into a set of training/validation containing 260 images and a set for testing containing 29 images. All results reported in this chapter, excluding our benchmark submissions, are referent to these 29 images. The evaluations are performed in the same way as the benchmark server, i.e. the prediction and the ground truth images are both converted into BEV space and are compared pixel-wise.

The appearance-based road detection module was implemented using the Python-based SciPy software ecosystem and scikit-image library for feature extraction. The geometry-based detection was implemented in *C++*, using the Point Cloud Library (PCL) (RUSU; COUSINS, 2011) and the OpenCV[1] library. We use the MLP GPU implementation provided by the Pylearn2 (GOODFELLOW *et al.*, 2013) library, conduct the PSO hyperparameter optimization using the Optunity (CLAESEN *et al.*, 2014) library and implemented the CRF using PyStruct (MÜLLER; BEHNKE, 2014). The tests were conducted on a machine equipped with an Intel Core i7-4930K, 64GB RAM and an NVIDIA Titan X. The GPU was utilized only for model training and testing, the rest of the system runs on a single core.

For every test and the benchmark submission, we fixed the blocks size at 10×10 for the classification blocks and 20×20 for the contextual blocks, hence we always use

[1] http://opencv.org/

Table 2 – Hyperparameter Search Setup

Parameter	Value
N. Iterations	10
Particles	10
N. Hidden Neurons	$(16, 2000)$
Learning Rate	$(0.001, 0.5)$
Max. Norm Hidden	$(0.5, 5)$
Max. Norm Output	$(0.5, 5)$

Source: Elaborated by the author.

the additional support block. We believe that those sizes yield a good compromise of computational cost, discriminative power and classification granularity.

4.5.2 Training Scheme

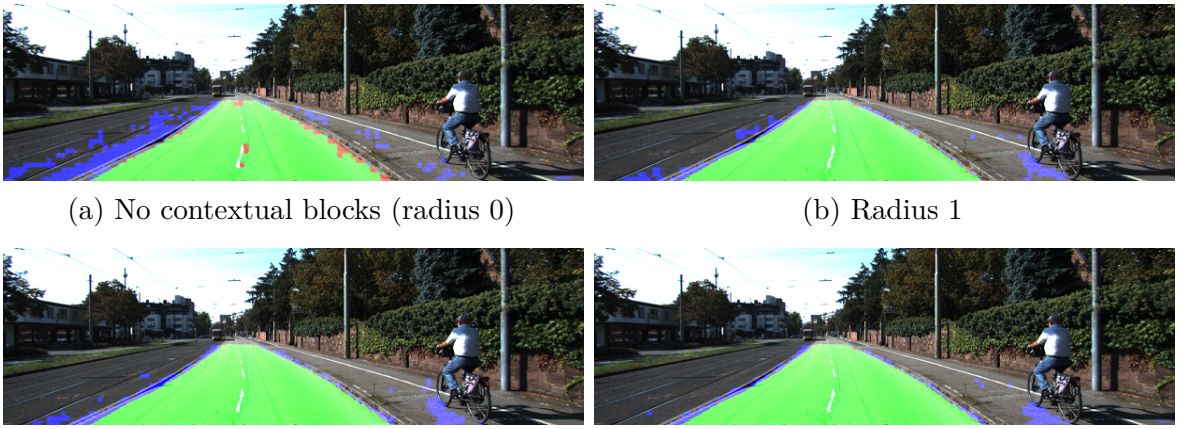
To generate the features vectors (samples) for training, we used only classification blocks whose ground truth pixels are all of the same class, excluding, therefore, ambiguous cases. We adequately pad images to accommodate the selected block sizes and contextual blocks in order for the classification blocks to cover the full original image. As the top 150 lines of every image contain only negative examples and are not considered in the BEV space evaluation, we ignore this region when generating the feature vectors for training. This measure reduces the training time and helps to improve the class balance.

The samples extracted from the 260 images that are selected for training/validation are randomly split into 70% training and 30% validation. Each of these datasets is further subsampled at 20% for the hyperparameters search, where the validation set is used for early stopping and for the hyperparameter selection. Once the best hyperparameters are established, the training proceeds by using the initial $70 - 30$ split, where the validation set is used only for early stopping. All samples are standardized feature-wise based on the training dataset. The hyperparameter search configuration is presented in Table 2. We use 10 particles and 10 iterations resulting in 100 training procedures. Further MLP parameters are the 100 mini-batch size, 0.9 momentum and 30 patience.

4.5.3 Evaluation of Contextual and Road Blocks

We initially evaluate the effects of using contextual blocks and their radius parameter using all image features. Table 3 shows the results when varying the radius on the 29 testing images, “Pre.” stands for precision and “Rec.” for recall. A radius of 0 means that no contextual block is in use. The results show a substantial increase in the F-measure from no contextual block use (radius 0) to radius 1 and further radius increases yield a

Figure 26 – Classification results using different radius parameter values where green represents true positive, red false negative and blue false positive



(a) No contextual blocks (radius 0) (b) Radius 1

(c) Radius 2 (d) Radius 3

Source: Elaborated by the author.

Table 3 – Contextual Blocks Radius Evaluation (in %)

Radius	F-measure	Accuracy	Pre.	Rec.
0	83.7	87.8	87.8	79.9
1	86.3	89.4	87.5	85.0
2	87.3	90.4	89.6	85.2
3	88.2	91.0	90.2	86.2

Source: Research data.

small but consistent improvement. This effect is also clearly visible in the classification results as shown in Figure 26. The image classified using no contextual blocks presents a significant amount of false positive and false negative pixels. With a radius of 1, all false negative pixels are removed and the number of false positives is reduced. The number of false negatives continues to decrease until radius 3, when the left side of the resulting image is almost clear of false positives.

These results highlight the validity of our contextual blocks approach and despite the higher dimensionality of the feature vector, the classifier was able to take advantage of the additional information. We did not test radiuses larger than 3 due to hardware constraints (especially the working memory), nevertheless the benefit of larger radiuses is expected to fade and not compensate the additional computational cost.

We also evaluate how the road blocks affect the performance. For that purpose, we removed the road blocks features while maintaining the best radius parameter previous obtained (3) and all image features. Table 4 shows the results where the column "Diff." refers to the difference in F-measure when using all blocks. The removal of the road blocks has a minor effect on the performance, affecting it less than a single decrease in the

Table 4 – Road Block Evaluation (in %)

Blocks	F-measure	Accuracy	Pre.	Rec.	Diff.
All	88.2	91.0	90.2	86.2	0.0
No Road	87.9	90.8	90.1	85.9	-0.3

Source: Research data.

Table 5 – Features Evaluation (in %)

Feature Subset	F-measure	Accuracy	Pre.	Rec.	Diff.
All	88.2	91.0	90.2	86.2	0.0
No RGB	87.7	90.7	90.4	85.2	-0.5
No Gray	88.1	90.9	90.4	85.8	-0.1
No Entropy	87.7	90.7	90.3	85.3	-0.5
No LBP	87.3	90.4	90.2	84.6	-0.9
No LM 1	88.5	91.2	90.4	86.8	+0.3
No LM 2	87.0	90.0	88.6	85.5	-1.2
No Spatial	88.0	90.8	90.4	85.7	-0.2

Source: Research data.

contextual blocks radius. We can, therefore, conclude that, for this dataset, our method is robust and does not depend on the usage of road blocks. However in datasets where the change in road appearance between training and test sets is more enunciated, these blocks could play a major role in helping with generalization.

4.5.4 Features Evaluation

Using the best radius deducted from previous experiments, we evaluated the contribution of each feature subset. To do so, we removed each feature subset and evaluated the performance on the 29 test images. The results are show in Table 5. These results show that the LBP and LM 2 texture features provided the most significant contribution despite LBP using the unusual 4 neighbor parameter and the small subset of filter selected for the LM features. The LM 1 features did not provide benefit and, in fact, their removal resulted in a 0.3 F-measure increase. Considering that a non-linear parametric model is employed, we suspect that the unique information content of the LM1 features did not compensate for their relative high dimensionality (750 considering all blocks). The RGB features provided a reasonable contribution while the gray features made little difference, probably due to their redundancy with the RGB ones. The spatial prior showed of little importance for our method, which is expected since we use a large contextual support. Methods with smaller or no contextual support would greatly benefit from using a spatial prior. Overall, the method is robust to the feature selection as no feature subset removal

Table 6 – Urban Road KITTI Benchmark Results (in %)

Method	MaxF	Pre.	Rec.	FPR	FNR	Runtime
DNN	93.43	95.09	91.82	2.61	8.18	2s
HIM	90.64	91.62	89.68	4.52	10.32	7s
App. + Stereo + CRF	90.13	89.74	90.51	5.33	9.73	2.7s
App. + Stereo	89.88	89.52	90.24	5.82	9.76	2.5s
NNP	89.68	89.67	89.68	5.69	10.32	5s
NED	89.12	85.80	92.71	8.45	7.29	1s
App. only	88.97	89.50	88.44	5.71	11.56	2s
FusedCRF	88.25	83.62	93.44	10.08	6.56	2s
ProbBoost	87.78	86.59	89.01	7.60	10.99	150s
SPRAY	87.09	87.10	87.08	7.10	12.92	0.04s
RES3D-Velo	86.58	82.63	90.92	10.53	9.08	0.36s

Source: <http://www.cvlibs.net/datasets/kitti/eval_road.php> (31 Mar. 2015)

reduced the F-measure to the level of not using contextual blocks.

4.6 Benchmark Submission

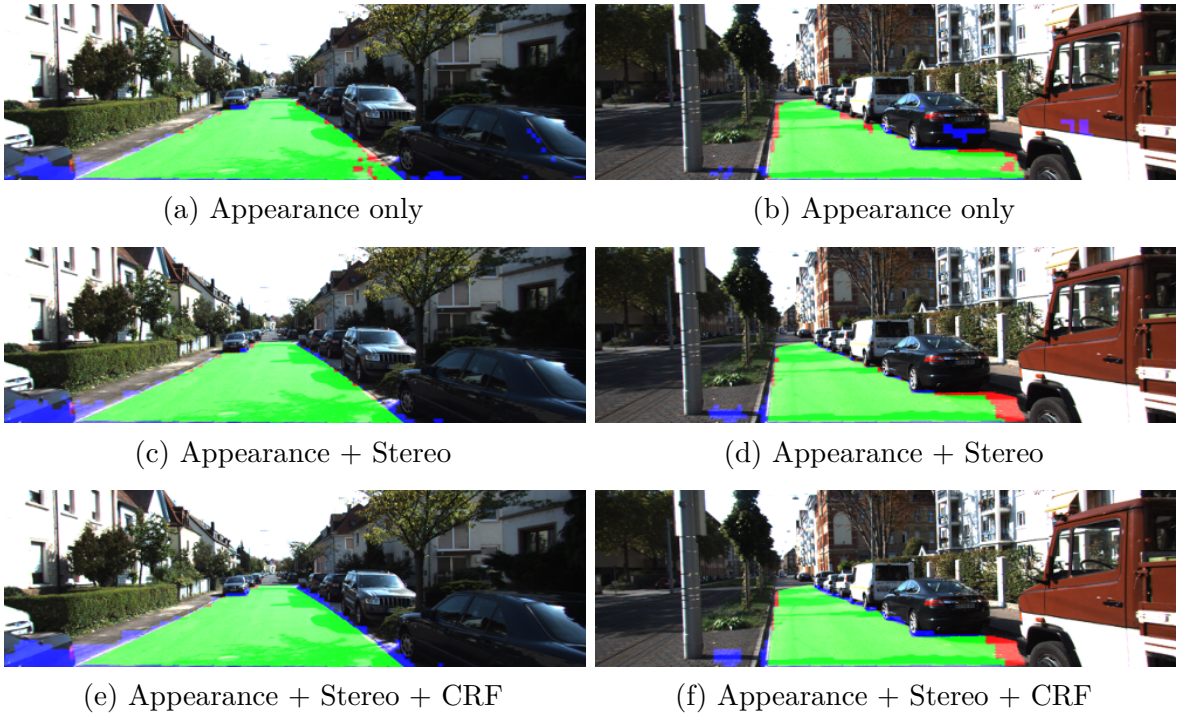
To compare our method with others, we submitted our method results, under the name **CB**, to the road detection KITTI Benchmark[2]. We have submitted the best configuration according to the performed experiments, that is, using all but the LM 1 features and a radius of 3 for the contextual blocks. We also submitted the results of using the additional stereo features and with the CRF model. Table 6 presents the first eleventh benchmark results in the *Urban Road* category which includes all road images types (UU, UM and UMM), "MaxF" stands for maximum f-measure, "FPR" for false positive rate and "FNR" for false negative rate. Our best submission achieved the third best score out of 31 participants, including the ones taking advantage of LIDAR (FusedCRF and RES3D-Velo).

The first two methods (DNN and HIM) uses a global context (takes the whole image into consideration) which may explain their higher scores. The next two methods are yet to be referenced, the only information available tells that the NNP method uses stereo vision (plane fitting) and NED uses some form of CNN. The fastest method in the benchmark is the SPRAY method. As our work, this method focuses on providing the classifier contextual cues in an efficient way. All methods scoring better than ours uses some form of parallel processing and could not achieve real-time.

Figure 27 shows a comparison of the three summited methods. The addition of the stereo features removed some of the false positives, especially those in high areas. The

[2] <http://www.cvlibs.net/datasets/kitti/index.php>

Figure 27 – Comparison of each stage of the proposed system



(a) Appearance only (b) Appearance only

(c) Appearance + Stereo (d) Appearance + Stereo

(e) Appearance + Stereo + CRF (f) Appearance + Stereo + CRF

Source: Elaborated by the author.

CRF results show of little importance, as it is confirmed by the numeric results of Table 6. Although it smooths out errors it may also smooth correctly classified regions, therefore evening out its benefits. One reason for this is that the appearance-based detector uses a large contextual region already taking into account the influence of nearby regions.

4.7 Final Considerations

In this chapter, we have presented our classical approach for road detection. This approach takes as input the stereo image pair and outputs the road/non-road segmented reference (left) image. At the core of the system is a block scheme to efficiently aggregate contextual information, which we believe to be our main contribution. The results show that the use of such scheme significantly increases the accuracy and that this increase scales adequately with the radius parameter. The appearance-based module run-time depends mostly on the images features selection, while the block scheme itself have a low computational cost since their features can be pre-calculated and simply concatenated afterwards. One advantage of our method is its simplicity, especially when compared to other road detection works (VITOR *et al.*, 2014; PASSANI *et al.*, 2014). We provide a set of visual features selection that seems to be adequate for road detection and whose implementation can be highly optimized. We also presented other details such as the training scheme and hyperparameters search that may have contributed to the method

performance.

Despite encouraging results, our method has some limitations. Global features are unpractical to include in the appearance-based module due to padding requirements and, even with a large radius, the whole image can not be considered. The presented implementation is not optimized and, although it might be optimized for real-time purposes, that would require a lot of work. Compared to deep learning methods (MOHAN, 2014), our method has the disadvantage of requiring a selection of hand-crafted features, which is mostly intuitive since it is not possible to evaluate all combinations of image features present in the literature. Finally, the use of geometric information and the CRF model are hardly justified in terms of accuracy. In the next chapter, we attempt to address these limitations.

CHAPTER

5

DEEP LEARNING APPROACHES

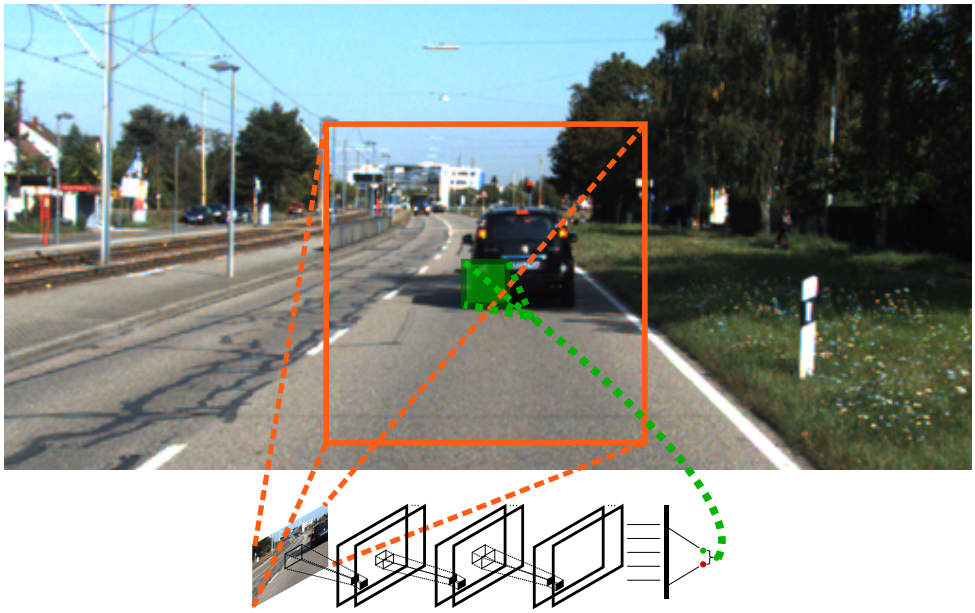
Following the implementation and tests related to the appearance and geometry based detection we came to the following conclusions: (I) the benefit from using stereo to road detection is small and do not compensate its computational and hardware cost, we hence changed our focus to monocular-only detection and (II) deep learning, especially Convolutional Neural Networks (CNNs), could substitute the whole system and provide better accuracy and processing time. We hence studied how to use CNNs for the task of road detection. First, we proposed a CNN architecture focusing on processing speed, which is described in Section 5.1. We then proposed a new neural network layer, which effectively groups contextual information. This layer and the results obtained with it are detailed in Section 5.2.

5.1 Fast Network Architecture for Road Detection

In this Section, we present a road detection system based on Convolutional Neural Networks (CNNs). It operates by extracting patches around a pixel or region of the image and classifying those patches using a trained CNN. The output class is attributed to the pixel or region from which the patch was centered, as shown by Figure 28. We hence classify the pixel or region using not only its information but also information about its surrounding, that is, contextual information. The specific CNN architecture was designed to allow its conversion to a Fully Convolutional Network (FCN) at inference time. In turn, this conversion may allow the use of a large contextual window while maintaining real-time inference.

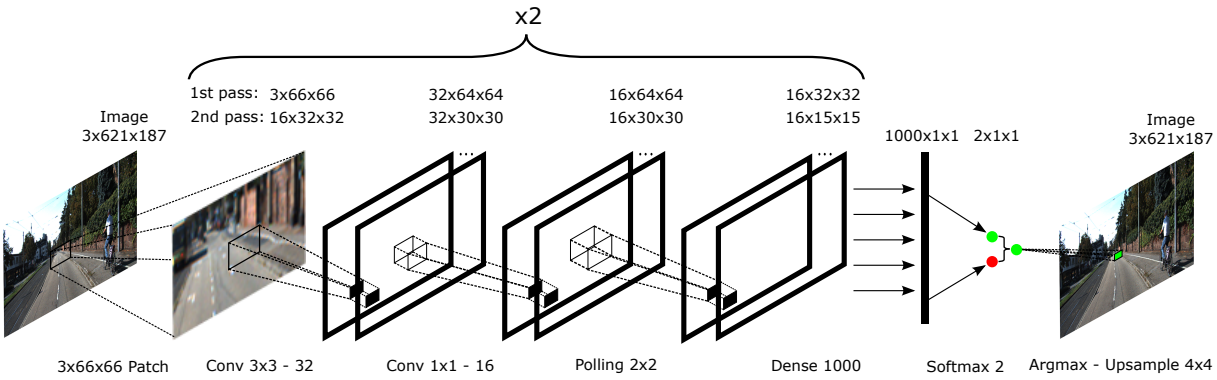
The proposed network architecture, which we have named **FCN-LC**, can be seen in Figure 29. The input to the network is a three-channel image patch and its output is a class (road or non-road) that is attributed to the 4×4 region in the center of the patch. To classify a whole image, a patch should be extracted and classified for every 4×4 region

Figure 28 – Overview of the proposed approach. A patch (marked in orange) is feed to a convolutional neural network, its output (road or non-road) is attributed to the center region of the patch (marked in green)



Source: Elaborated by the author.

Figure 29 – Detailed view of the **FCN-LC** network architecture. The input to the model is a patch extracted from the image, its output is the class (road or non-road) which is attributed to a 4×4 region in the center of the input patch.



Source: Elaborated by the author.

of the original image (i.e. patches are extracted with a 4×4 stride). The network itself starts with a conv. 3×3 - 32 (32 filters sized 3×3 each) layer, followed by a conv. 1×1 - 16 and a max-polling layer of 2×2. These three layers are repeated in sequence with the same parameters. Finally, there is a fully-connected layer with 1000 neurons and a final layer with 2 neurons (one for each class). All convolutional layers have a stride of 1×1, are followed by the ReLU activation function and do not employ padding. The first fully connected layer is followed by the ReLU function while the final layer implements the Softmax loss function.

This architecture was inspired by the LeNet-5 architecture (convolutions followed by polling) and by the more recent "Network-in-Network" (NiN) (LIN *et al.*, 2013) architecture which adds additional 1×1 convolution layers before polling. The reason for adding this 1×1 convolutional layer in our network is threefold: (I) it increases the network depth allowing it to learn higher-level features; (II) it reduces the number of channels by a factor of half, speeding up training and inference and (III) it does not reduce the channel size which is important since we do not use padding between layers. We did not increase the number of convolution filters after the first polling (as it is usually the case) since preliminary experiments showed little to no benefit of doing so. Further design decisions will be clarified in the next Section.

5.1.1 Inference

CNN architectures, as the one proposed, can be converted into a Fully Convolutional Network (FCNs) by converting the fully connected layers into convolutional layers (LONG *et al.*, 2015). The benefit of doing so is that, instead of inputting only a patch, the whole image could be used as input and the network would output the classes for every image region. The use of FCNs for inference is especially important in cases where the contextual window or patch is large and there is a lot of overlap between subsequent patches, which is the case of the proposed architecture. Although we could directly train a FCN, instead of using it only for inference, we have chosen not to do so for the following reasons: (I) its training is unstable due to the lack of fine batch size control (i.e. the minimum batch size is an entire image) and (II) our sampling approach (we use only 25% of all samples) would reduce the speed advantage of directly training a FCN.

To convert a standard CNN architecture (with fully connected layers) to a FCN there are at least three constraints that should be followed so that the FCN produces the exact same results as a per patch classification: (I) there could be no padding between layers, as the padding in the FCN and the regular CNN would be placed in different places; (II) the size of the channels to the first fully connected layer should be odd sized and (III) the output class should be upsampled to compensated the effects of the polling layers present in the network. These constraints further clarify the design decisions involving the proposed model, for instance, that is why we attribute the classification result to a 4×4 region of the image (we have 2 polling layers of 2×2 each).

Concretely, at inference time, we convert the two fully connected layer in our model to convolutional layers. Taking, for example, an input patch of 66×66, the first fully-connected layer with 1000 neurons would be converted into a 15×15 - 1000 convolutional layer. This conversion is possible because both layers share the same number of weights. The last fully connected layer would become a 1×1 - 2 convolutional layer. It should be clear that both networks produce the same results using effectively the same operations,

the only difference is that the FCN network is able to deal arbitrarily sized inputs and outputs hence being able to classify the whole image at once. This inference scheme allows efficient inference even when using large contextual window sizes.

5.1.2 Experiments

Dataset and Setup

To train and evaluate our approach we employed the KITTI Vision Benchmark Suite as in the classic approach. In this case, however, we only made use of the monocular color images and we do not distinguish between the three road categories.

We divided the 289 training images into two different sets, one for training with 260 images and one for validation containing 29 images. All the quantitative results, excluding the benchmark submission, are referent to these 29 images. We use the images at half of their original resolution and the results are linearly interpolated back to the original resolution for evaluation.

We implemented our work using the Python language, the OpenCV[1] library and a modified version of the Caffe framework (JIA *et al.*, 2014) with the cuDNN v2 library[2]. The tests were conducted on a machine equipped with an Intel Core i7-4930K, 64GB RAM, and a NVIDIA Titan X.

Training Scheme

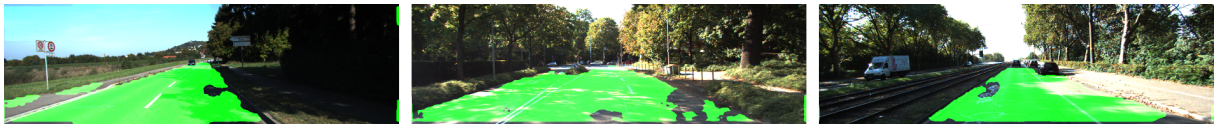
Using the training and validation images, we created training and validation datasets for each of the patch sizes tested. Each sample consists of the RGB patch and its referent class. To create the samples, we scan the image skipping 4 pixels in each axis (stride of 4) and extracted the patch centered around each 4×4 region. We included only samples whose 4×4 regions are of a single class, ignoring ambiguous samples. All images are padded (using reflection) so the 4×4 regions cover the full original image. The probability of including a sample in our dataset is proportional to the region it would occupy in the BEV space, hence the loss of our network approximates the benchmark evaluation more closely. In total, only 25% of all possible samples were included, using more samples yielded no performance benefit. All samples are standardized channel-wise.

We conducted the training using mini-batch gradient descent with a batch size of 100, initial learning rate of 0.01, momentum of 0.9 and an L2 weight decay rate of 0.0005. After each training session, we evaluated the model's F-measure using the validation images.

[1] <http://www.opencv.org/>
[2] <https://developer.nvidia.com/cudnn>

Figure 30 – Sample classification results using the test set of the KITTI road detection dataset

(a) Above the average sample results



(b) Average sample results

(c) Below the average sample results

Source: Elaborated by the author.

Table 7 – Patch Size Evaluation (in %)

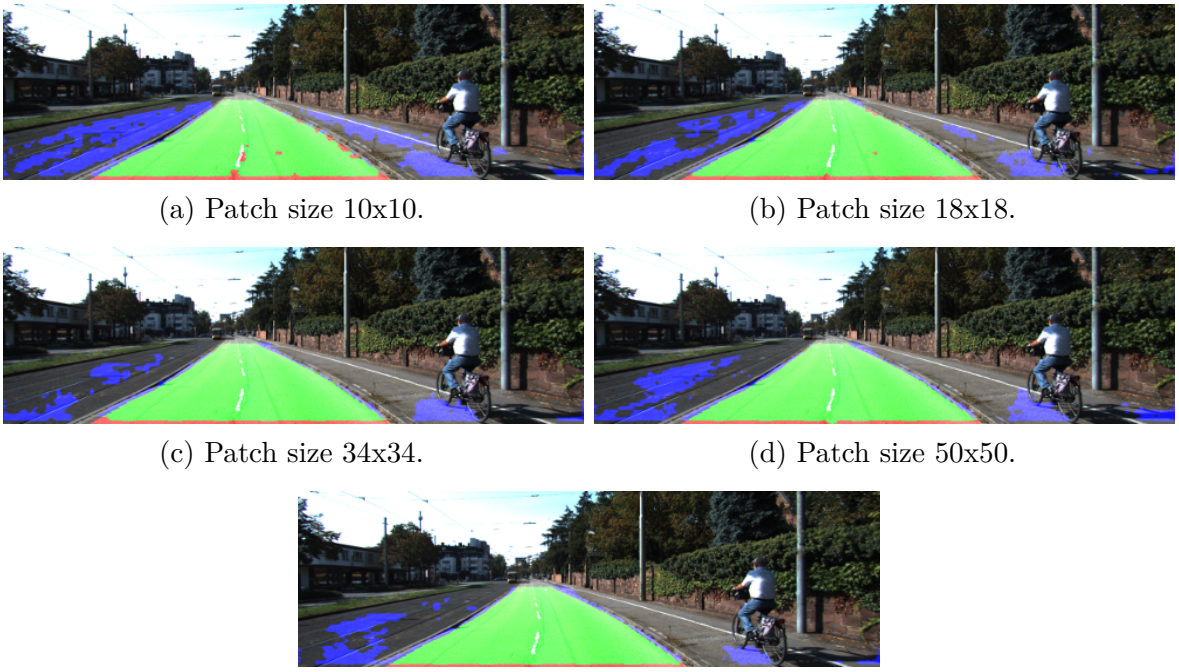
Patch Size	F-measure	Accuracy	Pre.	Rec.	Inf. Time
10x10	84.8	88.7	89.3	80.8	8 ms
18x18	88.3	91.1	91.1	85.8	9 ms
34x34	90.9	93.1	93.4	88.5	14 ms
50x50	91.7	93.6	92.6	90.9	21 ms
66x66	92.2	93.9	92.8	91.6	25 ms

Source: Research data.

Qualitative Results

Figure 30 shows a sample of the obtained results using the images of the test set. Each column represents a road category, starting with UM, followed by UMM and finally the UU category. The rows or sub-figures represent the subjective "quality" of the results, ranging from above the average to below the average. The first row reveals that our approach performs reasonably well in shadowed areas and, as the output from the FCN model is linearly interpolated, the artifacts of classifying "blocks" (due to the use of polling layers and half of the original resolution) is barely visible. The second row shows a limitation of our method, where it behaves erratically and may misclassify small regions. The final row displays our worst sample results in each road category and reveals two further limitations: its inability to deal with extreme lighting conditions and with different road surfaces types. More qualitative results can be seen as a video at <http://youtu.be/3FER84XD17w>.

Figure 31 – Classification results using different patches sizes where green represents true positive, red false negative and blue false positive.



(a) Patch size 10x10.

(b) Patch size 18x18.

(c) Patch size 34x34.

(d) Patch size 50x50.

(e) Patch size 66x66.

Source: Elaborated by the author.

Table 8 – Network-in-Network Evaluation (in %)

Model	F-measure	Accuracy	Prec.	Rec.	Inf. Time
With NiN	92.2	93.9	92.8	91.6	25 ms
Without	92.1	93.9	93.4	90.8	41 ms
Diff.	-0.1	0.0	+0.6	-0.8	+16 ms

Source: Research data.

Patch Size Evaluation

We initially tested the effect of the patch size on the accuracy and the inference time of the model. The specifically chosen sizes are related to the constraints imposed by the use of a FCN for inference. The quantitative results are shown in Table 7, where the inference time refers only to the model time. As expected, the F-measure increases with the patch size due to the larger context. The inference time scaled approximately linearly after the size 18×18. A visual sample of the results can be seen in Figure 31. The number of false positives decreases significantly with larger patch sizes and the false negatives are virtually all removed.

Table 9 – Urban Road KITTI Benchmark Results (in %)

Method	MaxF	Pre.	Rec.	FPR	FNR	Runtime
DNN (MOHAN, 2014)	93.43	95.09	91.82	2.61	8.18	2s
FCN-LC	90.79	90.87	90.72	5.02	9.28	0.03s
HIM (MUNOZ *et al.*, 2010)	90.64	91.62	89.68	4.52	10.32	7s
NNP	89.68	89.67	89.68	5.69	10.32	5s
StixelNet (LEVI *et al.*, 2015)	89.12	85.80	92.71	8.45	7.29	1s
CB	88.97	89.50	88.44	5.71	11.56	2s
FusedCRF (XIAO *et al.*, 2015)	88.25	83.62	93.44	10.08	6.56	2s
ProbBoost (VITOR *et al.*, 2014)	87.78	86.59	89.01	7.60	10.99	150s
SPRAY (KUEHNL *et al.*, 2012)	87.09	87.10	87.08	7.10	12.92	0.04s

Source: <http://www.cvlibs.net/datasets/kitti/eval_road.php> (27 Aug. 2015)

Network-in-Network Evaluation

We also evaluated the role of the additional 1×1 convolutional layer, which is a key component of the proposed architecture. We trained a network with a patch size of 66×66 but removing the two 1×1 convolutional layers of the model. The results are shown in Table 8. While the performance is virtually the same, the inference time is higher. This result validates our assumption that by adding a 1×1 convolutional layer we could maintain (or improve) performance while reducing the inference time.

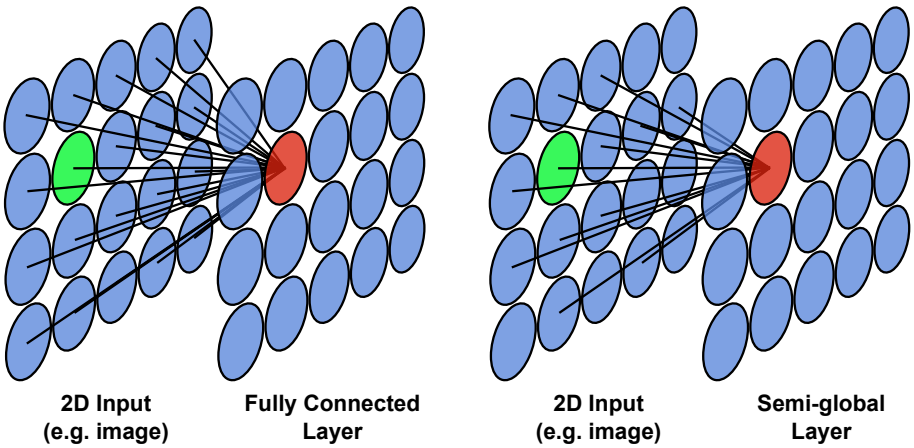
5.1.3 Benchmark Evaluation

To compare our method with others, we submitted our results, under the name of **FCN-LC** (Fully Convolutional Network - Large Context), to the KITTI road detection benchmark[3]. Table 9 presents the first nine results in the *Urban Road* category (at submission time) which is an aggregate of all three road categories (UU, UM, and UMM). Our method achieved the second best score out of 33 participants while being the fastest method (excluding the baseline submission). These results include the participants taking advantage of LIDAR (FusedCRF) or stereo vision (NNP and ProbBoost) data.

The first method of the benchmark (DNN), already mentioned in Chapter 3 uses a combination of convolutional and deconvolutional layers to include a global context and also uses position-specific classifiers to take advantage of a positional prior (i.e. the bottom-center region is more likely to be the road). Both the global context and the positional prior may explain their higher score, however, as they use many larger models, their inference time is two orders of magnitude slower than ours. The HIM method also

[3] <http://www.cvlibs.net/datasets/kitti/index.php>

Figure 32 – Illustration of the difference between the connection pattern of a fully connected layer and a semi-global one. Specifically, it shows how a single neuron (shown in red) connects to the layer input: in the case of the semi-global layer it connects with the neurons in all eight directions (up-down, left-right and the four diagonals) starting from the pivot neuron, i.e. the neuron, of the previous layer, in the same 2D position as the one in red (shown in green).



Source: Elaborated by the author.

uses a global context and while their scores are similar to ours, their inference time is far from being practical. Furthermore, methods employ a range of techniques to achieve their results, from post-processing their results with stereo information (NNP), to using "rays features" to include contextual information (SPRAY). One topic that is present in most of these works (DNN, HIM, CB, FusedCRF, and SPRAY) is the use of contextual information, this includes the classic approach (CB) described in Chapter 4 where we made use of "contextual blocks" to efficiently include contextual information, however our implementation did not reach real-time and, as we used a standard MLP (a shallow model), we had to use a selection of hand-crafted features which may not have been the most appropriated for the task.

5.2 The Semi-global Layer

Following the positive results achieved with CNNs, we proposed a new deep neural network layer for image segmentation. This layer could be used either substituting (fully connected) layers in a CNN architecture or as additional layers, providing a global context to the model. The proposed layer is basically a sparsely connected version of a fully connected layer (which connects every neuron of one layer to every neuron of the previous layer). The sparse pattern used was inspired by semi-global stereo methods and thus, we choose to call it the semi-global layer, it is also the same pattern we have used for the contextual blocks of the classic approach (Figure 20). The connection pattern can be seen in Figure 32. It connects in all eight directions (up-down, left-right and the four diagonals),

providing a global context to the network while avoiding the huge number of weights (parameters) of a fully connected layer. The layer is a good match for CNNs as such models usually work with 2D data and our layer maintains the 2D structure of the neurons (or feature maps) and thus it is compatible with most (if not all) CNN architectures. We expect this layer to enhance the performance, both in terms of inference speed and accuracy, of CNN architectures for the task of image segmentation and, in the context of this thesis, we applied it to the road detection task, where context has an important role.

5.2.1 Details

In a CNN, the input and output of a 2D convolutional layer are 2D structured (mostly images) and they present spatial dependencies, that is what convolutional layers exploit. Our layer is based on this same idea, where we are able to exploit spatial dependencies and the 2D structure of the input to incorporate a global context while being computationally feasible. Given a 2D input, our layer outputs a 2D output with the same size of the input. Figure 32 illustrates our layer and compares it to a fully connected layer, specifically the figure presents the case of a 4 by 5 input and only the connections (or weights) of a single neuron (colored in red). The output of the fully connected layer was 2D organized only for illustration purposes, in contrast, our layer depends upon its 2D structure since the specific connections of each neuron depend on its 2D position. It should be also noted that it is not possible to share weights in our layer as the number of weights and its specific connections vary between neurons.

The pattern of our layer was inspired by semi-global stereo methods. Stereo methods are methods for pixel-wise matching of stereo images. They are divided into three categories: local, semi-global and global. By analogy, there are local layers (convolution or more appropriately local convolution), global layers (fully connected) but there are not semi-global layers. The proposed layer fills in this gap in neural networks layers. Semi-global stereo methods are known for their accuracy, comparable or even better than global methods, and for their low computational cost. Our proposal follows on this idea with the added bonus of using fewer weights and hence being less prone to over-fitting than a fully connected layer.

The output of the semi-global layer can be calculated using dense-sparse vector dot product if the input and the weights of each neuron are flattened into vectors. Assuming that is the case, the output of the semi-global can be calculated as follows:

$$f\left(\mathbf{i};\mathbf{w}_j\right)=\mathbf{i}\cdot\mathbf{w}_j;\quad \forall j\in\{1,2,3\ldots N\}, \tag{5.1}$$

where $\mathbf{i}$ is the flattened input, $\mathbf{w}_j$ is the flattened (sparse) weights of the neuron j and N is the number of neurons of the layer. However, in practice, the input will probably come

in batches (many inputs at once); that being the case, we can organize the input into a matrix, where each column is a flattened 2D input and also organize the weights into a matrix, where each row is the weights of a neuron j. That being the case, we can calculate the output of the layer using a sparse-dense matrix product:

$$f(\mathbf{I}; \mathbf{W}) = \mathbf{W}\mathbf{I} : \mathbf{W} \in \mathbb{R}^{N \times K}, \mathbf{I} \in \mathbb{R}^{K \times N}, \tag{5.2}$$

where K is the batch size. To extend the layer so it can take batches of multi-channel images (batches of three-dimensional tensors) as input, all channels can be flattened into a single vector and the batch organized into a matrix, as in the previous case. The weight matrix should be extended by repeating itself column-wise so that the size of the input matches the number of weights in each row.

To enable learning, the gradient w.r.t. the input ($\mathbf{I}$) and the weights ($\mathbf{W}$) should also be computed, the gradient w.r.t. the input is given by:

$$\frac{\partial f(\mathbf{I} : \mathbf{W})}{\partial \mathbf{I}} = \mathbf{I}\mathbf{W}^T. \tag{5.3}$$

So far, the formulas are exactly the same for a fully connected layer (given the same input and weights organization) the only difference being that $\mathbf{W}$ is a sparse matrix instead of a dense one. However the gradient w.r.t. to the weights is different due to the sparsity and it is given by:

$$\frac{\partial f(\mathbf{I} : \mathbf{W})}{\partial \mathbf{W}_{jk}} = (\mathbf{I}^T)_k \cdot \mathbf{W}_j; \quad \forall (j, k), \tag{5.4}$$

where we use the subscriptions for indexing, e.g. $\mathbf{W}_{(j,k)}$ refers to a single weight in row j and column k and $(\mathbf{I}^T)_k$ refers to the vector in row k of the transposed input matrix $\mathbf{I}$.

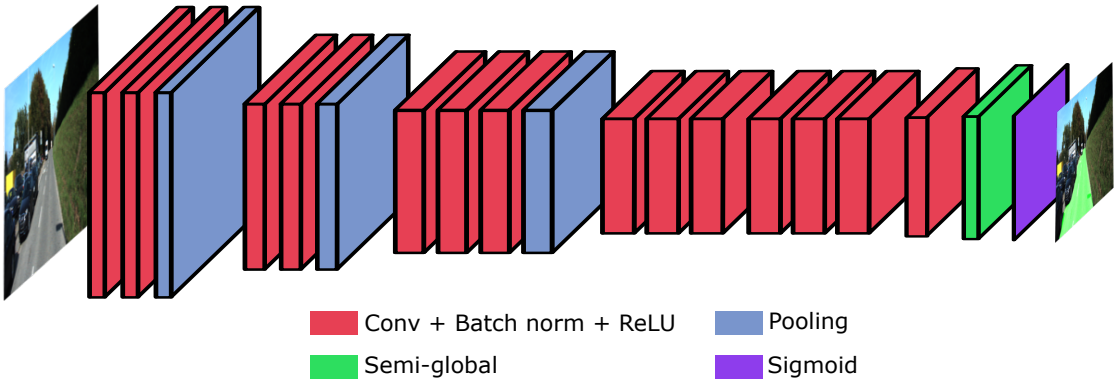
5.2.2 Implementation

We implemented the proposed layer for its usage with the Torch[4] deep learning library. The sparse weight matrix is generated using Numpy and loaded in Torch using *npy4th* library. The indexing of the sparse matrix is done in both the coordinate list (COO) and compressed sparse row (CSR) formats depending on the calculation being performed.

In the case of the CPU implementation, we employed the COO sparse matrix format and created custom C++ functions to carry on the calculations. The GPU implementation of Equation 5.2 and Equation 5.3 were implemented using the functions *csrmv* (vector product of sparse matrix) and *csrmm2* (matrix-matrix product) of the *Nvidia cuSparse* library. For Equation 5.4 we implemented a custom CUDA kernel. To access the *Nvidia cuSparse* from the Torch library a series of changes in the core files of the library had

[4] <http://www.torch.ch/>

Figure 33 – Convolutional neural network architecture used to validated the semi-global layer implementation and expected accuracy. It consists in a VGG-16 net as described in (SIMONYAN; ZISSERMAN, 2014), without the last three layers and the last polling layer and with the addition of a convolutional layer and a semi-global layer in the case of **VGG-SG**



Source: Adapted from Badrinarayanan *et al.* (2015).

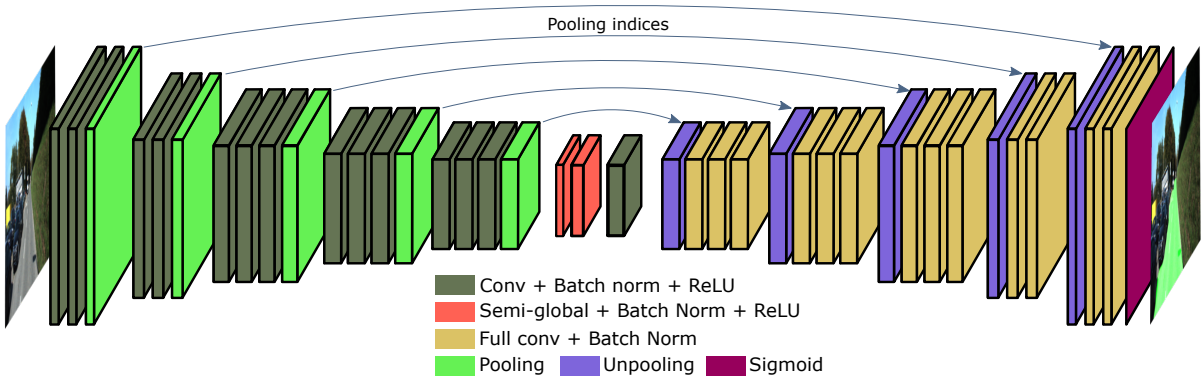
to be made. None of the calculations take advantage of the fixed pattern for speed optimizations as we employed a generic sparse matrix-matrix product implementation. While this compromises the processing speed for our specific connection pattern, it allows the test of others connections patterns. The library modifications and layer implementation were made publicly available through the GitHub[5] platform in the website <https://github.com/caiom/distro>.

5.2.3 Network Architectures

Initially, we used a simple model to validate our implementation of the proposed layer and to test whether or not the addition of the semi-global layer improves the accuracy of the model. We have chosen to use the 16-depth network proposed by (SIMONYAN; ZISSERMAN, 2014), popularly known as VGG-16. To make it a fully convolutional CNN architecture we removed all of its fully connected layers (the last three layers) and added a convolutional layer with 32 kernels sized 1×1 each in the end. This layer is used to reduce the number of channels for the output layer. We also have removed the last polling layer to have a more fine-grained segmentation result. Finally, based on the output layer, we have created two architectures. One is called **VGG-FC** and has a convolutional layer with one kernel sized 3×3 as the output. The other architecture is the **VGG-SG**, it has the semi-global layer as the output layer and can be seen in Figure 33. We use the sigmoid function as the activation function of the output layer in both cases. The only difference of **VGG-SG** and **VGG-FC** is the last layer, therefore we can test how the proposed layer performs in comparison to a standard fully convolutional CNN, which has only convolutional layers. Although the tests with this initial architecture were made

[5] <http://www.github.com>

Figure 34 – Illustration the selected convolutional neural network architecture. It consists of an encoder-decoder architecture, where convolutions are used as encoders and full convolutions (deconvolutions) as decoders; the proposed semi-global layer sits in the middle, providing a global context to the network



Source: Adapted from Badrinarayanan *et al.* (2015).

before other tests, we have chosen to group the results of all tests using the semi-global in a single Section and accordingly we will continue to describe the other tested architecture.

After the initial testing with the **VGG-***, we have chosen to use the proposed layer with a larger model aiming for a higher accuracy. The selected model is described in (BADRINARAYANAN *et al.*, 2015) and it can be seen, with the addition of the semi-global layers, in Figure 34. This type of architecture is known as a convolutional encoder-decoder, where the "encoding" is done by convolutional layers and the "decoding" by full convolutions (deconvolutions). This scheme allows the network to learn important invariances in the encoder part, but also to learn to upsample the results from the encoder so that the output matches the input size. We have created four model variants for testing, the first one, called **SegNet** is the original model, without any layer between the encoder and decoder parts. A second variant is called **SegNet-CV** where we added two convolutions in the middle, in the **SegNet-FC** variant we add two fully connected layers and finally **SegNet-SG** where we add two semi-global layers.

5.2.4 Experiments

Dataset and Setup

We maintained the dataset KITTI Vision Benchmark of the previous deep learning tests, again using only the monocular image and not distinguishing between the three road categories present. We also maintained the training and validation split, using 260 images for training and 29 for validation. We reduced the resolution of the images to 800 by 192 pixels and after inference, the images are scaled back to the original resolution. The tests were conducted on a machine equipped with an Intel Core i7-4930K, 64GB RAM, and a

Table 10 – Models comparison

Model	F-measure	Prec.	Rec.	FPR	FNR	Inf. Time
SegNet-SG	**96.6**	97.17	96.06	1.43	3.94	80 ms
SegNet	96.34	96.33	96.34	1.9	3.6	78 ms
SegNet-FC	96.27	96.82	95.72	1.78	4.28	92 ms
SegNet-CV	96.1	97.04	95.18	1.55	4.82	135 ms
VGG-SG	**95.5**	96.08	95.23	3.21	5.87	40 ms
VGG-FC	94.9	95.62	94.71	4.71	5.97	39 ms

Source: Research data.

NVIDIA Titan X.

Training

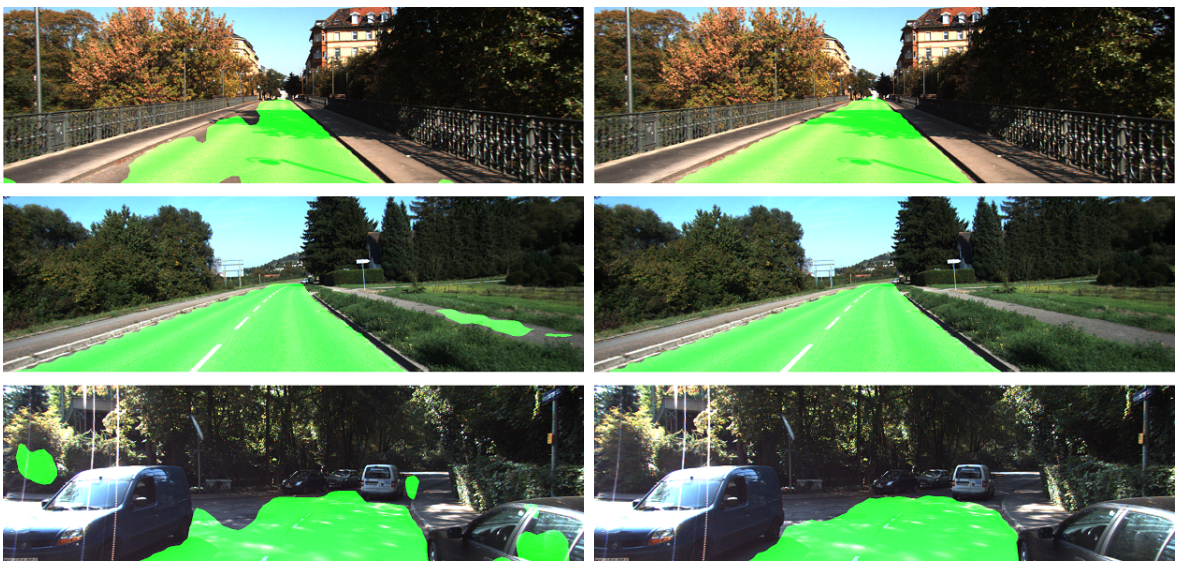
Differently from the initial deep learning approach, where examples were images patches, we now train the network directly with full images. This is important, because in this manner, the semi-global layers can learn a global context. We conducted the training using the standard mini-batch gradient descent with a batch size of two, an initial learning rate of 0.3, a momentum of 0.9, an l2 weight decay of 0.0005.

Semi-global Layer Evaluation

In Subsection 5.2.3 we described two base architectures, the SegNet and the VGG. For each of these architectures, we have created variations to test the effect that the semi-global layer has in terms of accuracy, computational cost and how it compares to other layers. Starting with the architectures that we have used to validate the semi-global layer implementation, that it, VGG-FC and VGG-SG, we can observe in Table 10 that VGG-SG performs significantly better than VGG-FC while having virtually the same processing time. More interesting however are the qualitative results, as shown in Figure 35. These pictures provide an insight on how the semi-global layer helps in the road segmentation task by providing a context to the classifier. All the context-related mistakes, such as misclassifying the sidewalk, are resolved by just changing the output layer of the network. These sorts of corrections are present in the majority of the test images and are not a particularity of the images from Figure 35. These initial results were the motivation behind the use of the semi-global in the SegNet architecture.

To test the semi-global layer using the SegNet base architecture we varied the layers between the encoder and decoder parts of the architecture. Table 10 shows the results using the validation set. We can see that the architecture with the semi-global layer yielded the best F-measure while maintaining almost the same inference time of the original model,

Figure 35 – Qualitative comparison between VGG-FC, which uses a convolutional layer as the output layer and VGG-SG, which uses the semi-global layer as the output one



(a) VGG-FC (b) VGG-SG

Source: Elaborated by the author.

repeating the results with the VGG architecture. The other layers performed worst than the original architecture probably due to overfitting, especially in the case of SegNet-FC model. In addition to these numeric results, the effects of the use of the semi-global layer can be seen qualitatively on the resulting images as shown in Figure 36. Similarly to the qualitative tests of the VGG architecture, the addition of the semi-global layers provided the contextual cues needed to correctly classify regions of similar appearance, such as the sidewalk region. It also improved the classification of different road surfaces as in the bottom images, where the road surface is made of rock blocks. More qualitative results can be seen in the form of a video at <https://goo.gl/7Oi3td>

5.2.5 Benchmark Evaluation

To compare our method with others, we submitted our results to the KITTI road detection benchmark[6]. Table 11 presents the first seven results in the *Urban Road* category (at submission time) which is an aggregate of all three road categories (UU, UM, and UMM). Our best architecture achieved the second best score out of 50 participants.

Unfortunately most submissions on the benchmark today do not have references and very little information in its descriptions fields. What it is safe to assume is that the top ten submissions employ some architecture of CNN. The work Marv3DNet seems to be using the SegNet architecture or a very similar one as it states in the description: “An

[6] <http://www.cvlibs.net/datasets/kitti/index.php>

Figure 36 – Comparison between the original SegNet and the SegNet with Semi-global layers

(a) SegNet (b) SegNet-SG

Source: Elaborated by the author.

Table 11 – Urban Road KITTI Benchmark Results (in %)

Method	MaxF	Pre.	Rec.	FPR	FNR	Runtime
SAIT	96.02	96.24	95.79	2.06	4.21	0.05 s
SegNet-SG	94.91	94.75	95.08	2.9	4.92	0.08s
Marv3DNet	94.88	94.84	94.91	2.85	5.09	0.17 s
AXN	94.69	94.70	94.69	2.92	5.31	0.06 s
SSL	94.69	94.89	94.49	2.81	5.51	0.05 s
Up-Conv-Poly (OLIVEIRA *et al.*, 2016)	93.83	94.00	93.67	3.29	6.33	0.08 s
DNN (MOHAN, 2014)	93.43	95.09	91.82	2.61	8.18	2s

Source: <http://www.cvlibs.net/datasets/kitti/eval_road.php> (6 May. 2016)

adapted deconvolution approach based on VGG as encoder and FCN as decoder", which is the SegNet architecture. The first method (SAIT) presents no description, in any case, we believe that the use of our layer could improve its results. A more detailed analysis of the results will be presented in the next chapter.

5.3 Final Considerations

In this chapter we have presented a CNN architecture for road detection focused on processing speed and a new neural network layer, the semi-global layer, that efficiently provides spatial context to a neural network. The general intention of the proposed solutions is to overcome the limitations of the classical approach. By using deep learning we avoid

the need to select or create features extractors and that is reflected in the accuracy. The processing speed is a major limitation of the proposed classic approach, naturally, the first architecture proposed (**FCN-LC**) was intended to alleviate such limitation. The semi-global layer has a similar purpose to the block scheme proposed in the classic approach, that is, aggregate the information of spatially distant regions. The semi-global layer, however, does it inside a neural network and requires no padding being able to extend to the whole image. In addition, the implementation of the semi-global layer was done using a generic sparsely connected layer that could be used to implement others patterns of connections and is publicly available through the GitHub platform. In next the chapter, a more in-depth analysis of the results and methods of this thesis is provided.

CHAPTER

6

DISCUSSION AND RESULTS SUMMARY

In this chapter, we provide a discussion about some of the recurrent topics in this thesis and also a summary of the main results.

6.1 Shallow vs Deep Models

Since the convincing 2012 ImageNet results, the computer vision community has been adopting deep learning models for all sort of visual tasks and this thesis reflects such a transition. Whether or not deep learning is the way forward in every visual task is an open question but there is no denying of its superiority in most of them, including the task of road detection. After completing the implementation of our initial idea using traditional ML models we had basically two options: continue to invest in traditional models or study and adopt deep learning models. At the occasion, we decided for deep learning and this thesis was divided by accordingly. Nowadays deep learning usage is widespread and they seem to be an obvious choice for any visual task; not only they learn direct from raw data (avoid the need to create/use a feature extractor) but, more importantly, their accuracy is higher than the one obtained using traditional models.

One downside to deep learning is the number of (hyper) parameters. For instance, to apply deep learning one has to decide what layers to use, the hyperparameters of each layer (e.g. the number of kernels), the learning rate, regularization methods and etc. This may lead someone to think that the manual effort that went into the creation and selection of features is now devoted to adjusting the model's hyperparameters. We believe that is true to some extent. Deep learning models do require some manual tuning but this work tends to pay off in terms of accuracy. Alternatively, this flexibility could be considered a strength, especially in the hands of someone who can make a good use of it. That is exactly what we intended to do in Chapter 5, where our efforts in that sense are described. The results obtained using deep learning favor our choice for deep learning. For instance,

our first deep model **FCN-LC** is more accurate and two order of magnitude faster than the traditional ML model. That holds even with the use of geometric information by the traditional model. These results motivated further investments into deep learning models which finally resulted in the semi-global layer proposal and implementation.

6.2 The Use of Geometric Information

From the start, we had the intention to use a stereo camera. It came as a natural choice since it is an affordable sensor and it provides geometric information of environment which is key to detect unobstructed areas. We hence have created a geometric road detector and used its output to augment the appearance-based (classic) classifier. The following tests, however, revealed a smaller improvement in accuracy than expected. In the KITTI road benchmark, the improvement from using the geometric-based detector is less than 1%. We believe that one of the reasons for such small improvement is the error correlation, that is, both detectors tend to make mistakes in the same areas. For instance, the stereo method, in which the geometric detector is dependent, tends to make mistakes in poorly illuminated areas and that is also the case of the appearance-based classifier. The computational cost is also a factor against the use of stereo cameras. The processing time for the stereo method alone is 0.5s in a standard *Intel i7* processor, which may prevent the real-time usage of the geometric detector if specialized hardware is not available. For those reasons, we decided to drop the use of the stereo camera in favor of using only a traditional camera with the deep learning approaches. An interesting future work could be on how to use the geometric information in a deep learning model. For instance, the depth map could be used as a channel of the input or a CNN-based geometric only detector could be created. We believe however that in both cases the improvement over a traditional camera would be small and hardly compensate the additional computational cost.

6.3 Benchmark Results Summary

In Table 12 we can see all the results that we submitted to the KITTI benchmark alongside other participants. We have added to the table only published works so we can compare methodologies and also because unpublished works may have used additional data for training, making for an unfair comparison. The table shows a progression in F-score from the proposed methods. Starting with appearance only classical approach (**App. only (CA)**) to the SegNet architecture with the proposed semi-global layer (**SegNet-SG**). The difference in F-score between classic approaches and deep ones is expected given the superiority of deep learning for visual tasks. We have however a big gap between **FCN-LC** and **SegNet-SG**, two deep approaches. This difference can be attributed to differences in

Table 12 – First thirteen results in the urban road KITTI benchmark, where the methods proposed in this thesis are marked in boldface. Only published works were considered (in %)

Method	MaxF	Pre.	Rec.	FPR	FNR	Runtime
SegNet-SG	94.91	94.75	95.08	2.9	4.92	0.08s
Up-Conv-Poly (OLIVEIRA *et al.*, 2016)	93.83	94.00	93.67	3.29	6.33	0.08s
DNN (MOHAN, 2014)	93.43	95.09	91.82	2.61	8.18	2s
Up-Conv (OLIVEIRA *et al.*, 2016)	92.39	93.03	91.76	3.79	8.24	0.05s
FTP (LADDHA *et al.*, 2016)	91.61	91.04	92.20	5.00	7.80	0.28s
FCN-LC	90.79	90.87	90.72	5.02	9.28	0.03s
App. + Stereo + CRF (CA)	90.13	89.74	90.51	5.33	9.73	2.7s
App. + Stereo (CA)	89.88	89.52	90.24	5.82	9.76	2.5s
HIM (MUNOZ *et al.*, 2010)	90.64	91.62	89.68	4.52	10.32	7s
NNP (CHEN *et al.*, 2015)	89.68	89.67	89.68	5.69	10.32	5s
StixelNet (LEVI *et al.*, 2015)	89.12	85.80	92.71	8.45	7.29	1s
App. only (CA)	88.97	89.50	88.44	5.71	11.56	2s
FusedCRF (XIAO *et al.*, 2015)	88.25	83.62	93.44	10.08	6.56	2s

Source: <http://www.cvlibs.net/datasets/kitti/eval_road.php> (23 Feb. 2017)

the design intentions, while **FCN-LC** was designed to be as fast as possible, **SegNet-SG** was designed to provide accurate results and hence uses a much deeper architecture.

Comparisons with other participants can also be made. From the works in Table 12, only HIM and FusedCRF use traditional ML methods. We hence have the best classic approach in the table with **App. + Stereo + CRF (CA)** however, HIM uses only appearance information and has a higher F-score than our appearance only detector (**App. only (CA)**). HIM creates a hierarchical series of classification where the input of one level is used as input to the next. Similar to our classic approach HIM focused on providing contextual information to the model and despite describing a compelling scheme for doing so, its processing time is higher than the proposed approach. Among the deep methods, ours is also the first one having more than 1% higher F-score than the second in the table (Up-Conv-Poly). The Up-Conv-Poly method uses a very similar architecture to SegNet, using convolutional layers as an encoder and deconvolutional layers as a decoder. Although there is a difference in the number of deconvolutional layers, we believe that the difference in F-score could be mostly attributed to our use of a semi-global layer.

6.4 Limitations

Despite achieving the first place among the published works in the KITTI road detection benchmark the methods proposed in this thesis have limitations. The **SegNet-SG** network is not accurate in every instance and may produce inaccurate segmentations in

some situations (see <https://goo.gl/7Oi3td>). Extreme lighting and ambiguous regions are a problem even for our most accurate method. One way to improve the accuracy of the proposed models is to use a larger training dataset. The KITTI benchmark provides only 290 training samples, covering a small range of possible scenarios. It is unreasonable to expect a practical road detector using such a small number of samples. In addition, the chosen connection pattern for the semi-global layer may not ideal for high-resolution segmentation, in this case, the number of connections becomes too large and may not fit in the memory for training.

6.5 Publications

In Appendix A we provide a list of all the publications generated while in the Ph.D. studies that this thesis is a result of. In total, we have ten publications, but only two publications are a direct result of this thesis: (I) "Vision-Based Road Detection using Contextual Blocks" and (II) "Exploiting Fully Convolutional Neural Networks for Fast Road Detection". In the first publication we described our appearance only classical approach (**App. only (CA)**) while the second one is concerning our fast deep architecture (**FCN-LC**). The proposed semi-global layer and its results are in the process of being submitted.

Of the other eight publications, six are related to this thesis topic or methods. For instance, the publications "Automatic detection of Ceratocystis wilt in Eucalyptus crops from aerial images" and "Segmentação e contagem de árvores em plantações de eucaliptos utilizando imagens aéreas" apply the proposed methodology for aggregating spatially distant features in the task of detecting crop diseases in aerial images. The publications "Real-time obstacle detection using range images: processing dynamically-sized sliding windows on a GPU", "An Efficient Obstacle Detection Approach for Organized Point Clouds" and "Detecção de Obstáculos em Tempo Real para Sensores de Profundidade Utilizando uma GPU" instead of detecting the road, they perform an obstacle detection. Unfortunately, their output, a binary obstacle/non-obstacle detection, was not the most appropriate to use with ML methods and hence we have decided not to mention it in this thesis. "CaRINA Intelligent Robotic Car: Architectural Design and Applications" is an overview of the CaRINA platform, mentioned in Chapter 1, where we have contributed with our obstacle detection method. The other two publications are reminiscent of the master's degree.

CHAPTER

7

CONCLUSION AND FUTURE WORKS

Autonomous vehicles have a rich development history and, although still under supervision, they are finally starting to gain the public roads. One of the current challenges is how to develop a trustworthy autonomous vehicle at a low, commercially viable, cost. So far, using cameras, instead of LIDARs, is the most compelling answer, but there are many challenges in doing so. One of them is how to reliably detect the navigable region or the road. This thesis tries to answer that question by using machine learning methods and proposing techniques and models targeted at the task. More concretely, in this thesis, we have presented our efforts to tackle the visual road detection task. We started by using a classic (shallow) model to learn the task from a set of manually created samples. To augment such model we have also created a geometric road detector using a stereo camera. Reflecting a trend in computer vision we have decided to switch our focus to deep learning models. As a result of that change, we have proposed a new deep architecture and a new deep neural network layer, both directed at the task of road detection. The results obtained during the referent Ph.D. studies and presented in this thesis have instigated some conclusions. Regarding the use of a stereo camera, we have come to the conclusion that the benefits are limited and probably do not compensate for the drawbacks. In addition, the results confirmed that the use of contextual information is indeed important in the task of road detection. This importance is reflected in both classic and deep models and was the guiding principle of the proposed semi-global layer. We hope that the proposed models and the insights from the obtained results will be useful in accelerating the development of autonomous vehicles or in improving ADAS systems.

As future works, we may use the implemented sparsely connected layer to test other connection patterns. For instance, it might be useful to create the connections using a Gaussian pattern instead of the pattern inspired by semi-global stereo methods. The Gaussian pattern could be a better representation of the importance of relative spatial locations. Another possibility is to use Generative Adversarial Networks (GANs)

(Goodfellow *et al.*, 2014) to train the model as performed in (Isola *et al.*, 2016). In the GAN framework, there are two networks, a generator, and a discriminator. The generator generates "fake" samples from a random latent representation and the discriminator attempts to discover whether or not its input is fake (produced by the generator) or real. These networks are trained in parallel to perform the designated tasks. The usual application is unsupervised learning, where a set of unlabeled images is used to train both networks. As a result, the generator network is able to produce real-looking samples. For the task of road detection, the generator network could do the standard segmentation process, as **SegNet-SG** does, while the discriminator could check the "plausibility" of the segmentation made by the generator. In this scheme, the discriminator might enforce the learning of important priors about the two-dimensional geometry of the road and have a better handle over ambiguous situations.

BIBLIOGRAPHY

AIZERMAN, M. A.; BRAVERMAN, E. A.; ROZONOER, L. Theoretical foundations of the potential function method in pattern recognition learning. In: **Automation and Remote Control**, 1964. p. 821–837. Citation on page 20.

ALENCAR, F. A. R.; FILHO, C. M.; SILVA, D. G. d.; WOLF, D. F. Pedestrian classification using k-means and random decision forests. In: **Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol**, 2014. p. 103–108. Citation on page 16.

ALPAYDIN, E. **Introduction to Machine Learning**: The MIT Press, 2004. ISBN 0262012111. Citation on page 19.

ALVAREZ, J. M.; GEVERS, T.; LECUN, Y.; LOPEZ, A. M. Road scene segmentation from a single image. In: SPRINGER. **European Conference on Computer Vision**, 2012. p. 376–389. Citation on page 42.

ALVAREZ, J. M. A.; LOPEZ, A. M. Road detection based on illuminant invariance. In: **IEEE Transactions on Intelligent Transportation Systems**, 2011. v. 12, p. 184–193. Citation on page 38.

ALY, M. Real time detection of lane markers in urban streets. **ArXiv e-prints**, 2014. Available: <http://arxiv.org/abs/1411.7113>. Citation on page 37.

AMBROGGI, L. D.; KONA, A. **Google's Driverless Car to Boost Chip Revenue**. IEEE Electronics 360, 2014. Available: <http://electronics360.globalspec.com/article/4289/google-s-driverless-car-to-boost-chip-revenue>. Accessed: 18/01/2017. Citation on page 13.

BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. **ArXiv e-prints**, 2015. Available: <http://arxiv.org/abs/1511.00561>. Citations on pages 71 and 72.

BALLARD, D. H. Generalizing the hough transform to detect arbitrary shapes. **Pattern Recognition**, v. 13, n. 2, p. 111–122, 1981. ISSN 0031-3203. Citation on page 37.

BOTTAZZI, V. S.; BORGES, P. V. K.; JO, J. A vision-based lane detection system combining appearance segmentation and tracking of salient points. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2013. p. 443–448. Citation on page 37.

BROGGI, A.; BUZZONI, M.; FELISA, M.; ZANI, P. Stereo obstacle detection in challenging environments: The viac experience. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, 2011. p. 1599–1604. Citation on page 40.

BROGGI, A.; CARAFFI, C.; FEDRIGA, R.; GRISLERI, P. Obstacle detection with stereo vision for off-road vehicle navigation. In: **IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops**, 2005. Citation on page 39.

BRUST, C.-A.; SICKERT, S.; SIMON, M.; RODNER, E.; DENZLER, J. Convolutional patch networks with spatial prior for road detection and urban scene understanding. **ArXiv e-prints**, 2015. Available: <http://arxiv.org/abs/1502.06344>. Citation on page 42.

BUEHLER, M.; IAGNEMMA, K.; SINGH, S. **The DARPA Urban Challenge: Autonomous Vehicles in City Traffic**. 1st. ed.: Springer Publishing Company, Incorporated, 2010. ISBN 3642039901, 9783642039904. Citation on page 14.

CARAFFI, C.; CATTANI, S.; GRISLERI, P. Off-road path and obstacle detection using decision networks and stereo vision. **IEEE Transactions on Intelligent Transportation Systems**, v. 8, n. 4, p. 607–618, December 2007. Citation on page 39.

CHEN, X.; KUNDU, K.; ZHU, Y.; BERNESHAWI, A.; MA, H.; FIDLER, S.; URTASUN, R. 3d object proposals for accurate object class detection. In: **Neural Information Processing Systems (NIPS)**, 2015. Citation on page 79.

CHURCHLAND, P. S.; SEJNOWSKI, T. J. **The Computational Brain**: MIT Press, 1994. ISBN 0262531208. Citation on page 21.

CLAESEN, M.; SIMM, J.; POPOVIC, D.; MOREAU, Y.; MOOR, B. D. Easy hyperparameter search using optunity. **ArXiv e-prints**, 2014. Available: <http://arxiv.org/abs/1412.1114>. Citation on page 54.

DAVIES, A. **We Take a Ride in the Self-Driving Uber Now Roaming Pittsburgh**. Wired, 2016. Available: <https://www.wired.com/2016/09/self-driving-autonomous-uber-pittsburgh/>. Accessed: 30/01/2017. Citations on pages 36 and 37.

_____. **Nissan's Path to Self-Driving Cars? Humans in Call Centers**. Wired, 2017. Available: <https://www.wired.com/2017/01/nissans-self-driving-teleoperation/>. Accessed: 02/02/2017. Citations on pages 36 and 37.

ERKAN, A.; HADSELL, R.; SERMANET, P.; BEN, J.; MULLER, U.; LECUN, Y. Adaptive long range vision in unstructured terrain. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, 2007. p. 2421–2426. Citation on page 40.

ETHERINGTON, D. **Delphi's self-driving car takes us on a stress-free tour of Las Vegas**. Tech Crunch, 2017. Available: <https://techcrunch.com/2017/01/04/delphis-self-driving-car-takes-us-on-a-stress-free-tour-of-las-vegas/>. Accessed: 02/02/2017. Citations on pages 36 and 37.

EUREKA PROMETHEUS Project. Eureka Network, 1987. Available: <http://www.eurekanetwork.org/project/id/45>. Accessed: 30/01/2017. Citation on page 35.

FARABET, C.; COUPRIE, C.; NAJMAN, L.; LECUN, Y. Learning hierarchical features for scene labeling. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1915–1929, 2013. Citation on page 25.

FERNANDES, L. C.; SOUZA, J. R.; PESSIN, G.; SHINZATO, P. Y.; SALES, D.; MENDES, C. C. T.; PRADO, M.; KLASER, R.; MAGALHÃES, A. C.; HATA, A.; PIGATTO, D.; BRANCO, K. C.; JR., V. G.; OSORIO, F. S.; WOLF, D. F. Carina

intelligent robotic car: Architectural design and applications. **Journal of Systems Architecture**, v. 60, n. 4, p. 372 – 392, 2014. ISSN 1383-7621. Citation on page 16.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. **Communications of the ACM**, ACM, v. 24, n. 6, p. 381–395, 1981. ISSN 0001-0782. Citations on pages 39 and 50.

FRITSCH, J.; KUEHNL, T.; GEIGER, A. A new performance measure and evaluation benchmark for road detection algorithms. In: **International Conference on Intelligent Transportation Systems (ITSC)**, 2013. Citations on pages 17, 51, and 54.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. **Biological Cybernetics**, Springer-Verlag, v. 36, n. 4, p. 193–202, 1980. ISSN 0340-1200. Citation on page 25.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**: MIT Press, 2016. Citations on pages 20 and 34.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. **ArXiv e-prints**, June 2014. Available: <http://arxiv.org/abs/1406.2661>. Citation on page 82.

GOODFELLOW, I. J.; WARDE-FARLEY, D.; LAMBLIN, P.; DUMOULIN, V.; MIRZA, M.; PASCANU, R.; BERGSTRA, J.; BASTIEN, F.; BENGIO, Y. Pylearn2: a machine learning research library. **ArXiv e-prints**, 2013. Available: <http://arxiv.org/abs/1308.4214>. Citation on page 54.

GUO, C.; MITA, S.; MCALLESTER, D. Adaptive non-planar road detection and tracking in challenging environments using segmentation-based markov random field. In: **IEEE International Conference on Robotics and Automation (ICRA)**, 2011. p. 1172–1179. Citation on page 41.

HADSELL, R.; SERMANET, P.; BEN, J.; ERKAN, A.; SCOFFIER, M.; KAVUKCUOGLU, K.; MULLER, U.; LECUN, Y. Learning long-range vision for autonomous off-road driving. **Journal of Field Robotics**, v. 26, n. 2, p. 120–144, February 2009. Citations on pages 40, 41, and 42.

HAPPOLD, M.; OLLIS, M. Autonomous learning of terrain classification within imagery for robot navigation. In: **IEEE International Conference on Systems, Man and Cybernetics**, 2006. v. 1, p. 260–266. Citations on pages 39 and 40.

HATA, A. Y.; WOLF, D. F. Feature detection for vehicle localization in urban environments using a multilayer lidar. **IEEE Transactions on Intelligent Transportation Systems**, v. 99, 2015. ISSN 1524-9050. Citation on page 16.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation (2nd Edition)**: Prentice Hall, 1998. Citation on page 20.

HINTON, G.; DENG, L.; YU, D.; DAHL, G.; MOHAMED, A. rahman; JAITLY, N.; SENIOR, A.; VANHOUCKE, V.; NGUYEN, P.; SAINATH, T.; KINGSBURY, B. Deep neural networks for acoustic modeling in speech recognition. **Signal Processing Magazine**, 2012. Citation on page 25.

HIRSCHMULLER, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In: **IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**, 2005. v. 2, p. 807–814. ISSN 1063-6919. Citations on pages 33 and 50.

INWOOD, A. **Mercedes-Benz to test autonomous cars in Australia**. Wheels, 2016. Available: <https://www.wheelsmag.com.au/news/1612/mercedes-benz-to-test-autonomous-cars-in-australia>. Accessed: 30/01/2017. Citations on pages 36 and 37.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **ArXiv e-prints**, 2015. Available: <http://arxiv.org/abs/1502.03167>. Citation on page 30.

Isola, P.; Zhu, J.-Y.; Zhou, T.; Efros, A. A. Image-to-Image Translation with Conditional Adversarial Networks. **ArXiv e-prints**, November 2016. Available: <http://arxiv.org/abs/1611.07004>. Citation on page 82.

JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. **ArXiv e-prints**, 2014. Available: <http://arxiv.org/abs/1408.5093>. Citation on page 64.

KANE, S. O. **Ford's new autonomous Fusion looks freakishly normal**. The Verge, 2016. Available: <http://www.theverge.com/2016/12/28/14100278/ford-new-self-driving-car-fusion-hybrid-testing>. Accessed: 30/01/2017. Citations on pages 36 and 37.

KIM, B.; SON, J.; SOHN, K. Illumination invariant road detection based on learning method. In: **IEEE Conference on Intelligent Transportation Systems (ITSC)**, 2011. p. 1009–1014. Citation on page 38.

KONG, H.; AUDIBERT, J. Y.; PONCE, J. General road detection from a single image. **IEEE Transactions on Image Processing**, v. 19, n. 8, p. 2211–2220, August 2010. ISSN 1057-7149. Citations on pages 37 and 38.

KONOLIGE, K.; AGRAWAL, M.; BOLLES, R.; COWAN, C.; FISCHLER, M.; GERKEY, B. Outdoor mapping and navigation using stereo vision. In: **Experimental Robotics**, 2008. v. 39, p. 179–190. Citations on pages 39 and 44.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems**, 2012. p. 1097–1105. Citation on page 25.

KUEHNL, T.; KUMMERT, F.; FRITSCH, J. Spatial ray features for real-time ego-lane extraction. In: **IEEE Conference on Intelligent Transportation Systems (ITSC)**, 2012. Citation on page 67.

KUHNL, T.; KUMMERT, F.; FRITSCH, J. Monocular road segmentation using slow feature analysis. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2011. p. 800–806. Citation on page 38.

LABAYRADE, R.; AUBERT, D.; TAREL, J.-P. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2002. v. 2, p. 646–651. Citation on page 39.

LADDHA, A.; KOCAMAZ, M. K.; NAVARRO-SERMENT, L. E.; HEBERT, M. Map-supervised road detection. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2016. Citation on page 79.

LAIBLE, S.; KHAN, Y. N.; ZELL, A. Terrain Classification With Conditional Random Fields on Fused 3D LIDAR and Camera Data. In: **European Conference on Mobile Robots**Barcelona: , 2013. p. 1–6. Citation on page 53.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–444, May 2015. Citation on page 24.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, November 1998. ISSN 0018-9219. Citation on page 25.

LECUN, Y.; MULLER, U.; BEN, J.; COSATTO, E.; FLEPP, B. Off-road obstacle avoidance through end-to-end learning. In: **Advances in Neural Information Processing Systems (NIPS)**: MIT Press, 2005. v. 18. Citation on page 42.

LEE, T. S. Image representation using 2d gabor wavelets. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 18, n. 10, p. 959–971, October 1996. Citation on page 37.

LEUNG, T.; MALIK, J. Representing and recognizing the visual appearance of materials using three-dimensional textons. **International Journal of Computer Vision**, Kluwer Academic Publishers, v. 43, n. 1, p. 29–44, 2001. ISSN 0920-5691. Citation on page 48.

LEVI, D.; GARNETT, N.; FETAYA, E. Stixelnet: A deep convolutional network for obstacle detection and road segmentation. In: **British Machine Vision Conference (BMVC)**: BMVA Press, 2015. p. 109.1–109.12. Citations on pages 42, 67, and 79.

LIMA, D.; PEREIRA, G. Um sistema de visão estéreo para navegação de um carro autônomo em ambientes com obstáculos. In: **XVII Congresso Brasileiro de Automática**, 2010. p. 224–231. Citation on page 39.

LIN, M.; CHEN, Q.; YAN, S. Network in network. **ArXiv e-prints**, 2013. Available: <http://arxiv.org/abs/1312.4400>. Citation on page 63.

LOMBARDI, P.; ZANIN, M.; MESSELODI, S. Switching models for vision-based on-board road detection. In: **IEEE Conference on Intelligent Transportation Systems (ITSC)**, 2005. p. 67–72. Citation on page 38.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2015. p. 3431–3440. ISSN 1063-6919. Citation on page 63.

LUM, H.; REAGAN, J. A. Interactive highway safety design model: Accident predictive module. **Public Roads Magazine**, v. 59, n. 2, 1985. Citation on page 13.

MA, J.; SHERIDAN, R. P.; LIAW, A.; DAHL, G. E.; SVETNIK, V. Deep neural nets as a method for quantitative structure–activity relationships. **Journal of Chemical Information and Modeling**, v. 55, n. 2, p. 263–274, 2015. Citation on page 25.

MANSO, L.; BUSTOS, P.; BURGOS, P. B.; MORENO, J. Multi-cue visual obstacle detection for mobile robots. **Journal of Physical Agents**, v. 4, n. 1, 2010. Citation on page 41.

MARK, W. van der; HEUVEL, J. van den; GROEN, F. Stereo based obstacle detection with uncertainty in rough terrain. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2007. p. 1005–1012. Citation on page 40.

MASSERA, C. M.; TERRA, M. H.; WOLF, D. F. A guaranteed cost approach to robust model predictive control of uncertain linear systems. **ArXiv e-prints**, 2016. Available: <http://arxiv.org/abs/1606.03437>. Citation on page 16.

MATIAS, L. P. N.; SANTOS, T. C.; WOLF, D. F.; SOUZA, J. R. Path planning and autonomous navigation using amcl and ad*. In: **12th Latin American Robotics Symposium and 3rd Brazilian Symposium on Robotics (LARS-SBR)**, 2015. p. 320–324. Citation on page 16.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, n. 4, p. 115–133, 1943. Citation on page 20.

MIKOLOV, T.; DEORAS, A.; POVEY, D.; BURGET, L.; CERNOCKY, J. H. Strategies for training large scale neural network language models. In: **IEEE Automatic Speech Recognition and Understanding Workshop**, 2011. Citation on page 25.

MITCHELL, T. **Machine Learning**. 1st. ed.: McGraw-Hill Education (ISE Editions), 1997. Paperback. ISBN 0071154671. Citation on page 19.

MOHAN, R. Deep deconvolutional networks for scene parsing. **ArXiv e-prints**, 2014. Available: <http://arxiv.org/abs/1411.4101>. Citations on pages 42, 43, 60, 67, 75, and 79.

MÜLLER, A. C.; BEHNKE, S. pystruct - learning structured prediction in python. **Journal of Machine Learning Research**, v. 15, p. 2055–2060, 2014. Citation on page 54.

MUNOZ, D.; BAGNELL, J. A.; HEBERT, M. Stacked hierarchical labeling. In: **European Conference on Computer Vision (ECCV)**, 2010. p. 57–70. Citations on pages 67 and 79.

MUOIO, D. **Bosch's self-driving car prototype could give us a glimpse of Tesla's Autopilot plans**. Business Insider, 2016. Available: <http://www.businessinsider.com/bosch-driverless-tesla-autopilot-2016-10>. Accessed: 02/02/2017. Citations on pages 36 and 37.

MURRAY, D.; JENNINGS, C. Stereo vision based mapping and navigation for mobile robots. In: **IEEE International Conference on Robotics and Automation (ICRA)**, 1997. v. 2, p. 1694–1699. Citation on page 39.

NAVLAB: The Carnegie Mellon University Navigation Laboratory. Carnegie Mellon University, 1984. Available: <http://www.cs.cmu.edu/afs/cs/project/alv/www/index.html>. Accessed: 30/01/2017. Citation on page 35.

NELSON, G. **Tesla beams down 'autopilot' mode to Model S**. 2015. Automotive News. Available: <http://www.autonews.com/article/20151014/OEM06/151019938/tesla-beams-down-autopilot-mode-to-model-s>. Accessed: 24/01/2017. Citation on page 13.

NOH, H.; HONG, S.; HAN, B. Learning deconvolution network for semantic segmentation. In: **IEEE International Conference on Computer Vision (ICCV)**, 2015. Citations on pages 43 and 44.

OLIVEIRA, G. L.; BURGARD, W.; BROX, T. Efficient deep methods for monocular road segmentation. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, 2016. Citations on pages 42, 43, 75, and 79.

PARAJULI, A.; CELENK, M.; RILEY, H. Robust lane detection in shadows and low illumination conditions using local gradient features. **Open Journal of Applied Sciences**, v. 3, n. 1B, p. 68–74, 2013. Citation on page 37.

PASSANI, M.; YEBES, J.; BERGASA, L. Crf-based semantic labeling in miniaturized road scenes. In: **IEEE Conference on Intelligent Transportation Systems (ITSC)**, 2014. p. 1902–1903. Citation on page 59.

'PHANTOM Auto' will tour city. The Milwaukee Sentinel, 1926. Available: <https://news.google.com/newspapers?id=unBQAAAAIBAJ&sjid=QQ8EAAAAIBAJ&pg=7304,3766749>. Accessed: 23/07/2013. Citation on page 35.

RAMSTROM, O.; CHRISTENSEN, H. A method for following unmarked roads. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2005. p. 650–655. Citation on page 38.

RANKIN, A.; HUERTAS, A.; MATTHIES, L. Evaluation of stereo vision obstacle detection algorithms for off-road autonomous navigation. In: **AUVSI Symposium on Unmanned Systems**, 2005. Citation on page 39.

RASMUSSEN, C. Grouping dominant orientations for ill-structured road following. In: **IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)**, 2004. v. 1, p. 470–477. Citation on page 37.

RODRIGUEZ, H.; COSENTINO, M. **Futurologia: Como será a vida com um carro autônomo**. 2014. Available: <http://oglobo.globo.com/economia/carros/futurologia-como-sera-vida-com-um-carro-autonomo-12710119>. Accessed: 24/01/2017. Citation on page 16.

ROTARU, C.; GRAF, T.; ZHANG, J. Color image segmentation in hsi space for automotive applications. **Journal of Real-Time Image Processing**, Springer-Verlag, v. 3, n. 4, p. 311–322, 2008. Citation on page 38.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning internal representations by error propagation**: MIT Press, 1986. 318-362 p. Citation on page 23.

RUSU, R. B.; COUSINS, S. 3D is here: Point Cloud Library (PCL). In: **IEEE International Conference on Robotics and Automation (ICRA)**, 2011. Citation on page 54.

SANTANA, P.; SANTOS, P.; CORREIA, L.; BARATA, J. Cross-country obstacle detection: Space-variant resolution and outliers removal. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, 2008. p. 1836–1841. Citation on page 40.

SCHARSTEIN, D.; SZELISKI, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. **International Journal of Computer Vision**, v. 47, n. 1, p. 7–42, 2002. Citation on page 33.

SCHNEIDER, F. E. **The European Land Robot Trial**. 2017. Available: <http://www.elrob.org/>. Accessed: 30/01/2017. Citation on page 36.

SHINZATO, P.; FERNANDES, L.; OSORIO, F.; WOLF, D. Path recognition for outdoor navigation using artificial neural networks: Case study. In: **IEEE International Conference on Industrial Technology (ICIT)**, 2010. p. 1457–1462. Citation on page 38.

SHINZATO, P. Y.; WOLF, D. F.; STILLER, C. Road terrain detection: Avoiding common obstacle detection assumptions using sensor fusion. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2014. p. 687–692. ISSN 1931-0587. Citation on page 16.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **ArXiv e-prints**, 2014. Available: <http://arxiv.org/abs/1409.1556>. Citation on page 71.

SOTELO, M.; RODRIGUEZ, F.; MAGDALENA, L.; BERGASA, L.; BOQUETE, L. A color vision-based lane tracking system for autonomous driving on unmarked roads. **Autonomous Robots**, Kluwer Academic Publishers, v. 16, n. 1, p. 95–116, 2004. ISSN 0929-5593. Citation on page 38.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014. Citation on page 25.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. **ArXiv e-prints**, 2014. Available: <http://arxiv.org/abs/1409.4842>. Citation on page 25.

TALUKDER, A.; MANDUCHI, R.; RANKIN, A.; MATTHIES, L. Fast and reliable obstacle detection and segmentation for cross-country navigation. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2002. p. 610–618. Citation on page 40.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L.-E.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; NIEKERK, J. van; JENSEN, E.; ALESSANDRINI, P.; BRADSKI, G.; DAVIES, B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P. Stanley: The robot that won the darpa grand challenge. In: BUEHLER, M.; IAGNEMMA, K.; SINGH, S. (Ed.). **The 2005**

DARPA Grand Challenge: The Great Robot Race. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1–43. ISBN 978-3-540-73429-1. Citations on pages 35 and 36.

URMSON, C.; ANHALT, J.; BAGNELL, D.; BAKER, C.; BITTNER, R.; CLARK, M. N.; DOLAN, J.; DUGGINS, D.; GALATALI, T.; GEYER, C.; GITTLEMAN, M.; HARBAUGH, S.; HEBERT, M.; HOWARD, T. M.; KOLSKI, S.; KELLY, A.; LIKHACHEV, M.; MCNAUGHTON, M.; MILLER, N.; PETERSON, K.; PILNICK, B.; RAJKUMAR, R.; RYBSKI, P.; SALESKY, B.; SEO, Y.-W.; SINGH, S.; SNIDER, J.; STENTZ, A.; WHITTAKER, W. R.; WOLKOWICKI, Z.; ZIGLAR, J.; BAE, H.; BROWN, T.; DEMITRISH, D.; LITKOUHI, B.; NICKOLAOU, J.; SADEKAR, V.; ZHANG, W.; STRUBLE, J.; TAYLOR, M.; DARMS, M.; FERGUSON, D. Autonomous driving in urban environments: Boss and the urban challenge. In: BUEHLER, M.; IAGNEMMA, K.; SINGH, S. (Ed.). **The DARPA Urban Challenge: Autonomous Vehicles in City Traffic**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 1–59. ISBN 978-3-642-03991-1. Citation on page 35.

VITOR, G. B.; VICTORINO, A. C.; FERREIRA, J. V. A probabilistic distribution approach for the classification of urban roads in complex environments. In: **Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots on IEEE International Conference on Robotics and Automation (ICRA)**, 2014. Citations on pages 59 and 67.

VOS, T.; BARBER, R. M.; BELL, B.; BERTOZZI-VILLA, A.; BIRYUKOV, S.; BOLLIGER, I.; CHARLSON, F.; DAVIS, A.; DEGENHARDT, L.; DICKER, D.; DUAN, L.; ERSKINE, H.; FEIGIN, V. L.; FERRARI, A. J.; FITZMAURICE, C.; FLEMING, T.; GRAETZ, N.; GUINOVART, C.; HAAGSMA, J.; HANSEN, G. M.; HANSON, S. W.; HEUTON, K. R.; HIGASHI, H.; KASSEBAUM, N.; KYU, H.; LAURIE, E.; LIANG, X.; LOFGREN, K.; LOZANO, R.; MACINTYRE, M. F.; MORADI-LAKEH, M.; NAGHAVI, M.; NGUYEN, G.; ODELL, S.; ORTBLAD, K.; ROBERTS, D. A. Global, regional, and national incidence, prevalence, and years lived with disability for 301 acute and chronic diseases and injuries in 188 countries, 1990-2013: a systematic analysis for the global burden of disease study 2013. **The Lancet**, v. 386, p. 743–800, August 2015. Citation on page 13.

WALLACE, R. First results in robot road-following. **International Joint Conference on Artificial Intelligence (IJCAI)**, 1985. Citations on pages 35 and 37.

WALTON, M. **Robots fail to complete Grand Challenge**. CNN International, 2004. Available: <http://edition.cnn.com/2004/TECH/ptech/03/14/darpa.race/index.html>. Accessed: 30/01/2017. Citation on page 35.

WANG, Y.; TEOH, E. K.; SHEN, D. Lane detection and tracking using b-snake. **Image and Vision Computing**, v. 22, n. 4, p. 269–280, 2004. ISSN 0262-8856. Citation on page 37.

WEDEL, A.; MEIßNER, A.; RABE, C.; FRANKE, U.; CREMERS, D. Detection and segmentation of independently moving objects from dense scene flow. In: **Energy Minimization Methods in Computer Vision and Pattern Recognition**: Springer Berlin Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5681). p. 14–27. Citation on page 51.

XIAO, L.; DAI, B.; LIU, D.; HU, T.; WU, T. Crf based road detection with multi-sensor fusion. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2015. Citations on pages 67 and 79.

XIONG, H. Y.; ALIPANAHI, B.; LEE, L. J.; BRETSCHNEIDER, H.; MERICO, D.; YUEN, R. K. C.; HUA, Y.; GUEROUSSOV, S.; NAJAFABADI, H. S.; HUGHES, T. R.; MORRIS, Q.; BARASH, Y.; KRAINER, A. R.; JOJIC, N.; SCHERER, S. W.; BLENCOWE, B. J.; FREY, B. J. The human splicing code reveals new insights into the genetic determinants of disease. **Science**, v. 347, n. 6218, 2015. Citation on page 25.

YUN, S.; GUO-YING, Z.; YONG, Y. A road detection algorithm by boosting using feature combination. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2007. p. 364–368. Citation on page 38.

ZHOU, S.; GONG, J.; XIONG, G.; CHEN, H.; IAGNEMMA, K. Road detection using support vector machine based on online learning and evaluation. In: **IEEE Intelligent Vehicles Symposium (IV)**, 2010. p. 256–261. Citation on page 38.

APPENDIX

A

PUBLICATIONS

The list below presents all articles/papers published during the doctorate program.

1. MENDES, C. C. T.; FREMONT, V. ; Wolf, D. F. . Exploiting Fully Convolutional Neural Networks for Fast Road Detection. In: IEEE International Conference on Robotics and Automation (ICRA), 2016.

2. MENDES, C. C. T.; FREMONT, V. ; Wolf, D. F. . Vision-Based Road Detection using Contextual Blocks. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2015, Hamburgo. 7th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, 2015.

3. SOUZA, JEFFERSON R. ; MENDES, CAIO C. T. ; GUIZILINI, VITOR ; VIVALDINI, KELEN C. T. ; COLTURATO, ADIMARA ; RAMOS, FABIO ; WOLF, DENIS F. . Automatic detection of Ceratocystis wilt in Eucalyptus crops from aerial images. In: IEEE International Conference on Robotics and Automation (ICRA), 2015, Seattle. Proceedings IEEE International Conference on Robotics and Automation, 2015.

4. MENDES, CAIO CÉSAR TEODORO; OSÓRIO, FERNANDO SANTOS ; WOLF, DENIS FERNANDO . Real-time obstacle detection using range images: processing dynamically-sized sliding windows on a GPU. Robotica (Cambridge. Print), v. 1, p. 1-16, 2015.

5. FERNANDES, LEANDRO C. ; SOUZA, JEFFERSON R. ; PESSIN, GUSTAVO ; SHINZATO, PATRICK Y. ; SALES, DANIEL ; MENDES, CAIO ; PRADO, MARCOS ; KLASER, RAFAEL ; MAGALHÃES, ANDRÉ CHAVES ; HATA, ALBERTO ; PIGATTO, DANIEL ; BRANCO, KALINKA CASTELO ; GRASSI, VALDIR ; OSORIO, FERNANDO S. ; WOLF, DENIS F. . CaRINA Intelligent

Robotic Car: Architectural Design and Applications. Journal of Systems Architecture, v. 6, p. 608, 2014.

6. OLIVEIRA, M. D. C. ; LA SCALEA, R. A. ; PONTI, M. ; SOUZA, J. R. ; MENDES, C. C. T. ; COLTURATO, A. B. ; NAGLE, F. B. ; FURTADO, E. L. ; KAWABATA, C. ; BRANCO, K. R. L. J. C. ; WOLF, D. F. . Segmentação e contagem de árvores em plantações de eucaliptos utilizando imagens aéreas. In: SIAGRO 2014, São Carlos. Anais do SIAGRO, 2014.

7. MENDES, C. C. T.; OSORIO, F. S. ; Wolf, D. F. . An Efficient Obstacle Detection Approach for Organized Point Clouds. In: IEEE Intelligent Vehicles Symposium (IV), 2013, Gold Coast. Proceedings of the IEEE Intelligent Vehicles Symposium, 2013.

8. MENDES, C. C. T.; OSORIO, F. S. ; Wolf, D. F. . Stereo-Based Autonomous Navigation and Obstacle Avoidance. In: IFAC Intelligent Autonomous Vehicles (IAV), 2013, Gold Coast. Proceedings of the IFAC Intelligent Autonomous Vehicles, 2013.

9. SANTOS, E. B. ; MENDES, C. C. T. ; OSORIO, F. S. ; Wolf, D. F. . Bayesian Networks for Obstacle Classification in Agricultural Environments. In: IEEE Conference on Intelligent Transportation Systems (ITSC), 2013, The Hague. Proceedings of the IEEE Conference on Intelligent Transportation Systems, 2103.

10. MENDES, C. C. T.; SHINZATO, P. Y. ; Wolf, D. F. ; OSORIO, F. S. . Detecção de Obstáculos em Tempo Real para Sensores de Profundidade Utilizando uma GPU. In: XIX Congresso Brasileiro de Automática, 2012, Campina Grande. Anais do XIX Congresso Brasileiro de Automática, 2012. p. 1000-1007.