Dissertation

Learning Context For Semantic Segmentation And Applications

Vladimir Haltakov

Technische Universität München
Department of Computer Science
Chair for Computer Aided Medical Procedures and Augmented Reality

campar.cs.tum.edu

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik
Computer Aided Medical Procedures & Augmented Reality / I16

Learning Context For Semantic Segmentation And Applications

Vladimir Haltakov

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. M. Nießner
Prüfer der Dissertation:	1.	Priv.-Doz. Dr. S. Ilic
	2.	Prof. Dr.-Ing. D. Burschka
	3.	Prof. Dr. A. Geiger, Universität Tübingen

Die Dissertation wurde am 29.03.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 23.07.2018 angenommen.

Abstract

Nowadays, cameras are an integral part of many devices and systems - from mobile phones to autonomous vehicles and from medical robots to surveillance cameras. While an active field of research, the task of understanding a camera image still poses significant challenges and is not solved in general. A key in interpreting camera images correctly is not to focus on individual image areas or objects, but to use context from the whole image in order to resolve ambiguities - something that we humans are very good at.

This thesis is built around the idea of learning context - we present several novel methods that extend the traditional semantic segmentation algorithms in order to allow them to incorporate context relations. We present a new classifier-based pairwise potential that can be integrated in a CRF segmentation framework, extending its abilities to learn local context relations. For learning both local and global context, we presented another method, relying on a classifier chain and novel feature formulation based on geodesic neighborhoods. We also show how this model can naturally handle both texture and 3D information and can also run in real-time.

Another approach for providing semantic segmentation methods with additional information is synthetic ground truth data. We present a generic framework for generating high-quality ground truth data, including camera images, pixelwise semantic labels, depth and optical flow data, based on a driving simulator.

Finally, we show how the proposed methods can be used in two real world automotive applications, in which semantic segmentation and a good context understanding are very important - parking space detection and traffic lights detection.

Zusammenfassung

Heutzutage sind Kameras ein wesentlicher Teil von vielen Geräten und Systemen - von Handys bis autonome Fahrzeuge und von Medizinroboter bis Überwachungskameras. Obwohl die Bildverarbeitung ein aktives Forschungsgebiet ist, stellt das Problem ein Kamerabild zu interpretieren immer noch erhebliche Herausforderungen dar und ist im Grunde noch nicht gelöst. Entscheidend dabei ist nicht nur einzelne Bildbereiche oder Objekte zu betrachten, sondern auch den Kontext aus dem ganzen Bild zu nutzen um Mehrdeutigkeiten aufzulösen - etwas was wir Menschen sehr gut können.

Diese Arbeit basiert auf der Idee Kontext zu lernen - wir präsentieren Methoden, die die traditionellen Algorithmen für semantische Bildsegmentierung erweitern, damit diese Kontextbeziehungen einbeziehen können. Wir stellen eine neue klassifikatorbasierte Potentialfunktion vor, die in einem CFR Frameworks integriert werden kann, um das Model zu ermöglichen lokale Kontoüberziehungen zu lernen. Um sowohl lokalen als auch globalen Kontext zu lernen, präsentieren wir eine weitere Methode, die auf einer Kette aus Klassifikatoren aufbaut. Ein wichtiger Teil dabei ist auch eine neue Formulierung der Merkmalsverktoren, die auf das Konzept einer geodätischen Nachbarschaft basiert. Wir zeigen, dass dieses Modell sowohl Bilder als auch dreidimensionale Daten in Echtzeit verarbeiten kann.

Eine weitere Möglichkeit Methoden für semantische Bildsegmentierung zu verbessern ist diese über künstliche Daten zu erweitern. In dieser Arbeit präsentieren wir ein generisches Framework zur Generierung von hochqualitative Ground Truth Daten, wie Kamerabilder, pixelgenaue semantische Labels, Tiefendaten und optischer Fluss, das auf einem Fahrsimulator basiert.

Schließlich zeigen wir wie die vorgestellten Methoden in zwei Fahrerassistenzfunktionen angewandt werden können: Parkplatzerkennung und Ampelerkennung. Dabei ist eine semantische Bildsegmentierung mit der Fähigkeit Kontext zu verarbeiten essenziel.

Acknowledgements

First, I would like to thank Slobodan Ilic for supervising my thesis and for the great support and guidance over the years. He was always open for discussion and provided many useful insights and ideas.

I would also like to express my gratitude to my supervisor at BMW Christian Unger for the many hours of interesting discussions, both technical and philosophical, and for his helpful guidance during my time as a PhD student and after that.

I would like to give special thanks to my colleagues at CAMP, Vasilis Belagiannis, David Tan, Paul Huang and Loic Peter for always being ready to provide feedback and exchange ideas. I also thank the more experienced PhD students that were a great help with their expertise especially in the beginning - Stefan Holzer, Stefan Hinterstoißer and Bertram Drost. Not less I am thankful to my fellow PhD students at BMW Joé Lallemand and Jöran Zeisler for the many fruitful discussions and shared thoughts and to all my students over the years: Caner, Stefan, Max, Alex, Dominik and Jakob. Moreover, I am also grateful to Mario Nagelstrasser and Christian Discher for their support and to BMW for funding my PhD program.

I am also indebted to all the people that supported me personally and especially my wife Maria who was by my side during the whole time and also during all the intense and sleepless nights of paper writing. I also thank my parents for their unconditional support, encouragement and advice. In addition, I should thank Mariela for proof reading every one of my papers. Finally, I would like to express my gratitude also to all my friends, family and colleagues.

Contents

1 Introduction

Even the very first computers were already much better than humans in performing calculations. Over the last decades, the computing power has increased exponentially and computers have become an unthinkable part of our life when it comes to storing and searching huge databases of information and performing complex computations and simulations. However, other fundamental tasks that are easy for humans, still remain difficult for computers to solve. The task of making a computer to "see" is one of them. While solving this task was initially thought to be doable in the course of a summer project [104], today it is still an unsolved problem and an active research field.

Nowadays, computer vision is getting increasingly important for many applications. In recent years, the prices of cameras and computing hardware have become so low, that many of the devices surrounding us are equipped with at least one camera: from mobile phones, TVs and laptops to cars, robots and factory production lines. All those sensors provide huge amounts of information with estimated 350 million photos uploaded to Facebook [143] and 720,000 hours of video uploaded to YouTube [112] every day. Therefore, computer vision methods are required for many applications in order to automatically analyze the image data and extract information from it. Such methods find applications in many industries like for example in consumer electronics, medical imaging, quality inspection, automotive and much more. While the methods in this thesis are not limited to any specific application, the experiments are focused on driver assistance and autonomous driving, which we discuss in more detail in Section 1.2.

While computer vision has many different aspects, one of the generic approaches to interpret the content of an image is semantic segmentation. The semantic segmentation problem is defined as segmenting an image into regions with a certain semantic meaning. The semantic segmentation of an image gives information about the objects contained in the image and where they are located in it, serving as the basis for further processing depending on the application. Semantic segmentation methods can deliver a very generic description of the scene. They have the advantage over more traditional object detection methods, that they can naturally describe not only objects with well-defined shapes like cars, dogs or people, but also parts of the image taken by the sky or the ground. Furthermore, using semantic segmentation the exact outline of the object can be obtained compared to a bounding box delivered by standard object detection methods.

1.1 Using Context

Semantic segmentation methods typically consist of multiple processing steps in order to produce the final semantic segmentation, like for example feature extraction, classification or regularization. Therefore, different research works usually focus on different parts of the pipeline in order to improve the performance: designing more discriminative features,

a) No features b) Unusual parts c) Reflections d) Low resolution e) Ambiguous shape

Figure 1.1: Examples of ambiguities that can be solved using context information. In the images on the top row it is difficult to recognize the correct object, but this is easy to do when looking at the images on the bottom row, where the same region is put in context.

building better and more general classifiers or integrating various models as constraints. In this thesis, we focus on developing methods that can learn context.

The Oxford English Dictionary [35] defines the term **context** as:

> *The circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood...*

This applies to images as well – in many cases, looking at an isolated part of the image, it is not possible to determine the object or part of the object in it. This can have different reasons and we show some examples in Figure 1.1.

- **Lack of distinctive features** An isolated part of a car is shown in Figure 1.1 a). Due to the lack of distinctive features, it is impossible to tell the type of the object by looking alone at the image above. The object may as well be a part of a building or a part of the road. However, once we see the full image, determining that this is a part of a car is trivial.

- **Unusual object parts** Looking at the cut-out in Figure 1.1 b) it is easy to believe that this is a house, however, the whole image reveals, that this is a part of a boat. While the structure is unusual, it is not difficult for a human to determine it as a part of a boat once we can see that it is floating on a river.

- **Reflections** While the small image in Figure 1.1 c) looks like a tree, it is actually a reflection on the bonnet of the car. While one can argue that two classes can be assigned to this region of the image, for most applications it will be more important to identify it as a car instead of a tree.

- **Low resolution** When objects are far away from the camera, the details of their shapes are lost due to the discretization of the image. Therefore, just by looking at the cut-out in Figure 1.1 d) it is very difficult to tell that this is a car. However,

when it is put on the street, surrounded by other cars and buildings, the object can easily be identified as a vehicle.

- **Ambiguous shape** The object in Figure 1.1 e) can easily be identified as a yield traffic sign when observed without any context, while it is actually a part of a red bridge, which has a very similar shape.

All the examples above are very difficult to classify correctly without looking at the bigger picture and using the context to identify the exact type of the object. If a semantic segmentation method is able to learn those context relations it will be able to resolve those ambiguities.

In this thesis, we distinguish between three types of context relations: local context, global context and co-occurrence.

- **Local context** A good example of local context relations are Figure 1.1 a) and c) - it is enough to just look directly around the region of interest in order to solve the ambiguity, because the surrounding is discriminative enough. The short-range context relations apply when a part of an object is surrounded by other parts of the same object.

- **Global context** Figure 1.1 b) is an example of how we use global context relations to resolve an ambiguity. Looking just around the vicinity of the cut-out region we will see that it is surrounded by other house-like structures and trees which would confirm the hypothesis that the object is a house. We need to see larger parts of the image in order to understand that the whole object is actually floating on a river and is therefore a boat. Similarly, we can identify the car in Figure 1.1 d) only after we interpret the whole image - that there is a road going away from the camera, buildings on the sides and another vehicle on the road.

- **Co-occurrence** Some objects tend to appear much more often together in the same image as others. For example, birds usually appear in the sky and on trees and very rarely inside of cars, while cars typically appear on the road among buildings and other cars and are very unlikely to be floating on a river. Such co-occurrence probabilities are the reason why we can correctly identify the object in Figure 1.1 b) as a boat and not as a house.

The usage of context has been shown to be an important factor in reaching a good performance for other computer vision problems as well, like for example for object detection and classification [97]. The goal of this thesis is to develop semantic segmentation methods that are able to automatically learn all those kinds of context relations and that are able to address all ambiguities shown above. While more discriminative features and better classification methods can certainly improve the performance of semantic segmentation methods, the examples above are not likely to be solved without incorporating context into the model. While one can integrate explicit models that are valid for certain type of scenes (for example that vehicles always stand on the road in traffic scenes) we aim to design methods that are able to learn those models implicitly and therefore be more general.

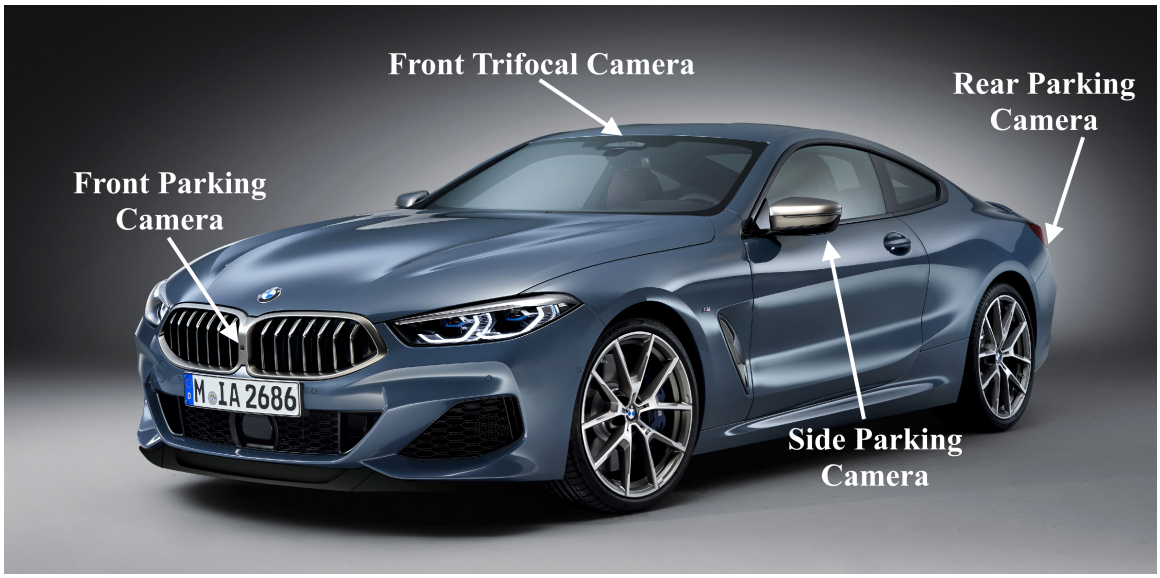


Figure 1.2: An example of a modern BMW 8 series equipped with a front facing trifocal camera and 4 parking cameras. Source: https://www.press.bmwgroup.com.

1.2 Automotive Applications

As discussed in the beginning of this chapter, computer vision has many different applications, but in this thesis, we focus our experiments on the development of methods for driver assistance systems and autonomous vehicles. Many modern cars already come equipped with at least one front facing camera and multiple parking cameras (see Figure 1.2) that provide a wide range of functions that assist the driver in different situations, like for example speed limit information, automatic emergency braking, adaptive cruise control, lane keeping and traffic jam assistance. While there are several other sensor technologies commonly used in production vehicles like RADARs, LIDARs and ultrasonic sensors, cameras have the widest range of applications since they can detect the highest variety of objects and are cheap and easy to integrate. Therefore, cameras are being increasingly adopted by all car manufacturers and are a central sensor for autonomous vehicles.

On the other hand, traffic safety agencies all over the world are continuously developing new assessment procedures and legislation which require the usage of camera systems. The European New Car Assessment Programme (Euro NCAP) and similar institutions in other countries already rate new vehicles based on their automatic braking systems for prevention of accidents with other vehicles or pedestrians [6], which are very hard to pass without a camera system. Further tests for cyclists protection and speed limit assistance are planned to come over the next years [5]. The U.S. Department of Transportation's National Highway Traffic Safety Administration (NHTSA) goes even one step further making backup cameras mandatory on all new vehicles from May 2018 [4].

The increasing requirements on camera systems in production cars and autonomous vehicles requires the development of robust and fast methods for scene understanding. Semantic segmentation methods are particularly useful for interpreting the camera images since they can answer several key questions about the scene:

1. **What dynamic objects are present in the scene?** The detection of vehicles, pedestrians and cyclists is important for both the longitudinal and the lateral control of the vehicle.

2. **What infrastructure objects are present in the scene?** The detection of infrastructure objects like traffic signs and traffic light is required in order to ensure that the vehicle conforms to traffic regulations.

3. **Where can the vehicle drive?** The system needs to detect artificial limitations like lane markings as well as physical delimiters like curb stones, guard rails and concrete barriers for the lateral control of the vehicle.

4. **What type of situation is the vehicle driving in?** Different systems need to react differently depending on the current situation, like for example in the city, on the highway or in a construction zone.

It is important to note that even though standard object detection methods are well suited to address problems 1. and 2., this is not the case for problems 3. and 4. Putting the computer vision problem in a semantic segmentation framework allows us to use the same methods in order to answer the questions above.

This thesis was developed in the context of automotive applications for advanced driver assistance systems (ADAS) and therefore concentrates on the challenges that arise in this domain. However, all of the developed methods are generic and do not focus explicitly on the automotive domain. Therefore, all the methods presented in the thesis and can be applied in other contexts as well.

1.3 Contributions

In this section we summarize the contributions that we have made in the course of this thesis, consisting of several new methods for learning context relations for semantic segmentation, two real-world applications in the driver assistance domain and two datasets that have been made public.

1. **Learned edge potentials for pairwise CRFs.** We model a new type of pairwise Conditional Random Field potentials that can learn local context relations. We train a classifier to determine the type of the edge between two neighboring cells and we show how the output of this classifier can be integrated into the energy function of a pairwise CRF model. We also introduce a new training procedure for the parameter estimation of the resulting energy function.

   We show that this model is able to significantly outperform standard pairwise CRF methods based on the contrast sensitive Potts model and even some higher-order models. We also show that the resulting model is able to learn local context in order to preserve small objects that tend to be smoothed away by the standard regularization approaches.

2. **Two-stage classification framework based on geodesic neighborhoods.** Our two-stage classification framework for multi-class image labeling is developed around the concept of geodesic pixel neighborhoods.

   We first introduce the concept of a pixel neighborhood and discuss several ways to define it based on some standard unsupervised segmentation methods. We then present a novel use of the geodesic distance transform for the definition of a local geodesic neighborhood that aligns well to object boundaries and is able to capture local context relations along with an efficient algorithm to compute the geodesic neighborhoods. In order to enable our method to learn global context and co-occurrence probabilities, we also formulate a global geodesic rays neighborhood.

   In order to integrate the geodesic neighborhoods in a classification framework, we present a new normalized voting histogram feature that is able to summarize the information of the neighborhood in an efficient way.

   We perform extensive evaluation of our method on six public datasets and show that our method is able to deliver results equal or better than the state-of-the-art. We also analyze in detail how different parts of our method work and show that it is able to capture both local and global context relations as well as to make use of co-occurrence information. We also directly compare our method to a number of closely related methods and discuss why our approach is able to achieve higher accuracy.

   We also show that our method can be optimized to run in real-time without the usage of a GPU, at the cost of only minimal decrease in performance.

3. **Usage of depth data.** We explore several ways to make use of depth data for semantic segmentation.

   We present a novel use of the 2D Walsh-Hadamard transform [63] on the 3D representation of the scene to compute features used by a unary classifier. While this transform is traditionally used in the 2D texture domain, we show how it can be also efficiently applied to depth data.

   We also show that depth data can also be naturally integrated in our neighborhood classification framework by formulating another type of geodesic neighborhood that aligns to depth discontinuities instead of texture edges.

   During our analysis, we show that our framework allows depth information to be naturally integrated and that by combining 2D and 3D information we can achieve better results than using each of the modalities separately.

4. **Framework for generation of synthetic data.**

   We present a framework for generation of ground truth data for semantic segmentation, depth estimation and optical flow estimation based on an open-source driving simulator called VDrift [1].

   We describe the changes to the simulator and its rendering engine that allow us to extract camera images along with highly accurate depth maps, optical flow maps and pixelwise semantic labels. We use the framework to generate two ground truth dataset containing a large number of images in many different traffic scenarios.

We show how this framework can be used to compare different methods for semantic segmentation and how synthetically generated data can be used in certain scenarios to train a system that performs as good as a system trained with manually labeled data.

5. **Parking space detection application.** We show how our neighborhood classification framework can be used in a real-time application for the detection of parking spaces on the side of the road.

   We apply our semantic segmentation methods on image from a camera mounted sideways in the side mirror of a vehicle and segment the images in order to find the regions belonging to the free space where the car can drive and obstacles like parked cars, curbs and other objects. We integrate this segmentation over multiple frames in order to create a map of the surroundings in which we can detect regions that are free parking spaces and also identify the number of parked cars. The application is able to run in real-time on an office workstation PC without GPU support.

6. **Traffic lights detection application.** Another application that we present deals with the detection of traffic lights, which is an important basis detection function for multiple safety and comfort driver assistance functions and autonomous vehicles. In comparison to the majority of methods in the literature, our approach is able to handle both day and night conditions in a unified manner.

   We again use our neighborhood classification framework to segment the camera image into traffic light candidate regions and background. The resulting segmentation is trained separately for day and night scenes, which pose fundamentally different challenges. We show, however, that after that, working with the segmented images allows to use the same standard shape classification and tracking methods both at day and night conditions.

   We evaluate our method on a publicly available dataset at day and on a new dataset for traffic lights detection containing both day and night scenes that we also make public for the community. The presented application runs in real-time without the usage of special hardware like a GPU.

1.4 Publications

The papers listed below have been published during the course of this thesis.

Vladimir Haltakov, Heidrun Belzner, and Slobodan Ilic. **Scene Understanding From a Moving Camera for Object Detection and Free Space Estimation**. In Intelligent Vehicles Symposium, Alcalá de Henares 4.-6. June 2012. [54]

Vladimir Haltakov, Christian Unger, and Slobodan Ilic. **Framework for generation of synthetic ground truth data for driver assistance applications**. In German Conference on Pattern Recognition, 3.-6. September 2013. [56]

Vladimir Haltakov, Christian Unger, and Slobodan Ilic. **Geodesic pixel neighborhoods for multi-class image segmentation**. In British Machine Vision Conference, 1.-5. September 2014. [57]

Vladimir Haltakov, Jakob Mayr, Christian Unger, and Slobodan Ilic. **Semantic segmentation based traffic light detection at day and at night**. In German Conference on Pattern Recognition, 7.-10. October 2015. [55]

Vladimir Haltakov, Christian Unger, and Slobodan Ilic. **Geodesic pixel neighborhoods for 2D and 3D scene understanding**. In Computer Vision and Image Understanding, 2016. [58]

1.5 Outline

In Chapter 1 we introduce the general goals of the thesis and summarize our contributions. In Chapter 2 we discuss the standard methods of the semantic segmentation pipeline including feature extraction, classification and CRF models. Chapter 3 presents an overview of the state-of-the-art and methods related to our research. In Chapter 4 we introduce our new learned pairwise potentials for CRF based semantic segmentation. In Chapter 5 we describe our two-stage classification framework based on geodesic neighborhoods. Chapter 6 describes our framework for generation of synthetic data. In Chapter 7 and Chapter 8 we present two applications based on our neighborhood classification framework: parking space detection and traffic light detection respectively. We conclude in Chapter 9.

2 Background

In this chapter we present an overview of the theoretical background for semantic segmentation methods. Here, we discuss in detail some of the more frequently used general approaches to the problem, while in Chapter 3 we review the most relevant recent works in the field.

2.1 Problem Definition

Given an input image I, the goal is to assign every image pixel a semantic class label from a predefined set of labels $\mathcal{L}$, like for example TREE, ROAD, SKY or CAR. We denote the set of image pixel intensities as $\mathbf{x}$ and the intensity of each individual pixel i as x_i. In our work, we use the variable x_i to encode both grayscale and color pixels as well as other image modalities, like for example depth images.

Each image pixel is assigned another variable $y_i \in \mathcal{L}$ that encodes the semantic class label associated with this pixel. The semantic class variables of all pixels form the set $\mathbf{y}$.

Given this notation, the semantic segmentation problem can be expressed as the conditional probability of the semantic classes given the whole image:

$$P(\mathbf{y}|\mathbf{x}) \tag{2.1}$$

In order to simplify this probability function, different independence assumptions can be made, resulting in different approaches, which are described in more detail below.

2.2 Independent Variables Assumption

The easiest approach is to assume that all pixels are independent on each other. In this case, the probability distribution can be factorized as:

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}) \tag{2.2}$$

The probability of the type $P(y_i|\mathbf{x})$ can easily be estimated by a classifier that looks at the surrounding of each pixel independently in order to estimate its class. We describe a method based on this assumption in more detail in Chapter 5.

This assumption leads to a great simplification of the problem, but rarely applies in practice, because neighboring pixels usually have the same label. Furthermore, this model cannot include almost any context, because each pixel is classified independently on the others. Relying on this assumption usually produces very noisy results, but is often used as a first step in the segmentation pipeline by many methods.

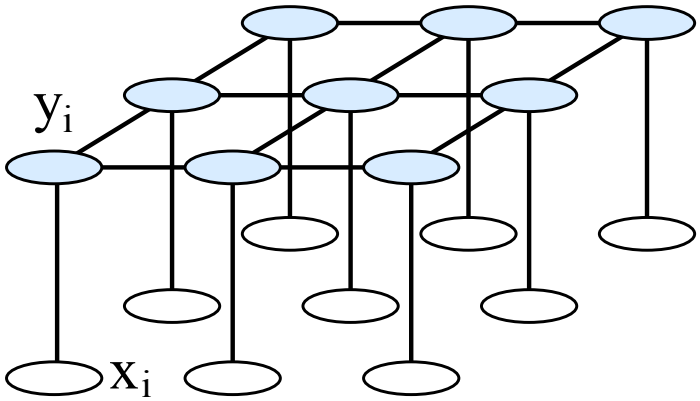


Figure 2.1: Grid based graphical model over an image. The variables denoted with x_i correspond to pixels in the image, while the variables denoted with y_i encode the semantic label for the corresponding pixel. The links in the graph indicate conditional dependence between variables.

2.3 Probabilistic Graphical Models

Another popular method for simplifying the conditional probability distribution $P(\mathbf{y}|\mathbf{x})$ is to use probabilistic graphical models in order to model relationships between variables. This approach has the advantage that the probability distribution can be factorized in such a way, that many well-known and efficient methods can be used to perform inference and parameter estimation. Furthermore, modeling relations between different variables is a way of integrating context relations in the model. On the downside, more complex graph structures tend to quickly lead to generally intractable problems, which require a lot of approximation in order to be applicable in practice.

2.3.1 Markov Random Fields (MRF)

Given an undirected graph G with a set of vertices $\mathbf{y}$ containing the random variables of a probability distribution $P(\mathbf{y})$, this distribution can be factorized as:

$$P(\mathbf{y}) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \phi_C(y_C), \tag{2.3}$$

where $\mathcal{C}(G)$ is the set of all cliques of G, y_C is the subset of $\mathbf{y}$ that belongs to the clique C and ϕ_C is a potential function that defines how strong the dependence between the variables in the clique C are. A clique in a graph is a subset of vertices that are all connected between each other. The distribution is normalized by Z, which is also known as the partition function, and is defined as:

$$Z = \sum_{y \in \mathcal{Y}} \prod_{C \in \mathcal{C}(G)} \phi_C(y_C), \tag{2.4}$$

where y iterates over all possible assignments of the variables $\mathcal{Y}$. This model based on an undirected graph is also known as a Markov Random Field (MRF).

While in theory any undirected graph structure is possible, in this section we will focus on grids since those are the most popular graphs used for images in practice. More complex models that employ more complex graph structures are discussed in Section 2.3.7. Figure 2.1 shows an example of a 4-connected grid over an image. In this graph, there are only cliques of two vertices - the edges of the graph. Therefore, the joint probability distribution $P(\mathbf{y}, \mathbf{x})$ can be factorized as two big products corresponding to the vertical edges between $\mathbf{x}$ and $\mathbf{y}$ and to the edges between the variables in $\mathbf{y}$:

$$P(\mathbf{y}, \mathbf{x}) = \frac{1}{Z} \prod_{(y_i, x_i)} \phi_u(y_i, x_i) \prod_{(y_i, y_j)} \phi_p(y_i, y_j). \tag{2.5}$$

This is a good example of a model that is able to express local context relations, by explicitly modeling the interaction between the neighboring variables. In theory, it is possible information from a variable from one side of the image to be passed along the chain of edges to a variable on the other side and influence its label. However, interactions at such high distances are usually insignificant in practice.

2.3.2 Conditional Random Fields (CRF)

Since the image is given as input and therefore the exact values of the variables in $\mathbf{x}$ are known (the pixel intensities), the joint probability distribution $P(\mathbf{y}, \mathbf{x})$ can be written as a conditional probability distribution, also known as a Conditional Random Field (CRF) [87]:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{(y_i, x_i)} \phi_u(y_i|x_i) \prod_{(y_i, y_j)} \phi_p(y_i, y_j). \tag{2.6}$$

The functions $\phi_u(y_i|x_i)$ and $\phi_p(y_i, y_j)$ are called unary and pairwise potential functions respectively and they model the interactions between the two variables taken as arguments. A common practice is to choose the potential functions in the form of a negative exponent function:

$$\phi(y_i, x_i) = e^{-\psi(y_i, x_i)}. \tag{2.7}$$

In this way, the product in the conditional probability function can be written as a sum, which makes the problem easier to compute in practice:

$$-\log P(\mathbf{y}|\mathbf{x}) = \sum_{(y_i, x_i)} \psi_u(y_i|x_i) + \sum_{(y_i, y_j)} \psi_p(y_i, y_j) + \log Z(\mathbf{x}). \tag{2.8}$$

Finding the labeling $\mathbf{y}$ with the highest probability is then equivalent to minimizing the so called energy function:

$$E(\mathbf{y}, \mathbf{x}) = \sum_{(y_i, x_i)} \psi_u(y_i|x_i) + \sum_{(y_i, y_j)} \psi_p(y_i, y_j). \tag{2.9}$$

Note that the partition function $Z(\mathbf{x})$ does not need to be computed in this case, because it does not depend on $\mathbf{y}$. This is very important in practice, because computing $Z(\mathbf{x})$ would require to iterate over all possible labelings of the image which are $|\mathcal{L}|^N$, where N is the number of pixels in the image. This is obviously an intractable problem, except for very simple cases, which are rarely encountered in practice.

2.3.3 Unary Potential Functions

The unary potential function $\psi_u(y_i|x_i)$ models the probability of a pixel i taking a specific label y_i given the underlying image pixel x_i. Since not only the pixel x_i is given in the conditional probability distribution, but also the whole image $\mathbf{x}$, the function can also be written as $\psi_u(y_i|\mathbf{x})$. This function is usually modeled as the output of a multi-class classifier that assigns a label from $\mathcal{L}$ to an image patch centered around the pixel of interest i.

In this case, any supervised multi-class classifier can be used. The classifier usually does not operate directly on the image intensities, but on a feature vector extracted from the image patch around the pixel i. Different classifiers and features used in practice are discussed in more detail in Section 5.1.4.

2.3.4 Pairwise Potential Functions

The pairwise potential function $\psi_p(y_i, y_j)$ models the relation between two neighboring class variables. A commonly used assumption when working with images is that two neighboring pixels tend to have the same class, since objects in the image usually cover areas much bigger than a single pixel. This can be modeled using the Potts model [14], where the pairwise function is formulated as:

$$\psi_p(y_i, y_j) = \begin{cases} 0 & \text{if } y_i = y_j \\ \theta_p & \text{if } y_i \neq y_j \end{cases} \tag{2.10}$$

Here, a potential defined by the model parameter θ_p, is assigned to neighboring variables that take different labels, which can be seen as a penalty.

This formulation leads to smooth labeled images, but has the disadvantage that object boundaries may not be accurate and small objects, like for example lane markings, can be smoothed over and lost completely. A model that tries to minimize this effect is the contrast sensitive Potts model [15] that also takes the difference in intensity between the two pixels:

$$\psi_p(y_i, y_j) = \begin{cases} 0 & \text{if } y_i = y_j \\ \theta_p + \theta_v e^{-\theta_\beta |I(x_i) - I(x_j)|} & \text{if } y_i \neq y_j \end{cases} \tag{2.11}$$

Here, a penalty is assigned to neighboring variables having different labels, only if they have a similar intensity. Therefore, the smoothing applies less at object boundaries with strong contrast. The model parameters θ_p, θ_v and θ_β define the exact amount of the penalty depending on the difference in intensity of the pixels. The optimal values for those parameters need to be found during a parameter estimation step.

While these approaches lead to energy functions that can be efficiently optimized, they have the disadvantage, that the same constraint is applied to all parts of the image and to all different combinations of classes. Therefore, in practice those constraints tend to either smooth over small objects or leave a lot of noise in the final labeled image. Therefore, the ability of such models to express context is limited to the local smoothness constraint.

There are also other ways to define pairwise potentials, that employ more complex, data driven models, which are discussed in more detail in Section 4.1.2. We also propose a novel definition of the pairwise potential function that relies on an edge classifier in order

to capture local context and co-occurrence relations. Our method is described in detail in Chapter 4.

2.3.5 Inference

Inference is the process of finding the optimal labeling $\mathbf{y}$, given the input image $\mathbf{x}$. This problem is equivalent to minimizing the energy function E from Equation 2.9. While the function can be easily computed for a given labeling, since the partition function Z does not need to be computed (see Equation 2.9), it is in general nonlinear and therefore difficult to optimize. There are two classes of approaches that find an approximate solution of the minimization problem that are widely used in practice.

Message Passing The main idea of the message passing algorithms or also called belief propagation [105] is to compute the solution of the labeling problem by iteratively passing messages between the neighboring variables in the graph. Each message represents what the node sending the message "thinks" about the label of the node the message is being sent to. The message sent from node i to node j is denoted as m_{ij} and is represented by a vector with so many elements as the number of possible classes. So the l-th element of the message m_{ij} represents the confidence of node i that node j has label l and it can be written as $m_{ij}(l)$ for convenience. From these messages the term called belief $b_j(l)$ can be computed, which represents the confidence of node j having a label l:

$$b_j(l) = \frac{1}{Z_j}\psi_u(l|x_j) \prod_{i \in N_j} m_{ij}(l). \tag{2.12}$$

Here Z_j is a normalization constant, which guarantees that the beliefs for the different labels of one node sum up to 1, $\psi_u(l|x_j)$ is the confidence given by the unary potential for the label l at the current node and N_j is the set of all neighboring nodes of j.

There are two common ways to generate the messages which define the two main algorithms for belief propagation:

1. Max-product belief propagation

$$m_{ij}(l) = \max_l \left[\psi_u(y_i|x_i)\psi_p(y_i, y_j) \prod_{k \in N_i \setminus \{j\}} m_{ki}(l) \right]. \tag{2.13}$$

2. Sum-product belief propagation

$$m_{ij}(l) = \sum_l \left[\psi_u(y_i|x_i)\psi_p(y_i, y_j) \prod_{k \in N_i \setminus \{j\}} m_{ki}(l) \right]. \tag{2.14}$$

The main difference between the two variants is the way that the final result is estimated after enough iterations have passed. The max-product algorithm finds the maximum a posteriori (MAP) estimate:

$$x_i^* = \operatorname*{argmax}_l b_i(l). \tag{2.15}$$

The sum-product algorithm optimizes the minimum mean squared error (MMSE):

$$x_i^* = \sum_l b_i(l) l. \tag{2.16}$$

There are some variations of the algorithms with respect to the message passing schedules: messages can be passed one after the other in a serial manner or all messages can be passed in parallel in each iteration. It is also important to note, that belief propagation computes an exact solution only when the graph is a tree. When the graph contains loops the belief propagation (or also called loopy belief propagation in this case [43]) is not guaranteed to give the optimal result and may not even converge. Nevertheless, the algorithm is shown to provide good results for a lot of practical applications.

Another variant is the sequential tree-reweighted message passing (TRW-S) method [75], which is a more efficient version that also tends to find lower energy solutions than the traditional message passing methods.

Graph Cuts Graph cuts are widely used in binary optimization problems and especially in the field of computer vision. The basic idea is to represent the problem as a weighted graph with two special terminal nodes s and t. The graph is then split into parts using the well-studied min-cut/max-flow algorithm [41], where one half contains the node s and the other half the node t. This separation represents the result of the binary optimization problem [52].

In the case of image labeling, however, the optimization problem is not binary, but multi-class. The so-called move algorithms are a way to represent the multi-class optimization problem as a series of smaller binary problems that can be solved by graph cuts.

One such method is α-expansion [13], which starts with an initial labeling and performs multiple expansion moves until a maximum count is reached or until no further minimization of the energy is possible. At each expansion move each node has to choose to change its label to α or to keep its old label, which is a binary optimization problem.

Another similar algorithm is the α-β swap algorithm [15]. At each iteration, it tries to separate the nodes with labels α from the nodes with labels β for all possible α-β combinations. The so formulated binary problem is then solved using graph cuts.

There are also extensions of those methods for higher-level graphs with specific energy function formulations [74, 83].

2.3.6 Parameter Estimation

As discussed above, the unary and pairwise potential functions, usually rely on a number of parameters that need to be estimated. This is typically done in an offline training step using a set of training data. In this way, the optimal values of the parameters, given the training data can be found. For this, we need to find the set of parameters θ that maximizes the conditional probability distribution $P(\mathbf{y}|\mathbf{x})$ over all samples from the training dataset. Taking the parameters θ into account it can be written as:

$$-\log P(\mathbf{y}|\mathbf{x}, \theta) = \sum_{(y_i, x_i)} \psi_u(y_i|x_i, \theta) + \sum_{(y_i, y_j)} \psi_p(y_i, y_j|\theta) + \log Z(\mathbf{x}, \theta). \tag{2.17}$$

Here, in contrast to the inference problem, we need to also compute the partition function $Z(\mathbf{x}, \theta)$, because it depends on θ. Since computing $Z(\mathbf{x}, \theta)$ is an intractable problem in practice we need to rely on strong assumptions in order to be able to perform parameter estimation in reasonable time.

If the training dataset is denoted as $\mathcal{D} = \{(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\}$ the optimal parameters can be determined by maximizing the log likelihood of the training data:

$$l(\boldsymbol{\theta}) = \sum_k \left[\sum_i \psi_u(y_i^{(k)} | x_i^{(k)}, \theta) + \sum_{(i,j)} \psi_p(y_i^{(k)}, y_j^{(k)}, \theta) \right] + \sum_k \log Z(\mathbf{x}^{(k)}, \boldsymbol{\theta}). \tag{2.18}$$

In order to avoid overfitting usually a regularization term is added to the objective function:

$$l(\boldsymbol{\theta}) = \sum_k \left[\sum_i \psi_u(y_i^{(k)} | x_i^{(k)}, \theta) + \sum_{(i,j)} \psi_p(y_i^{(k)}, y_j^{(k)}, \theta) \right] + \sum_k \log Z(\mathbf{x}^{(k)}, \boldsymbol{\theta}) - \sum_{t=1}^{|\theta|} \frac{\Theta_t^2}{2\sigma^2}. \tag{2.19}$$

This regularization term is equivalent to performing a maximum a posteriori estimation of the parameters with a Gaussian prior with mean 0 and covariance σ^2. The constant σ encodes the penalty of the parameters becoming too large. Regularization is especially important, when the number of parameters is big.

This objective function is convex and can be optimized by standard techniques like preconditioned conjugate gradient or limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [100, 93]. The actual choice of the optimization routine has no big influence on the accuracy of the training, but on the time needed for the optimization to converge. Even though the training is an offline phase and is done only once, the training time is still important in the case of big datasets and high-dimensional problems like those in computer vision.

Because of the convexity of the function, a solution can be found relatively easy, however, computing the function value involves the computation of the partition function $Z(\mathbf{x}, \boldsymbol{\theta})$ for every training sample. As mentioned above, this is infeasible for any practical problem and therefore, an approximation is needed.

One of the reasons for the complexity of the normalization function is, that for each clique it iterates over all possible assignments over all variables in the model. One idea for simplifying the computation is to iterate only over the neighboring cells of each variable. The pseudolikelihood method [12] uses only the Markov blanket (in the case of a grid this is the set of all neighboring variables) of each node instead of the whole graph. This is equivalent to splitting the graph into smaller pieces, training them separately and then combining the results. In this case, the normalization is done only locally over the different factors instead of globally and therefore only a local maximum for the parameter values can be found. Nevertheless, pseudolikelihood is shown to work very well in practice.

Another method for estimating the partition function, called piecewise learning [136], extends the pseudolikelihood model, but it splits the graph into even smaller pieces - one per clique. This avoids sharing the parameters between different pieces and results in better performance considering both accuracy and time for training. Another method by the same authors is the piecewise pseudolikelihood training [137], which focuses on further improving training time, but the accuracy can decrease, when the training dataset is big.

Another interesting approach is presented in [139] which uses a graph cuts inference algorithm to iteratively improve the model parameters.

2.3.7 Higher Order CRF Models

Conditional Random Fields containing potentials that cover more than two variables are called higher-order CRFs. Such models usually build on top of a pairwise CRF by including one or more additional potential functions that model the relation between multiple variables. Such models have the advantage that they can express more interactions between bigger image regions and therefore model context relations better. Such models, however, become very expensive to compute using the standard methods, so they usually need some specific optimization technique that is applicable only to a very specific class of potential functions. Some recent works that introduce such higher-order models are reviewed in detail in Section 3.1.2.

An extreme case of a higher-order CRFs are fully connected CRFs, where every two variables are connected between each other. Such models are in general even more computationally expensive than higher-order CRFs, but are in theory able to model much more complex relations. In order to be applicable in practice, some approximation methods exist that reduce the complexity of the model optimization and can lead to good results. We review some of those methods in Section 3.1.3.

2.4 Classifier-Based Models

Alternatively, to the probabilistic graphical models approach, there are many methods that use one or multiple classifiers to determine the class of each pixel. Using a single classifier is similar to the independent variables model and to using a CRF model with unary potentials only. However, multiple classifiers can be applied in a chain or hierarchically on the image in order to consider interactions between different parts of the image.

The main idea behind many classifier-based methods is to form a chain of multiple classifiers in such a way that each classifier uses the output of the previous one in its input. There are many different ways to achieve this, relying on different features and sampling strategies. Some of the recent classifier-based methods are discussed in detail in Section 3.2.

One of the main contributions of this thesis is a novel classifier-based semantic segmentation framework that relies on a sequence of classifiers that are able to learn both local and global context relations. We describe our method in detail and present an extensive evaluation and comparison to other state-of-the-art methods in Chapter 5.

2.5 Deep Learning Based Models

In recent years, deep learning methods and especially Convolutional Neural Networks (CNN) have been successfully applied to a variety of problems like image classification [79, 133, 138], object detection [48, 53, 59, 60, 61, 110, 111, 121, 158], action recognition [71, 132], human pose estimation [89] and key-point matching [40]. The main advantage of CNNs in comparison to the classical methods is that they do not rely on hand-crafted features as

input, but they learn the optimal feature representation for the given problem in the form of convolutions, leading to much better performance. Furthermore, deep networks with multiple layers are able to learn concepts with increasing complexity - from simple edges in the first layers to parts or whole objects in the last layers. This performance requires a lot of computing power especially during training, but recent advances in GPU technology makes those computations feasible.

Methods for object detection and classification have largely benefited from the availability of very large ground truth datasets like ImageNet [116] (14M images, 1M of them with bounding boxes), MS COCO [91] (328K images with 2.5K object instances) or Places [161] (2.5M images). This big amount of data allows CNNs to learn generic concepts that generalize well for various applications. If for some problem, a limited amount of training data is available, it is a common practice to use a network pre-trained on ImageNet (or another large datasets) and then only fine-tune the parameters of the network in order to specialize to the specific problem.

Generating ground truth for semantic segmentation problems is significantly more complex than ground truth for image classification or object detection, because each object outline and class must be labeled instead of just one label for the whole image or a bounding box with the corresponding label. For example, labeling a single image in the Cityscapes dataset requires 90 minutes on average [28]. Therefore, datasets focused on semantic segmentation usually contain much less images with fine pixelwise ground truth, like for example Cityscapes [28] (5K images with fine and 20K with coarse annotations), ADE20K (22K images) or PASCAL VOC [37] (10K images).

Deep learning based methods for semantic segmentation are usually based on a classification CNN pre-trained on a large dataset like ImageNet or MS COCO that is modified in some way in order to provide pixelwise segmentation. Methods that rely on classification networks like VGG [133] or ResNet [61] as basis can benefit from the recent advances in the field and start with a network that already performs very well on the image classification tasks and is trained on large amount of data. However, there are problems that arise from some fundamental differences in the challenges of image classification and semantic segmentation.

For image classification, the network should be invariant to the position of the objects in the image in order to generalize well for different situations. Therefore, such networks usually perform operations like pooling and striding of the convolutions which decreases the spatial resolution of the feature maps and makes the network resistant to translation. Those operations also increase the receptive field of each neuron, allowing it to learn more context relations. However, in the case of semantic segmentation, the position and the outline of the objects is exactly what needs to be determined (along with the objects class, of course). Therefore, such architectures require strategies to balance the tradeoff between the ability of the network to learn context relations and the ability to segment object boundaries accurately.

Pooling and striding operations increase the receptive field the pixels in the final output layer, allowing the network to consider more context around the pixel of interest, but this also leads to prediction layers with a lower resolution and therefore a lot of the details are lost and cannot be recovered by the following layers. Decreasing the pooling operations, on the other hand, leads to more details in the final prediction layer at the expense of the receptive field and therefore the ability to learn context.

The strategies to deal with this problem can be broadly divided in two categories: encoder-decoder based methods and methods based on dilated convolutions. Here we describe these two main approaches, while in Section 3.3 we review the most important works in more detail.

2.5.1 Encoder-Decoder Methods

The main idea of this approach is to first take an image classification network and either replace the fully connected layers with convolution layers resulting in a fully convolutional network (FCN) [95] or completely remove them [9]. This so-called encoder network is not enough, because the resulting class predictions from the network (also called heatmap) has a very low resolution in comparison to the original image (32 times in the case of FCN [95]) and provides only very coarse segmentation. Therefore, another network, called the decoder, is appended that upscales the predictions of the encoder to the size of the original image resolution. There are multiple strategies to do this, ranging from simple bilinear interpolation and deconvolution to integration of higher resolution features in a sophisticated manner.

2.5.2 Methods Based on Dilated Convolution

Pooling and striding are important operations for increasing the receptive field of the neurons, but as discussed above, they reduce the resolution of the class predictions. A way to deal with this problem is to use dilated convolutions instead of pooling and striding. The dilated convolution, also called convolution with a dilated filter, is a well-known concept from the field of signal processing used in the *algorithm à trouts* (algorithm with holes) [65, 125] and are also sometimes called atrous convolutions. The formal definition of the discrete convolution and the dilated convolution is given here as described in [157]. The standard discrete convolution operator $*$ is defined as:

$$(F * k)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t}), \tag{2.20}$$

where $F : \mathbb{Z}^2 \to \mathbb{R}$ is a discrete function and k is a discrete filter. In this notation, the discrete dilated convolution operator $*_l$ is defined as:

$$(F *_l k)(\mathbf{p}) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s})k(\mathbf{t}), \tag{2.21}$$

with l being the dilation factor.

In practice, the dilated convolution samples pixels not in the direct vicinity of the pixel of interest, but in a larger radius, allowing it to cover a larger part of the image. Therefore, a dilated convolution has a higher receptive field than the standard one and its size can be controlled by the dilation factor. The authors of [23] show that directly computing features by a dilated convolution with a large receptive field is more efficient and provides denser results than applying pooling, a standard convolution and then a deconvolution.

There are many works that rely on this concept in order to keep the resolution of the class predictions as big as possible in order to avoid the need for a decoder network.

3 Related Work

There is a huge amount of work on semantic segmentation, but here we focus on the methods that have similar goals or use similar approaches as the methods that we present in this thesis. We cluster the related works in three main categories: random fields based methods, classification based methods and deep learning based methods. Additionally, we also review methods explicitly designed for 3D scenes.

3.1 Random Fields Based Methods

As discussed in Section 2.3 one big class of methods is based on Markov Random Fields and Conditional Random Fields. Here we cluster the different models based on the structure of the underlying graph.

3.1.1 Pairwise Models

Pairwise methods model explicit relations only between two adjacent pixels. Due to the simple form of the resulting energy function, such models allow for fast training and inference, but are limited in their ability to learn context relations.

TextonBoost. The TextonBoost method [128] introduces sophisticated and descriptive unary features, called textons. The classifier output based on those features is integrated into a pairwise CRF, that relies on the contrast sensitive Potts model [15] for the pairwise features. This makes training and inference very efficient, but as already discussed, this pairwise term is only used to smooth the segmentation image and cannot learn any context.

Model of Kumar *et al.* Kumar *et al.* [80, 81] propose to use a linear classifier and a dedicated pairwise feature vector μ_{ij} in order to define the pairwise term, which would allow it to be learned from training data and potentially capture context relations. In practice, however, they use a constant pairwise feature vector $\mu_{ij} = 1$ [81], which means that the regularization term is only a class specific constant. In this way, the pairwise term is again only used for smoothing and not for context learning.

Model of Wojek *et al.* The work of Wojek *et al.* [152] extends the pairwise formulation as [80], but they use the difference of the unary feature vectors as the pairwise feature vector μ_{ij}. This is essentially a more sophisticated way to define the difference between pixels instead of only relying on the pixel intensities. Furthermore, Wojek *et al.* train different versions of the pairwise classifier, depending on the position of the pixel pair in the image and the orientation of the edge. In this way, the pairwise classifier is able to learn local

context relations and adjust the penalty of the pairwise term accordingly. However, the used framework is limited to the use of a linear classifier, which is rarely enough to model the complex interactions that arise in the real world.

In Chapter 4 we present a method based on a similar pairwise CRF as the one in [152], which uses a novel pairwise potential function that is able to classify edges in the image with a classifier of arbitrary complexity, allowing our model to learn more complex pairwise interactions.

3.1.2 Higher-Order Models

Modeling more complex pixel or region interactions requires augmenting the pairwise CRFs with higher-order potential functions. Although, more difficult to optimize efficiently, they improve the segmentation performance significantly.

Robust P^n model. The Robust P^n model of [74] introduces a special type of higher-order potentials based on large pixel segments. To generate those segments, an unsupervised method is applied multiple times to the image with different parameters in order to produce segmentations ranging from over-segmentation to under-segmentation. In this way, the higher-order potentials have access to the surroundings of the pixel and can therefore exploit local context information. While the resulting energy function can be efficiently optimized by graph cuts based methods, the parameter estimation in the training phase is difficult in practice and requires exhaustive parameter search on a validation dataset.

Various other works extend the Robust P^n model in different ways. The work of [135] add 3D information by computing structure from motion point clouds from video. The work of [85] uses a classical object detection algorithm in order to improve the detection of the classes that can be detected by it, like for example vehicles and pedestrians.

Associative Hierarchical Random Fields. A further extension of the original Robust P^n model [74] is the Associative Hierarchical Random Fields model [82, 83]. The authors introduce a hierarchical structure on top of the superpixel-based random field of [74] allowing the model to learn co-occurrence information. Furthermore, going up on the hierarchy levels, more distant parts of the image are connected with each other by the graph, which allows context information to be spread further across the image. The authors also propose an inference algorithm to deal with the complex graph structure. The parameters of the model are tuned on validation data. This model also improves the unary classifier by employing several different unary features, which are expressive, but also slow to compute.

Tree Fields. More complex potential functions enable random fields to better model object relations and to improve the segmentation performance, but this usually leads to an increased number of model parameters. This in turn makes parameter estimation and training more difficult and requires significant approximations in order to be tractable. Another problem with the standard CRF and MRF models is that the parameters, e.g. the weights of the different terms of the energy function, depend strictly on the training

dataset for which they are optimized and are fixed for all other input data. However, different input images in the same dataset may require different weights for optimal results. To overcome this problem, methods like Decision Tree Fields [102] and Regression Tree Fields [68] make the parameters dependent on the input image and use a trained tree model to find the best parameters for the given input. This formulation can be seen as a way to learn global context, because the tree classifier considers the whole image, but it only adjusts the weights of the underlying CRF model and does not directly influence the pixel labels. While powerful, these models introduce additional complexity on top of the already complex CRF models and are usually difficult and computationally expensive to train.

Inference machines. An alternative approach is the inference machines method [115], which employs a classifier to learn the messages passed during belief propagation, when performing inference in the CRF model. The authors of [122] present an extension of this idea specialized for 3D point clouds, which is able to learn spatial semantic context from several source regions defined for each point. The source regions cover bigger parts of the 3D space, but they do not adapt to the input structure.

The method we present in Chapter 5 relies on a classification framework instead on a CRF model. We focus on defining features that cover large parts of the image similar to the higher-level CRF potentials. Those features are specifically designed to express both local and global context information and are used by the classifiers to learn the context relations from the data. Furthermore, the classifier handles both the training and the inference phase, while CRF models require different methods during training and evaluation.

3.1.3 Fully Connected Models

In the fully connected CRF model of [78] every two pixels in the image are connected by a pairwise link, resulting in a fully connected graph. While this allows context information to be shared between pixels across the whole image, inference in such graphs would be intractable in general. The authors propose a special pairwise potentials defined as a linear combination of Gaussian kernels that can be optimized very efficiently. However, this also limits the amount of context data shared between pixels at high distance in the image.

The results show very detailed segmentation around the object boundaries, but there are still many regions that are labeled wrong, which could be fixed by using global context. This indicates that, while the proposed pairwise potentials are very efficient in finding the object boundaries, they are limited in their ability to capture long range and co-occurrence context.

3.2 Classification Based Methods

An alternative approach to the CRF based methods is to use a classifier or a sequence of classifiers to directly predict the class of each pixel.

Semantic Texton Forests. The Semantic Texton Forests [127] employ two random forest classifiers. The first one is applied directly on the image and computes novel low-level features that eliminate the need to compute expensive filter bank responses. A bag of semantic textons is then computed in rectangular regions around each pixel which serves as input to the second random forest, enabling it to learn local context relations.

Auto-context. The auto-context method [145] trains a chain of classifiers where each one has access to both the image features and the predictions of the previous classifier at fixed positions around each pixel. In general, this allows the classifier chain to capture both local and global context relations, however, the method only samples predefined locations relative to the pixel of interest, that do not depend on the image itself. Therefore, not all relevant image areas are always considered by the classifier, which limits its ability to learn context. Increasing the number of sample points is a natural way to deal with this problem, but this also directly increases the required computation time.

Stacked Hierarchical Labeling. Similar to auto-context [145], the stacked hierarchical labeling method of [98] employs a series of classifiers based on the concept of stacking [26], but in a hierarchical way. The image is segmented at different levels with different parameters such that on the top level there is only one segment covering the whole image, while on the bottom level the segments are in the form of superpixels. A classifier is trained for each level to predict the proportion of the labels in each segment. The proposed hierarchy is a good way to pass context information at long distances across the image, but it also leads to segmentation that does not always align well to the object boundaries, because local context relations are not modeled explicitly depending on the edges in the image.

Iterative Context Forests (ICF). Another approach using the auto-context [145] idea are the Iterative Context Forests (ICF) of [44, 45]. They are based on the well-known Random Decision Forests, but in each level of the decision tree the class probability map from the previous level is made available to the classifier in the form of additional feature map. This allows the classifier to iteratively learn context by choosing features operating on the class probability maps. Similarly to the Stacker Hierarchical Labeling method [98], the segmentation produces by ICF is not aligned well to the objects boundaries in many cases, because local context is not modeled explicitly.

Entangled Forest. A very similar idea to ICF [44, 45] is the Entangled Forest method [96]. Here, several decision trees are separately trained as it is usually done in a random forest classifiers, but at several levels of the tree, the predictions of the parent nodes are pooled together with the image features.

GeoF. The GeoF model [76] extends the entangled forests [96] by employing the geodesic distance transform to model long range context relations. The basic idea is that points belonging to the same object have a small geodesic distance, meaning that there is no strong edge between them. This helps the random forest classifier to align the predictions well to the object boundaries. However, these features cannot express more complicated relations, like for example, that cars are usually standing on the road.

In Chapter 5 we present an efficient classifier-based method similar to auto-context [145], but our classifier operates on features specifically designed to capture both local context aligned to the object boundaries and global context that can express different relations between classes.

3.3 Deep Learning Methods

As discussed in Section 2.5 there are two main approaches to apply CNNs to the semantic segmentation problem. In this section we review the most recent papers in both categories and comment on their ability to capture context.

3.3.1 Encoder-Decoder Methods

Encoder-decoder based methods typically rely on pooling and striding to achieve larger receptive fields and therefore increase the ability of the network to capture context, which has the negative effect that the resulting segmentation is coarse and lacks details. Therefore, encode-decoder methods need to apply different strategies to recover the details in the final segmentation.

Fully Convolutional Networks (FCN). The fully convolutional networks (FCN) [95, 123] have set the basis for almost all methods using the encoder-decoder approach. The idea is to substitute the fully connected layers of a classical image classification network by convolutional layers that output a coarse heat map for each possible class. This approach is much more efficient than classifying individual image patches centered around each pixel. However, each pooling layer reduces the resolution of the image that is passed through the network and the final output is much coarser than the input (32 times), which requires some upscaling strategy in order to get a detailed segmentation. In the standard FCN, a learnable deconvolution layer is applied to the encoder network output in order to go back to the desired higher resolution. The deconvolution operation can be seen as the inverse of the convolution and is also called backward convolution, upconvolution or a convolution with a fractional stride. Because upscaling the predictions 32 times, even with parameters learned from training data, provides only very coarse results, which lack details, skip connection from several of the previous pooling layers are added to the output layer, recovering some of the details.

U-Net. The U-Net architecture [114] is focused strongly on the medical image domain, where training data is very difficult to obtain. It is similar to FCN [123], but employs a completely symmetrical architecture consisting of an encoder part as in FCN and a decoder part that performs deconvolution. In contrast to FCN, the U-Net architecture mirrors exactly the encoder part, but replaces the 2×2 max pooling layers with 2×2 deconvolution layers. The decoder part of this architecture is more complex than that of FCN and uses more features to perform the upscaling.

DeconvNet. Another architecture similar to U-Net [114] is DeconvNet [101]. Here again, the encoder part is the same as in FCN [123], but the decoder part employs so called unpooling layers, that use the pooling indices chosen in the encoder part symmetric to the max pooling layers, which eliminates the need to learn deconvolution layers. In order to deal with large image sizes, however, DeconvNet does not process the whole image, but only the parts of the images provided by a region proposals algorithm [163] and combines them to generate the final segmentation image. Here the ability of the network to learn global context is limited even more than in FCN, because the network will only consider the information in the proposal window, which ignores large parts of the image.

SegNet. The idea of unpooling using the indices from the pooling layers, similar to DeconvNet [101], is also used in another FCN based architecture called SegNet [9]. In contrast to FCN, however, SegNet removes the fully connected layers of the original classification network completely, which account for around 90% of the network parameters. Therefore, SegNet is more efficient during training and inference, which also removes the need for object proposals as in DeconvNet, while showing results superior to those of FCN and DeconvNet. Those optimizations, however, have no effect on the ability of the network to learn context, compared to FCN.

ParseNet. The authors of [94] extend the FCN [123] model by adding an additional global context feature. It is an image level feature vector computed by applying global average pooling on the feature map of a particular layer and appending this feature vector to each element of the same feature map. The global context feature can be added to each layer of the network and shows significant improvement of the segmentation. This feature gives each neuron access to summarized information from the whole image, which is a good way to capture co-occurrence information, but it ignores the spatial relation between the objects. For example, this global feature can encode the information that cows are usually appear in the same scene as grass, but not that cars are on top of the road and below the sky. The authors of ParseNet [94] show good performance on the PASCAL VOC dataset [37] in which the images usually contain only a two or three object classes and where co-occurrence information is very important. However, as shown in [159], this strategy is not enough to deal with more complex scenes as encountered in the ADE20K dataset [162].

RefineNet. Another variant of the encoder-decoder architecture is RefineNet [90]. It first generates predictions by a ResNet [61] classification network and then continuously increase the resolution of the predictions by refining them with input from the ResNet feature maps at each layer. This refinement is done by dedicated RefineNet blocks that are learned end-to-end with the whole network and that combines multiple paths in the network, usually combining the class predictions and the higher resolution feature maps from the encoder network.

This architecture deals with the problem of the small resolution of the final prediction map by continuously enhancing the predictions with the proposed RefineNet blocks, which allows the network to keep the pooling and striding layers and therefore keep the large receptive field of the network, giving it access to more context information. The authors also employ a block called chained residual pooling that combines features pooled

at different scales in order to capture context relations.

Global Convolutional Network. The authors of [106] approach the tradeoff between receptive field size and the ability to capture fine details by introducing the idea of a Global Convolution Network that compensates the decrease of the spatial feature maps resolution by increasing the kernel size of the following convolutions. The proposed architecture also aggregates the information from multiple levels of the ResNet [61] encoder network to increase the level of details recovered by the network. Furthermore, the authors introduce a residual boundary refinement blocks in order to get sharp edges without the help of CRFs.

Full-Resolution Residual Network (FRRN). The Full-Resolution Residual Network architecture [109] employs a standard pooling and unpooling processing chain (called also stream in the paper) which allows the network to have access to large portions in the image and thus capture context. However, the network features a parallel full resolution residual stream, in which the information always stays at the full input resolution. All blocks in the network operate on both streams simultaneously so that they can combine both the sources.

Even though, encoder-decoder networks can achieve very large theoretical receptive fields, some studies [94, 160] show that in practice the network uses only a small part of the receptive field centered in the middle. This means that such networks only learns local context relations, but not a lot of global context. In Chapter 5 we show that the results produced by a encoder-decoder segmentation network (SegNet [9]) can be further improved by our method that explicitly models global context across the whole image.

3.3.2 Dilated Convolutions Based Methods

In contrast to the encoder-decoder architectures, networks based on dilated convolutions can increase the receptive field of each neuron without sacrificing the segmentation resolution.

DeepLab v2. The method of [22, 23] also known as DeepLab v2, is one of the first architectures that takes an approach different than the encoder-decoder architecture, relying on dilated convolutions instead. The authors take a VGG-16 network [133], remove the last 2 pooling layers of the network and replace the convolution in the following layers with dilated convolution. In this way, the receptive field is enlarged without reducing the resolution. While in theory it would be possible to eliminate all pooling operations in a classification network, the authors of [23] point out that this is very inefficient. This strategy allows the network to compute a final predictions layer with a resolution 8 times smaller than the input image in comparison to 32 times for the original FCN approach [123], those predictions are still too coarse and lack important details. Therefore, the authors apply the well-known fully connected pairwise CRF [78] on a bilinearly upscaled predictions map in a post processing step in order to obtain the final segmentation output.

Another problem addressed [23] is that objects are encountered at multiple scales in the semantic segmentation problem. The authors propose a method called atrous spatial

pyramid pooling (ASPP) which is an efficient variant of processing the same image multiple times at different scales and combining the results. The idea behind ASPP is that multiple dilated convolutions with different rates are computed in parallel on the same image, effectively simulating sampling at different scales.

The combination of dilated convolutions and processing of the image at different scales gives the DeepLab v2 network a large receptive field and access to large image regions, allowing the network to learn context relations in those regions. Analyzing the results of the paper shows, however, the network still has problems with certain regions that need to be corrected by the smoothing of the CRF. Those problems could in general be solved by using more context information.

DilatedNet. The authors of [157] use a similar idea to [23] and replace the last layers (here both pooling and convolution) with dilated convolution layers. Another important extension, is a context module designed specifically to capture context information. It consists of a cascade of dilated convolutional layers with increasing dilation factor that can be plugged to the predictions layer of any segmentation architecture. The authors present this idea as a way to increase the receptive field of the network without reducing resolution and by this capturing global context.

The ability of the network to learn context is improved by the consecutive use of dilated convolutions to enlarge the receptive field of the network. Furthermore, the authors show that the proposed context module is efficient and can be plugged after every network and increases the performance.

DeepLab v3. An extension of the DeepLab architecture [23] is presented in [24], which includes improved ASPP module and a cascade of dilated convolutions in order to remove the need of using a CRF. An important observation in the paper is that having larger rates of the dilated convolutions helps to capture more long range context relations, but it starts causing problems, when the rate is too high, because most of the pixels are then sampled outside of the image and are therefore not valid. The authors also integrate an additional image level feature as in [94] in the ASPP module to deal with this problem. Furthermore, a cascade of dilated convolutions is introduced inside of the ResNet blocks, which is similar to the context module of [157], but it operated on the feature level and not on the final predictions as in [157].

Similar to [94], the image level features in the ASPP are useful for encoding co-occurrence information, but not for spatial context. Otherwise, the ability of the network to capture context is improved by the usage of a dilated convolutions cascade.

PSPNet. The Pyramid Scene Parsing Network (PSPNet) of [159] combines ideas similar to ParseNet [94] and DilatedNet [157] - the authors use a ResNet [61] network modified with dilated convolutions in the last layers as in [24]. The idea of the proposed pyramid pooling module can be seen as extension to the global feature vector of [94]. Here, however, pooling is applied not only on the whole image but also on increasingly smaller regions. This allows the network to capture information not only on image level, but also in different regions around the pixel of interest and therefore learn more powerful context

relations. Furthermore, the authors of [159] propose a deep supervision scheme to improve training of the network.

The good results on several benchmarks show that PSPNet is able to learn context relations that are not limited to co-occurrence, because of the global features computed at multiple scales.

3.4 Models Specialized for 3D Scenes

Several methods in the literature explicitly make use of 3D information in order to improve the segmentation performance. The authors of [119] segment the image into regions based on stixels [107] computed form dense disparity maps, which are then classified into one of several semantic classes. An extension of this model, called Stixmantics [120], introduces a method for enforcing temporal consistency and a CRF model to achieve spatial smoothness. While the stixel representation is well suited for road scenes, it can be used only when 3D information is available and only to recognize classes that stay on the ground plane and have 3D structure. Therefore, this method cannot be used to detect objects like traffic signs, traffic lights or lane markings. Furthermore, since the stixels are a hard constraint on the shape of the segments, if an error occurs while computing the stixel regions, it cannot be corrected at a later stage.

The authors of [86] model the semantic label and the stereo disparity of each pixels jointly in a CRF framework and show that both tasks solved jointly benefit from each other. Another method [84] models the semantic class and the depth jointly only using a mono camera image. The goal is to achieve a 3D normalization of the scene so that the classifier does not need to learn the appearance of the objects at different scales, but only at a certain canonical distance.

In Chapter 5 we also explore ways to integrate depth information into a segmentation framework. We show that 3D data can be used to compute a novel feature for unary classification. Furthermore, we discuss how the 3D information can be used on a higher level in combination with 2D intensities to improve the ability of our method to learn context.

4 Context for CRF Models

In this chapter we focus on the problem of enabling a pairwise CRF model to learn local context relations. While pairwise models are fairly simple and not very expressive compared to higher-order or fully connected CRFs, it is important to note that many of those complex models contain a pairwise term in their energy function [74, 82, 83, 85, 86]. Therefore, all these models could potentially benefit from a more expressive pairwise term that is able to capture local context relations. In fact, all of the models cited above, rely on the fairly simple contrast sensitive Potts model, which has limited ability to express context relations.

Compared to the higher order models, surprisingly little effort has been spent on improving the pairwise potentials of the standard CRF segmentation model. As discussed in Section 2.3.4, the widely used contrast sensitive Potts model [13] penalizes neighboring pixels that have different labels, but similar appearance. This formulation is not able to express any interaction between classes and only relies on contrast differences. Therefore, such models tend to provide good object delineation only when there is a strong edge in the image. Furthermore, pairwise CRFs based on the contrast sensitive Potts model tend to smooth over small objects, which may be important for various applications, like for example lane markings and traffic lights.

We propose a new kind of pairwise term called class sensitive pairwise potentials (CSPP), which is based on a strong, nonlinear classifier that can model complex label interactions from image features. In this way, the proposed pairwise potentials define a class specific regularization constraint that is learned from the training data and that preserves small objects. We compare our method to the standard contrast sensitive Potts model [15] and to the more sophisticated method of [152] which is based on the linear classifier of [81] and we show that our formulation of the pairwise potentials leads to a significant improvement in segmentation accuracy. We also evaluate the class sensitive pairwise potentials on a new dataset for traffic light detection and show that our method is particularly suited for this problem, because it does not smooth over the relatively small traffic lights. A more detailed review of the related methods is presented in Section 3.1.1.

Our main contribution are the new class sensitive pairwise potentials that learn local context relations and the way that they are integrated in the energy function of the CRF.

4.1 CRF Model

In this section we present our class sensitive pairwise potentials (CSPP) and our pairwise CRF model. We also discuss the feature extraction, parameter estimation and inference.

Here, we use the notation introduced in Section 2.3.2, defining a pairwise CRF based on a 4-connected grid with vertices V and edges E. The resulting energy function of the

model can be written as:

$$E(\mathbf{y}, \mathbf{x}) = \sum_{i \in V} \psi_u(y_i|x_i) + \sum_{(i,j) \in E} \psi_p(y_i, y_j). \tag{4.1}$$

4.1.1 Unary Potentials

The unary potentials $\psi_u(y_i|x_i)$ are defined as the probability distribution computed by a JointBoost classifier [142] working on a feature vector $\mathbf{f_u}(x_i)$ computed from the image:

$$\psi_u(y_i|x_i) = P(y_i|\mathbf{f_u}(x_i)) \tag{4.2}$$

The JointBoost classifier [142] is an efficient and fast multi-class variant of the well-known AdaBoost classifier [148]. The original AdaBoost method is a binary classifier that consists of a chain of comparisons between a feature and a specific threshold. Therefore, the typical approach to use AdaBoost for multi-class classification is to train a separate classifier for each class and then apply each one to the feature vector in a one-vs-all manner. The JointBoost method shares the comparison between mutliple classes and is therefore able to provide a much more compact and efficient model. The unary feature vectors used here are described in detail on Section 4.1.3.

4.1.2 Class Sensitive Pairwise Potentials

For the pairwise potentials we again rely on a JointBoost classifier. It is trained on all possible $\frac{n(n+1)}{2}$ symmetric pairs of labels from $\mathcal{L} \times \mathcal{L}$. Analogously to the unary case, we use the probability $P(Y_i = y_i, Y_j = y_j|\mathbf{f_p}(x_i, x_j))$ of assigning labels y_i and y_j to pixels i and j given the pairwise feature vector $\mathbf{f_p}(x_i, x_j)$ to define the potential function. However, in the case of $y_i = y_j$, the pairwise potentials in some sense duplicate the unary potentials and do not provide much more information. Therefore, in this case we do not use the output of the classifier, but a class specific regularization parameter θ_{y_i}, where $y_i \in \mathcal{L}$. The final formulation of our pairwise potentials becomes:

$$\psi_p(y_i, y_j) = \begin{cases} -\theta_{y_i} & \text{if } y_i = y_j \\ -P(Y_i = y_i, Y_j = y_j|\mathbf{f_p}(x_i, x_j)) & \text{if } y_i \neq y_j \end{cases} \tag{4.3}$$

Our formulation differs from the interaction potentials of Kumar *et al.* [81] in two important ways. First, the model of [81] is limited to use a linear classifier, because in the training phase the derivative of the log-likelihood with respect to the classifier parameters needs to be computed, which is not feasible for more complex classifiers. Our formulation is much more general and allows an arbitrary classifier to be used, which means that we can employ a strong, nonlinear classifier (e.g. JointBoost [142]), which is able to learn much more complex local context relations. Second, in the case of equal labels, we do not use the output of the classifier, but a class specific parameter θ_{y_i}. These parameters are estimated from the training data and provide a regularization term that is adjusted to the different classes. We discuss the effect of this term in detail in Section 4.2 and show that our formulation provides a smoothing effect that preserves small regions like lane markings and traffic lights.

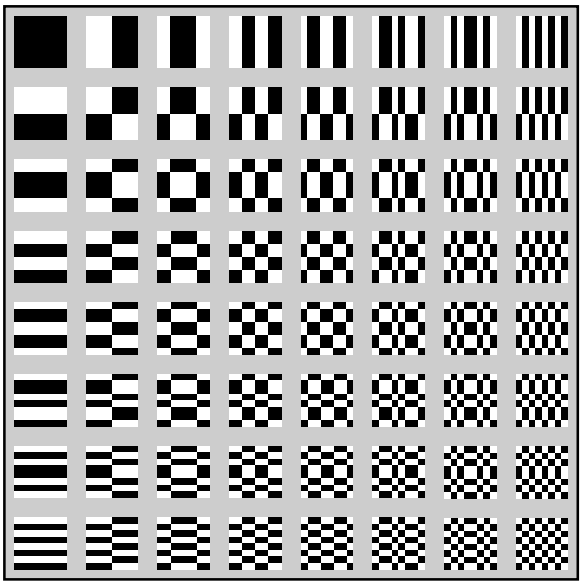


Figure 4.1: The basis vectors of the 2D Walsh-Hadamard transform of order 8. The black squares denote the coefficient -1 and the white squares the coefficient 1.

When we compute the sum over all graph edges, we can split the pairwise potentials into two different terms, depending on whether two pixels have the same labels or whether they differ. Combining the unary term from Eq. 4.2 and the pairwise term from Eq. 4.3 with a weighting parameter θ_u we get the final form of our energy function:

$$E(\mathbf{y},\mathbf{x}) = \overbrace{-\theta_u \sum_{i\in V} P(y_i|\mathbf{f_u}(x_i))}^{\text{unary}} \overbrace{-(1-\theta_u) \sum_{\substack{(i,j)\in E \\ y_i \neq y_j}} P(y_i,y_j|\mathbf{f_p}(x_i,x_j))}^{\text{pairwise}} \overbrace{-(1-\theta_u) \sum_{\substack{(i,j)\in E \\ y_i = y_j}} \theta_{y_i}}^{\text{regularization}}. \tag{4.4}$$

4.1.3 Feature Vectors Computation

In the proposed model we consider two different sets of feature vectors, $\mathbf{f_u}(x_i)$ for the unary and $\mathbf{f_p}(x_i,x_j)$ for the pairwise classifier.

Unary features We use the same unary features as proposed in Wojek *et al.* [152] in order to make a fair comparison between the two formulations of the pairwise potentials. The input camera image is first converted in Lab color space and then the *a* and *b* channels are normalized under the gray world assumption, while the *L* channel is normalized to fit a fixed mean Gaussian distribution.

The features are computed using the 2D Walsh-Hadamard transform [63], which is a discrete approximation of the cosine transform and is very fast to compute, since it involves only addition and subtraction operations (see Figure 4.1). The first 16 coefficients of the Walsh-Hadamard transform are computed at different scales in a window around every pixel separately for each color channel, using the dyadic ordering. As in [152], we also

add the image coordinates of each pixel to the feature vector in order to encode location information.

Pairwise features In the pairwise feature vector we use the difference of the unary features for the two pixels as in [152]. Additionally, we add the image coordinates of the first pixel to the pairwise feature vector along with an identifier specifying if the edge is horizontal or vertical. Note that the method of [152] uses exactly the same information, but they train different classifiers depending on the position and orientation of the edge. Our formulation allows us to include all this information in the feature vector and train only one classifier.

4.1.4 Parameter Estimation and Inference

In the learning phase, we first train the unary and the pairwise classifier separately given a ground truth training dataset. We then have to estimate the weighting parameter θ_u and the $|\mathcal{L}|$ class regularization parameters θ_{y_i}. The energy function of our model (see Eq. 4.4) can easily be transformed to be linear in the model parameters by substituting $\theta'_{y_i} = (1 - \theta_u)\theta_{y_i}$.

As discussed in Section 2.3.6 parameter estimation is a difficult problem for grid-like CRF models, with a lot of variables, such as those used in computer vision, because it requires the computation of the partition function $Z(\mathbf{x})$, which iterates over all possible labelings and is therefore intractable. We tried to approximate the partition function with the piecewise method of [136], but the results were poor. Therefore, we decided to directly minimize the pixelwise error on the set of training images, because the number of parameters of our model is relatively small and their values are between 0 and 1. In order to make the process more efficient, we first assume that all regularization parameters have the same value θ_r. We discretize the parameter θ_u in the range from 0 to 1 with a step of 0.01 and for each possible value we find the best value of θ_r. After that we initialize all of the regularization parameters with θ_r and optimize them using coordinate descent with a momentum term in order to overcome small local minima.

For inference in our model we use max-product loopy belief propagation [105]. We also performed the experiments with the Tree-reweighted Message Passing (TRW-S) method of [75], but it gave slightly worse results. Because of the more complex form of the pairwise potentials, the energy function is not submodular and therefore graph cuts based methods like α-expansion and α-β-swap [15] cannot be applied.

4.2 Results

In this section we present the results of evaluating our method on two challenging datasets. First, we use the publicly available Dynamic Scenes Dataset of [152] and we show that our formulation outperforms even the higher order dynamic CRF model of [152]. Second, we evaluate our method on a specialized traffic light detection dataset in order to show that our method is very well suited when the objects to be detected are very small.

Method	Accuracy
Unary only	85.9%
Potts model	86.3%
Plain CRF model [152]	88.3%
Object CRF model [152]	88.4%
Dynamic CRF model [152]	88.7%
CSPP model	**89.0%**

Table 4.1: Overall pixelwise accuracy of our method compared to the Potts model and the method of Wojek *et al.* [152].

	Void	Sky	Road	Marking	Tree	Grass	Building	Car
Void	0.2	9.7	29.5	2.7	27.3	29.0	0.3	1.4
Sky	0.0	92.4	0.0	0.0	7.4	0.0	0.1	0.0
Road	0.0	0.0	97.3	1.3	0.1	0.6	0.0	0.7
Marking	0.1	0.0	37.5	61.5	0.1	0.5	0.0	0.3
Tree	0.1	1.6	0.3	0.0	92.1	5.6	0.2	0.1
Grass	0.4	0.0	11.2	0.4	7.4	80.4	0.0	0.3
Building	0.0	3.3	0.1	0.0	45.4	2.5	48.7	0.0
Car	0.1	2.5	25.2	2.0	19.7	8.3	0.0	42.2

Figure 4.2: Confusion matrix in percent for the Dynamic Scenes Dataset [152].

4.2.1 Dynamic Scenes Dataset

The Dynamic Scenes Dataset [152] consists of traffic sequences containing images of resolution 752 $\times$ 480 pixels and 8 classes: VOID, SKY, ROAD, MARKING, TREES, GRASS, BUILDING and CAR. We use the same training and test split as the authors in [152] and we also use cells of size 8 $\times$ 8 as nodes in the graph instead of every pixel. The Walsh-Hadamard features are extracted at scales of 8, 16 and 32 pixels as in [152]. The quantitative results are presented in Table 4.1, while Figure 4.3 shows several result images for several methods.

First, we compare to a pairwise CRF based on the contrast sensitive Potts model, which gives only a small improvement over the unary classifier. From the result images in Figure 4.3 we can see that it tends to smooth over small objects like the lane markings. On the other hand, our class sensitive pairwise potentials not only give better overall accuracy, but also preserve the small classes pretty well as can be seen both in the result images and in the confusion matrix in Figure 4.2. Note that a lot of the errors for the lane markings are caused by the discretization of the cells.

On the basis of the Dynamic Scenes Dataset, we can also compare our method to the three CRF models of Wojek *et al.* [152]. We use the same features and classifier for the unary potentials (our own implementation) and we also provide our pairwise classifier with the same information as in [152]. The models differ in the way that the information of the pairwise classifiers is integrated into the energy function and in the classifiers themselves.

The total pixelwise accuracy of our model is 89.0% comparing to 88.3% for the plain CRF model of [152]. This is because our pairwise classifier is able to model more complex local context relations than the linear classifier of [152]. In fact, our formulation outperforms even the more powerful dynamic CRF model (88.7% accuracy) of [152] which uses additional object detectors for cars and makes use of the temporal information in the sequence. Comparing the confusion matrices of both methods, we also see that our model is able to better detect relatively small object classes like lane MARKINGS and BUILDING. The dynamic CRF model of [152] gives significantly better results for the class CAR, which can be attributed to the additional object detector and Kalman filter tracking.

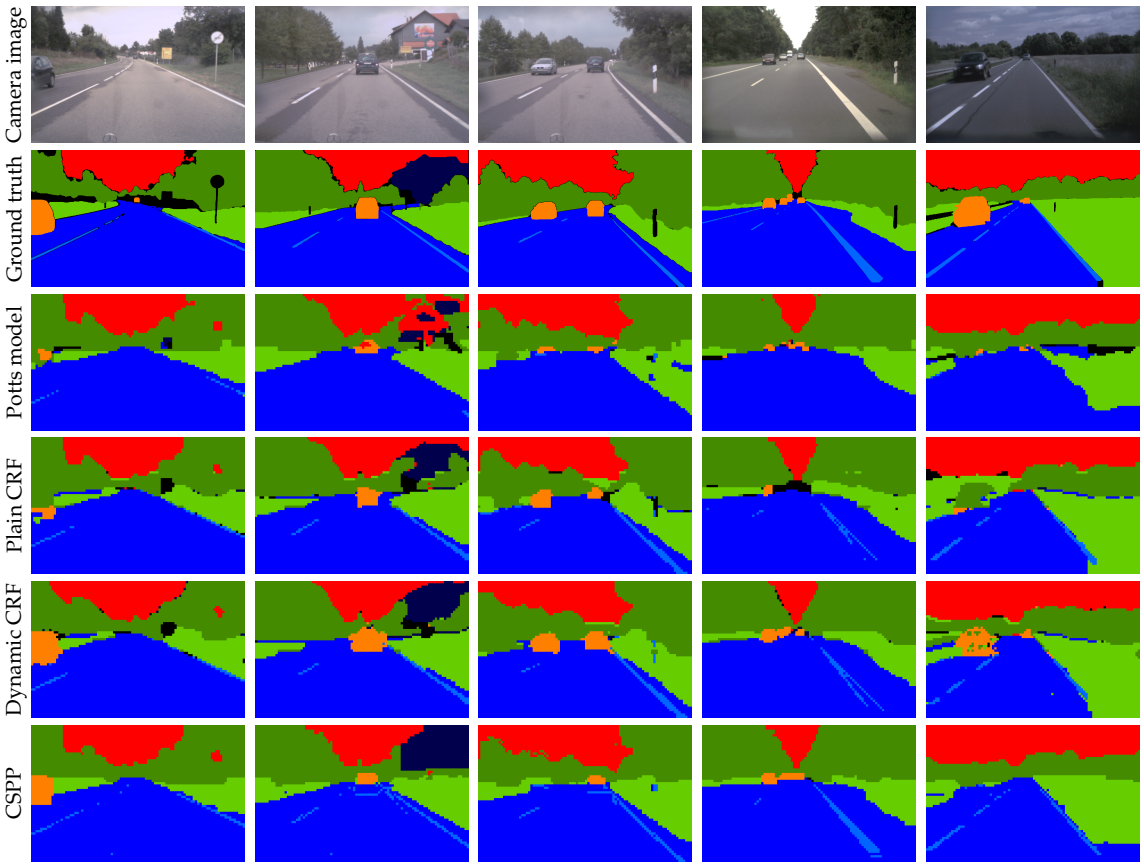


Figure 4.3: Visual comparison of the results of our method compared to the Potts model, the plain CRF model and the dynamic CRF model of [152].

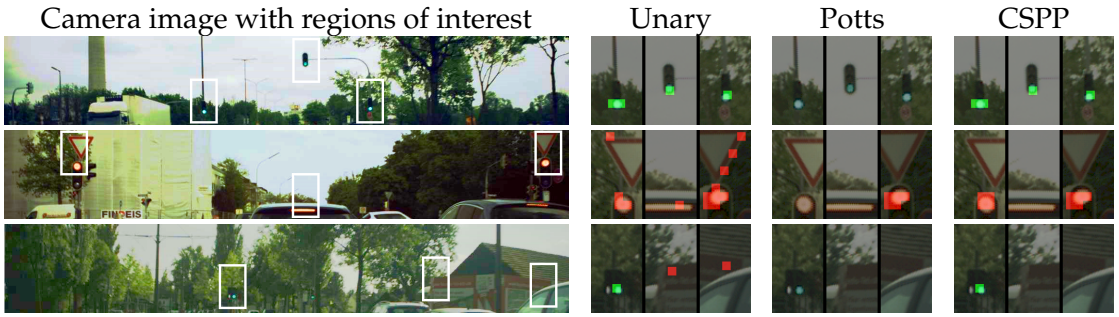


Figure 4.4: Images from the Traffic Lights Dataset with regions of interest. For each region we show the results for unary classification, using the Potts model and using our CSPP. Pixels outside of the regions are labeled as BACKGROUND.

Method	**Overall**	**Red light**		**Yellow light**		**Green light**	
	Accuracy	Precision	Recall	Precision	Recall	Precision	Recall
Unary	99.78%	19.9%	74.1%	4.0%	17.6%	42.8%	85.8%
Potts	99.92%	63.6%	25.9%	0.0%	0.0%	78.7%	31.0%
CSPP	**99.92%**	**44.1%**	**69.6%**	**7.7%**	**2.9%**	**65.9%**	**74.8%**

Table 4.2: Quantitative evaluation on the Traffic Lights Dataset. The unary classification detects most traffic lights (high recall), but has many false detections (low precision). The Potts model solves the problem with the false positives, but misses a lot of the traffic lights (low recall). Our method is able to filter out the false positives and still detect almost all lights.

4.2.2 Traffic Lights Dataset

We evaluate our method on a dataset that is particularly challenging for most methods containing a smoothing term, because the objects to be detected are very small. The dataset consists of 234 images (128 for training and 106 for testing) of road intersections with traffic lights, taken with a camera mounted behind the windshield of a car. After choosing a region of interest in the upper image part, where the traffic lights may appear, we get images with resolution of 1176×184 pixels. The images are labeled with 4 classes: RED, YELLOW and GREEN light and BACKGROUND. The labels are assigned in a square only around the illuminated spot of the traffic light and not around the whole box. For performance reasons we assign one node of the graph to a cell of 8×8 pixels. We use the same features as before, except that the Walsh-Hadamard transform is performed at 4 scales.

Here, the data is heavily unbalanced - more than 99.9% of the cells in the images are of class BACKGROUND. In such cases, most classifiers typically ignore the underrepresented classes almost completely. Therefore, during the training of the classifier we increase the penalty of not detecting a traffic light. While this measure significantly increases the recall of the traffic light classes, a lot of false positives appear in the case of pure unary classification and the resulting precision is very low as can be seen in Table 4.2.

Employing a pairwise term in the form of the contrast sensitive Potts model reduces the number of false positives, but many of the traffic lights are also smoothed away in

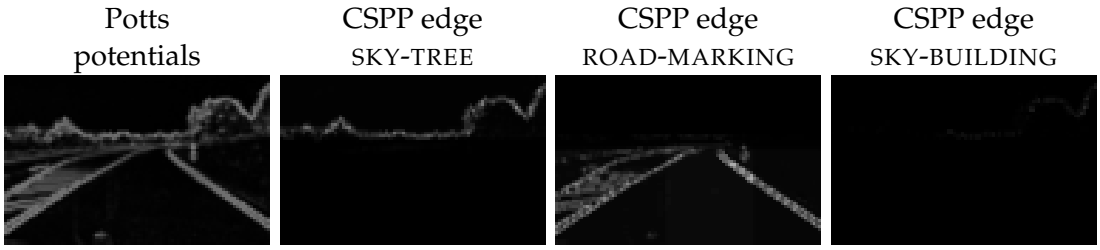


Figure 4.5: Comparison of the contarst sensitive Potts model potentials and the CSPP. While the Potts model activates on strong image edges, regardless of the class, the CSPP are able to classify the different object boundaries, like for example SKY-TREE or ROAD-MARKING.

the process resulting in very low recall. In contrast, by using our pairwise potentials, we can filter out almost all false positives, while the recall for the small classes is not affected significantly and therefore the majority of traffic lights are detected. This effect is illustrated in Figure 4.4.

4.3 Analysis

In order to better understand, where the shown improvement is coming from, we analyze the results in more detail and discuss some of the key concepts of the model.

First, we visualize the pairwise potentials for different class pairs in Figure 4.5. For the CSPP we show the edge potentials of type SKY-TREE, MARKING-ROAD and SKY-BUILDING for the whole image. The pairwise classifier is able to distinguish the edges of different types - the SKY-TREE and MARKING-ROAD edges are well aligned to the real edges of this type and an edge of type SKY-BUILDING is not detected since there are no buildings in this image. The edge potentials of the Potts model are identical for all label combinations and simply follow the edges in the camera image. Therefore, the Potts model has no understanding of the possible object boundaries and cannot exploit this local context information.

Similarly to the Potts model, our pairwise term also has a smoothing effect, but because of the pairwise classifier, our model is able to learn when to regularize and when not, depending on the local context. In order to illustrate this effect, we focus on one particular cell (marked in red) and visualize all the CRF potentials in Figure 4.6. The unary potentials are shown in the middle, while the pairwise potentials of the 4 edges to the neighboring cells are shown on the left, right, top and bottom respectively. In this case, the unary classifier would classify the cell as ROAD with higher probability than MARKING, which would be a mistake. Our pairwise classifier, however, detects a strong edge of type MARKING-ROAD on three of the edges with a value of up to 0.34 (marked in green) that is bigger than the regularization parameter for the class ROAD - 0.30 (marked in blue). The pairwise potentials do not only stop the neighbors from propagating the wrong label, but are also strong enough to correct the wrong unary potentials.

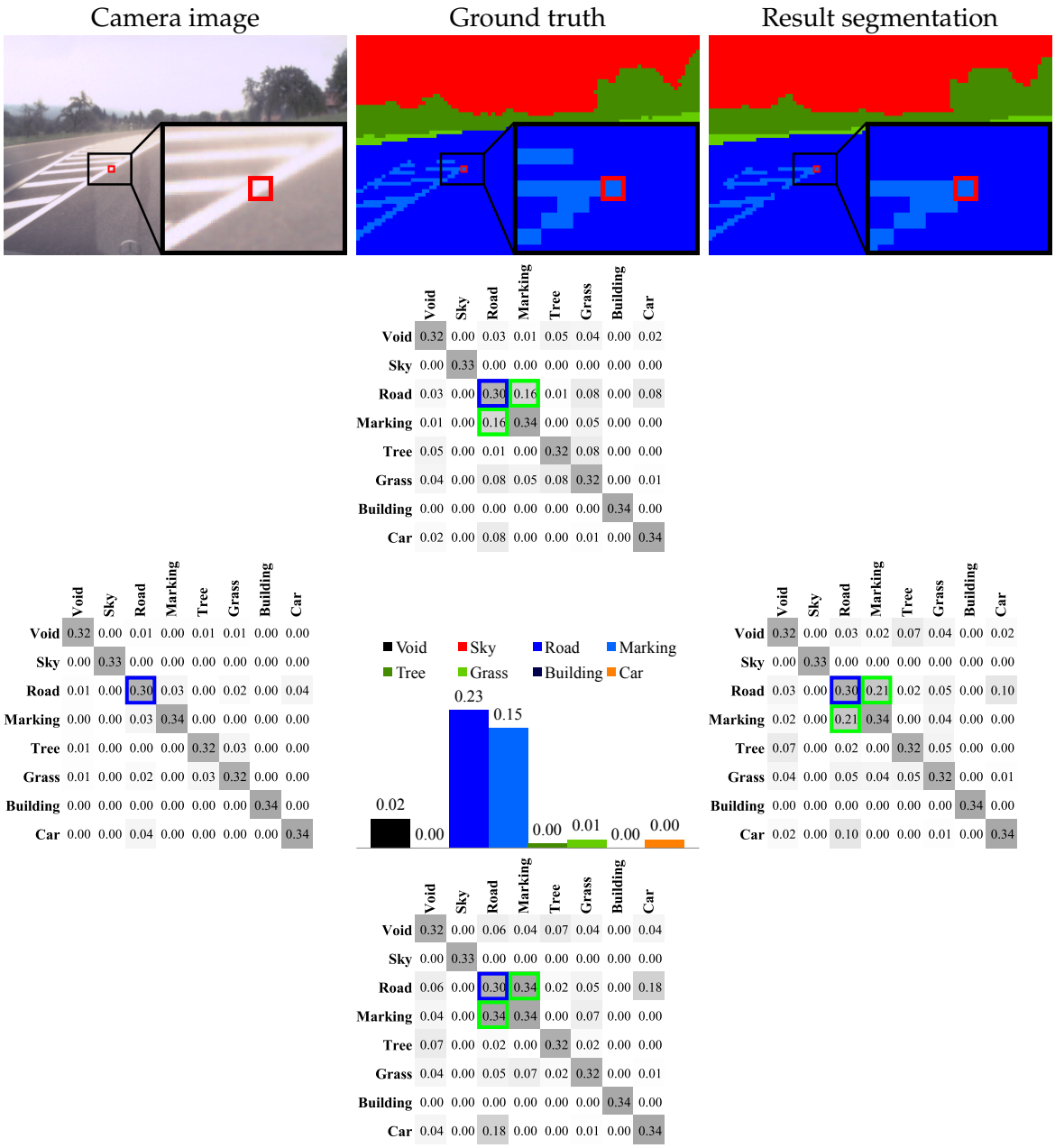


Figure 4.6: Detailed analysis of the CSPP. The figure focuses on the image cell marked in red and shows that the unary classifier has a higher probability for the class ROAD instead of the correct class MARKING (see the bar chart), but since the pairwise classifier provides a strong indication for an edge of type ROAD-MARKING (see the values marked in green) from the local context, the final segmentation is actually correct.

4.4 Discussion

In this chapter we presented a novel formulation for the pairwise term in a CRF model for semantic segmentation. Our class sensitive pairwise potentials are based on a nonlinear classifier that effectively classifies edges in the image into classes indicating all possible object boundaries. We show that this strategy is effective in capturing local context relations that allow to correct problems of the unary classifier. We also show that while these formulations delivers smooth segmentation, small objects are well preserved and not smoothed over by the classifier. We show that our method is able to deliver better results than similar methods that rely on the same information and even some higher-order CRF models.

5 Context for Classifier Models

The class sensitive pairwise potentials presented in the previous section are a good way to incorporate local context in a pairwise CRF framework, but still many ambiguities in the image need longer rang context to be resolved. In this section we present a novel classifier-based method for semantic segmentation that is designed to capture both local and global context as well as learning co-occurrence of objects. We build our method around the concept of the pixel neighborhood, which represents a set of pixels related in some way to the pixel of interest. In our two-stage classification framework, the output of a unary classifier based on image features is summarized over the pixel neighborhood by a novel voting histogram feature and given as input to a second classifier.

In this section we explore different ways to define neighborhoods and introduce a novel neighborhood type based on the geodesic distance transform. We also show that this concept can naturally be applied to both 2D and 3D images and that the combination of both modalities allows us to deal with problems, which are difficult to solve by each one separately. We compare our method to multiple state-of-the art approaches and show similar or increased performance, while keeping the runtime of our method low. Furthermore, our method models the context relations as soft constraints in a framework based on standard classifiers which is very easy to train and use.

Our main contributions are the two-stage segmentation framework based on local and global geodesic neighborhoods and the voting histogram features that provide a compact representation of context information. Furthermore, we show how to combine 2D and 3D data in the unary features and in the geodesic neighborhood by a novel use of the 2D Walsh-Hadamard transform and a combined distance measure. Additionally, we show how to use the geodesic neighborhood to efficiently perform smoothing of the final segmentation.

5.1 Method

The general pipeline of our two-stage classification method is illustrated in Figure 5.1. We first compute simple image features from the input camera image (and depth map if available) and we use a standard multi-class classifier to get the probability distribution

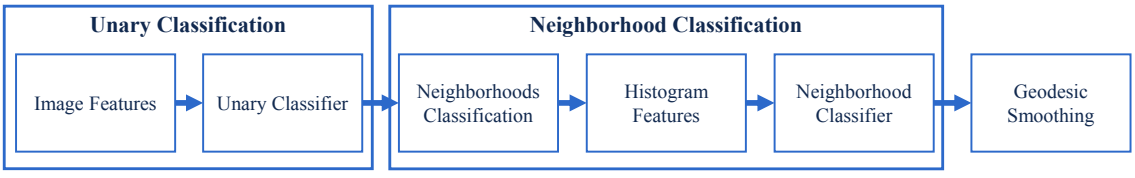


Figure 5.1: The pipeline of our two-stage classification method.

of every pixel over the possible classes. This process is fairly standard for many semantic segmentation methods. Next, we compute one or more neighborhoods for each pixel. Based on the neighborhoods and the predictions of the first classifier we compute a new type of features, called voting histograms, which serve as the input of the second classifier. The resulting segmentation is then geodesically smoothed to get the final segmentation.

5.1.1 Neighborhood Classification Framework

Here, we describe the two stages of our classification framework. The second stage can actually be repeated multiple times, forming a classifier chain, however, our experiments showed that this doesn't lead to significant improvements in most cases.

Unary classification stage. For the unary classification stage we make the independent variables assumption as described in Section 2.2. We model the conditional probability distribution of the pixel labels $\mathbf{y}$ given the input image $\mathbf{x}$ as:

$$P_1(\mathbf{y}|\mathbf{x}) = \prod_i P_1(y_i|\mathbf{x}), \tag{5.1}$$

where $P_1(y_i|\mathbf{x})$ is the probability of pixel i taking the label y_i. We estimate the probability distribution over the possible classes for each pixel by a multi-class classifier trained on multiple pixel samples and the corresponding ground truth labels. Similarly to the unary potentials in Section 4.1.1, the classifier does not operate on the image pixels $\mathbf{x}$ directly, but on a feature vector $\mathbf{f_u}(x_i)$ extracted from the image. Training such a classifier is a very common first step in many segmentation methods, especially for CRF, which typically use similar classifiers to define the unary potentials [74, 83, 128, 144, 152]. We use this unary probability distribution as an input to our second classification stage.

Neighborhood classification stage. The assumption that all pixel labels are independent of each other leads to simple factorization of the conditional probability distribution, but it rarely holds true in reality. Therefore, the segmentation resulting from the model above is usually very noisy and contains a lot of errors. To overcome this problem, we make each label dependent on a set of labels that are related to it in a specific way. For a pixel i we call the set of pixels related to it pixel neighborhood and denote it as N_i. The choice of the neighborhood is of critical importance for the performance of the method. In Section 5.1.2 we analyze different ways to define the neighborhood and introduce a novel method based on geodesic distance. While the second classifier still classifies each pixel individually, it uses the information from the whole neighborhood and can therefore learn to use context information. We show that we can design neighborhoods that provide both local and global context information.

Under the assumption that each label is independent of the other labels given its neighborhood, the conditional probability distribution for the neighborhood classification stage factorizes as:

$$P_2(\mathbf{y}|\mathbf{x}) = \prod_i P_2(y_i|\mathbf{x}, N_i). \tag{5.2}$$

Superpixels Mean-shift

Figure 5.2: Unsupervised segmentation with superpixels and mean-shift for different parameters.

We again model this probability distribution for pixel i as the output of a classifier operating on a feature vector $\mathbf{f_n}(N_i, P_1(y_{N_i}|\mathbf{x}))$ computed from the unary probability distribution of the pixels in the neighborhood N_i. For the computation of this feature vector $\mathbf{f_n}$ we propose a new compact feature, called voting histogram, described in detail in Section 5.1.4.

5.1.2 Pixel Neighborhoods

As described above, the neighborhood N_i of pixel i is defined as a set of pixels that are related in some specific way to the pixel of interest i. The choice of the neighborhood is important, because the neighborhood classifier determines the class of the pixel depending only on the pixels in its neighborhood. Here, we divide the possible neighborhoods in two variants: local and global neighborhoods. The local neighborhoods cover an area in the vicinity of the pixel i, while the global neighborhoods may include pixels throughout the whole image. Below we show several ways to define local and global neighborhoods and in Section 5.2.2 we compare their performance.

Euclidean Neighborhood The most intuitive way to define a local neighborhood is to take all the pixels in a predefined radius around the pixel i. This neighborhood can be computed very fast, but it has the disadvantage that it is independent of the input image and therefore cannot adapt to the image structure.

Superpixel Neighborhood Segmenting the image in superpixels is another natural way to define a local pixel neighborhood. We define the set N_i as all pixels belonging to the same superpixel as the pixel i. In this case, all the pixels in one superpixel have the same neighborhood. The fact that all superpixels have approximately the same size means that they may be too big for some small objects like signs or poles, while at the same time covering only very small part of bigger areas like the street or the sky. To compensate for this, we generate the superpixels neighborhood for several different values of the size parameter to get superpixels with different sizes (see Figure 5.2 for examples). For the computation of the superpixels, we use the state-of-the-art SLICO method [7], which is a version of SLIC [7], which automatically adapts the compactness parameter.

a) Local $\gamma = 0$ b) Local $\gamma = 50$ c) Local $\gamma = 1000$ d) Global $\gamma = 1000$

Figure 5.3: Visualization of the shapes of the geodesic neighborhoods for selected pixels (marked in black). The first 3 images show the shape of the local geodesic neighborhood for different values of γ, while the last image shows the global geodesic neighborhood, consisting of 8 separate ray neighborhoods.

Mean-shift Neighborhood Mean-shift segmentation [27] is another unsupervised segmentation approach, in which the segments are not constrained to have similar sizes. We define the mean-shift neighborhood in a way similar to the superpixel neighborhood: N_i is the set of all pixels belonging to the same segment as pixel i. Since it is very unlikely that all segments perfectly align to all object boundaries at the same time, we follow the approach presented in [74] to generate several segmentations with different parameters ranging from over-segmentation to under-segmentation (see Figure 5.2 for examples).

Geodesic Neighborhood The goal of the geodesic neighborhood is to define a local neighborhood that covers pixels in the vicinity of the pixel of interest that only belong to the same object. In this way, if a pixel is wrongly classified in the first stage, it can get support from other pixels nearby that were classified correctly. We define the geodesic neighborhood N_i as the set of the n pixels with the lowest geodesic distance to the pixel i. The geodesic distance is an extension of the Euclidean distance that also considers the image intensities. Therefore, two points in the image that have a high gradient between them will have a bigger distance than two points at the same Euclidean distance, but without strong edge between them. Formally, the geodesic distance in the continuous domain is defined as:

$$\mathcal{D}(i,j) = \inf_{\mathbf{\Gamma} \in \mathcal{P}_{i,j}} \int_0^l \sqrt{1 + \gamma^2 (\nabla I \cdot \mathbf{\Gamma}'(\mathbf{s}))^2} ds, \tag{5.3}$$

with $\mathcal{P}_{i,j}$ denoting all possible paths between two points i and j, $\mathbf{\Gamma}$ and $\mathbf{\Gamma}'$ a path of length l and its spatial derivative correspondingly and γ is a parameter. Here, we denote the image as I and the image gradient as ∇I, but it can be both a texture or a depth image.

Since we are working in the discrete image domain, the discrete version geodesic distance can be written as:

$$\delta(i,j) = \min_{\mathbf{G} \in P_{i,j}} \sum_{s=1}^{l-1} \sqrt{1 + \gamma^2 d(x_s, x_{s+1})^2}, \tag{5.4}$$

where $P_{i,j}$ is the set of all possible paths in the image between two pixels i and j, $\mathbf{G} = (s_1, s_2, ..., s_l)$ is a path of length l, containing the indices of the pixels along the path,

$d(x_s, x_{s+1})$ is a distance measure between two pixels along the path (for example the difference in the image intensities).

The parameter γ regulates the weight between the Euclidean and the geodesic term such that for $\gamma = 0$ the geodesic distance is equivalent to the Euclidean distance and for big values of γ the geodesic term dominates. In Figure 5.3 a), b) and c) we show how the parameter γ influences the shape of the geodesic neighborhood. We see that with the increase of the value of γ, the neighborhoods align better to the image structure and therefore provide more consistent evidence of the object that the pixel of interest belongs to. We present an efficient algorithm to compute the geodesic neighborhoods of all pixels in the image in Section 5.1.3.

Global Geodesic Neighborhood One drawback of all of the neighborhoods, described above, is that they only cover the area in the vicinity of the pixel of interest and therefore can only provide local context. In many cases, however, global context relations can be a very important queue to resolve ambiguities. Therefore, we introduce another type of neighborhood that is able to provide global context coming from other parts of the image, which again makes use of the geodesic distance to increase the robustness of the method.

The neighborhood is formed by first shooting 8 rays equally spaced at an angle of $45°$ from the pixel of interest to the borders of the image. Then, we take the local geodesic neighborhood at each point along the ray and create the union of all those neighborhoods separately for each ray. In this way, we get a set of 8 different neighborhoods, one for each ray, capturing the context information in a certain direction. While the shape of the neighborhood follows the ray, it can adapt to the image structure around it guided by the geodesic distance. The shape of the global neighborhood is shown in Figure 5.3 d).

5.1.3 Computing the Geodesic Neighborhood

While equation 5.4 is the formal definition of the geodesic distance, computing it in such a way is very inefficient. While there are several methods for fast and approximate computation of the geodesic distance transform [76, 141, 156] they are only suitable for the computation of the geodesic distance to large image regions. Since we need to compute the distance from each pixel to the n closest points in the image, these methods cannot be applied in our case.

We propose an algorithm for the computation of the geodesic distance based on the Dijkstra algorithm for finding the shortest path from a point to all other points in the image. We define a 4-connected graph over the image such that every pixel is a node in the graph and each pixel is connected to its 4 adjacent pixels with an edge. In practice, we connect each pixel with 4 pixels that are several pixels away from it in order to allow the neighborhood to quickly cover bigger parts of the image. We denote this step size as g. This is a concept similar to the dilated convolution described in Section 2.5.2. The weight of an edge connecting two pixels i and j is defined as the geodesic distance between them and can be derived from Equation 5.4:

$$w(i,j) = \sqrt{1 + \gamma^2 d(x_i, x_j)^2}, \tag{5.5}$$

where $d(x_i, x_j)$ is some distance measure depending on the type of image that is used. For

Algorithm 1: Computation of the geodesic neighborhood of a pixel.

Input: I - texture or depth image, $d(i, j)$ - distance measure between two pixels, (x_s, y_s) - coordinates of the pixel of interest, n - neighborhood size, g - step size, γ - geodesic term weight

Output: N_s - set of neighbors of the pixel s

$N_s \leftarrow \{\}$
foreach $p \in I$ **do**
  $dist[p] \leftarrow \infty$
  $dist_{geo}[p] \leftarrow 0$
end
$dist[s] \leftarrow 0$

for $i \leftarrow 1$ **to** n **do**
  $p \leftarrow$ pixel with minimum $dist$
  $N_s \leftarrow N_s \cup \{p\}$
  $P \leftarrow \{(x_p + g, y_p), (x_p, y_p + g), (x_p - g, y_p), (x_p, y_p - g)\}$
  for $t \in P$ **do**
    if $t \notin N_s$ **then**
      $d_e \leftarrow \sqrt{(x_t - x_s)^2 + (y_t - y_s)^2}$
      $d_g \leftarrow dist_{geo}[p] + d(t, p)$
      $d \leftarrow \sqrt{d_e^2 + \gamma^2 d_g^2}$
      if $d < dist[t]$ **then**
        $dist[t] \leftarrow d$
        $dist_{geo}[t] \leftarrow d_g$
      end
    end
  end
end

a color image I we define $d(x_i, x_j)$ as the distance between the pixels in the RGB color space:

$$d_{rgb}(x_i, x_j) = ||I(i) - I(j)||_2. \tag{5.6}$$

For a depth image, on the other hand, we use the metric distance between the 3D points:

$$d_{3D}(x_i, x_j) = ||P(i) - P(j)||_2. \tag{5.7}$$

If both a color image and a depth image are present, the distance measure can be defined as a combination of both with the help of a parameter α to balance their contribution, since they are defined in different domains:

$$d_{combined}(x_i, x_j) = \max(\alpha d_{rgb}(x_i, x_j), d_{3D}(x_i, x_j)). \tag{5.8}$$

The computation of the geodesic neighborhood itself is done by performing N iterations of the Dijkstra algorithm on the graph defined above and the neighborhood consists of the nodes chosen at each iteration. For a formal description see Algorithm 1.

5.1.4 Feature Vectors

We use different types of features for the two classification stages of our method. The features for the unary classifier of the first stage are extracted directly from the texture image and the depth image (if available), while the features for the neighborhood classifier in the second stage are computed from the pixel neighborhood and the class probability maps of the unary classifier.

Unary Classification Features

For the unary features two well-known and efficient methods are used: histograms of oriented gradients (HOG) [29] and the 2D Walsh-Hadamard transform [63]. The HOG features are computed only for texture images in a window of size five around every pixel and divided into 16 directional bins, resulting in a 16-dimensional feature vector, containing the gradients for each direction.

As in our CRF method from Chapter 4, here we also use the 2D Walsh-Hadamard transform [63]. In the case of texture images, the image is converted in the Lab color space and then the first 16 coefficients of the Walsh-Hadamard transform are computed in a window around every pixel separately for each color channel. We use windows of 6 different sizes: 2, 4, 8, 16, 32 and 64 pixels. In this way, we have 48 features for each scale, except for the window of size 2, where we have 4 instead of 16 coefficients. In the case of a color image, the feature vector has a size of 252, while in the case of grayscale images, we only have one channel and therefore the feature vector has a size of 84.

For depth images, we propose a novel way to use the 2D Walsh-Hadaramd transform. Computing the 3D coordinates of each pixel in the image results in structure that can be interpreted as a 3-channel float image (one channel for each 3D coordinate) and we compute the 2D Walsh-Hadamard transform in the same way as for the 3-channel color images described above. In this way, we get an approximation of the spatial frequencies of the image structure along the 3 coordinate axes in a very efficient way. This approximation works well for street scenarios, since most objects, like roads and buildings, tend to be aligned to

the same axes. Note, that in this case the 3D Walsh-Hadamard transform is not suitable, because we only have information derived from a 2D image and not a dense 3D volume like in a MRI scan for example.

Finally, we add the 2D coordinates of each pixel and if available the 3D coordinates as well, in order to encode location context. The final feature vector is constructed by stacking all of the available features.

Neighborhood Classification Features

For the neighborhood classification stage, we introduce a new voting histogram feature. The goal is to create a descriptor for the content of each pixel's neighborhood, based on the class probability maps from the unary classifier.

First, we compute a normalized histogram of each pixel i over unary classifier responses of the pixels in its neighborhood N_i:

$$h_i(c) = \frac{\sum_{j \in N_i} [c = v_j]}{|N_i|}, \tag{5.9}$$

where $c \in \mathcal{L}$ is a class label and $v_j = \underset{c}{\operatorname{argmax}}\, P_1(y_j = c|\mathbf{x})$ is the most probable label for the pixel j according to the response of the unary classifier. The so computed voting histogram can be directly used as a feature vector for the neighborhood classifier. Furthermore, we can compute multiple different neighborhoods $N_i^1, N_i^2, \ldots, N_i^k$ for each pixel and stack the histograms $h_i^1, h_i^2, \ldots, h_i^k$ for the different neighborhoods into one large feature vector. Additionally, we also add the responses of the unary classifier for the pixel of interest i. The final neighborhood feature vector for pixel i becomes:

$$f_N(i, N_i) = \begin{pmatrix} h_i^1 \\ \vdots \\ h_i^k \\ P_1(y_i|\mathbf{x}) \end{pmatrix}. \tag{5.10}$$

The dimensionality of this feature vector is $|\mathcal{L}| \cdot (k+1)$. Directly taking the unary responses of all pixels in the neighborhoods as features, similarly to what is done in the auto-context method [145], would result in a much bigger feature vector of size $|\mathcal{L}| \cdot \sum_{j=1}^{k} |N_i^j|$. Taking as an example a typical configuration from our experiments, consisting of 3 geodesic neighborhoods with sizes of 10, 50 and 200 for an 11 class problem, results in 2860 features when taking all pixels and only 44 when using the voting histogram. It is important to note that all pixels in the neighborhood are considered when building the histogram. Therefore, the histogram represents a summary of the whole neighborhood and while the information about individual pixels is not available anymore, every individual pixel still plays its role into building the histogram. Our experiments show that in fact, there is no significant difference between the results of the two approaches, while our is much faster. Using the summarized information, the classifier cannot model relations between individual pixels, but between regions of different sizes, which are more robust to individual pixel errors.

It is important to point out that the neighborhood features depend on the unary classifier output, but not on the image itself. While this may be seen as a limitation of the expressive

power of the neighborhood classifier, this formulation allows us to use compact and efficient features and leads to very short training and evaluation times as we show in Section 5.2.5. Furthermore, in our experiments we do not observe big performance loss, since the unary classifier seems to be able to fully exploit the image information. The authors of auto-context [145] report similar findings in their multi-stage classification framework.

5.1.5 Geodesic Smoothing

The output of the neighborhood classifier already provides good quantitative results, but still every pixel is classified individually. Therefore, in some cases there are individual wrong pixels which look like noise (see Figure 5.4 d)). This is a common problem with classifier-based methods, so many rely on a post-processing step, like for example by a pairwise CRF, to get smooth segments. However, such post-processing stages are computationally expensive and require a lot of parameter tuning or a separate training step.

We propose an alternative and very efficient approach that is again based on the geodesic distance. We compute the voting histogram again, but this time using the output of the neighborhood classifier instead of the unary classifier and we denote it as $\hat{h}_i$. We use the same definition as in Equation 5.9, but now $v_i = \text{argmax}_c P_2(y_j = c|\mathbf{x}, N_i))$. This time, however, instead of interpreting the voting histogram as a feature vector, we interpret it directly as the final probability distribution for pixel i, which can then be written as $P(y_i|\mathbf{x}, N_i) = \hat{h}_i$. This method has a positive impact on the quantitative and especially the qualitative results, with smoother regions that align well to the image structure (see Figure 5.4 e)). However, note that the size of the neighborhood used for smoothing should be small, because otherwise smaller objects may be smoothed over. Refer to Section 5.2.4 for detailed evaluation of the size of the smoothing neighborhood.

5.2 Evaluation

In this section we present an extensive evaluation of our method based on six well-known and challenging datasets. First, we analyze how different parts of our method deal with different problems in segmenting the image. We then demonstrate how 3D information can be easily integrated in our framework. We also present more implementation details and discuss the model parameters in order to give more insight of how our method works. Finally, we evaluate the runtime of the method and its individual stages.

In order to measure the performance of different methods and configurations we rely on three widely used evaluation measures:

- Global pixelwise accuracy - the percentage of correctly classified pixels.
- Average per class accuracy - the average of the pixelwise accuracies of each class calculated separately.
- Average Pascal accuracy - the average per class intersection over union measure as used in the Pascal VOC challenge (Pascal accuracy) [37].

Dataset	Scene type	Camera images	Depth data	Image resolution	Labeled images	Classes count
CamVid [18, 17]	driving	color	no	320×240	601	11
MSRC-21 [128]	general objects	color	no	320×240	591	21
Stanford Background [51]	outdoor scenes	color	no	320×240	715	8
eTRIMS [77]	building facades	color	no	768×512	60	8
Daimler Urban [119]	driving	grayscale	stereo	976×360	500	6
KITTI Segmentation [84]	driving	color	stereo	1240×376	60	9

Table 5.1: Overview of the datasets used for evaluation.

5.2.1 Datasets

Our evaluation is based on six widely used public datasets, three of which come from the automotive domain, whereas the other three contain more generic scenes. A systematic overview of the datasets is presented in Table 5.1. Additionally, we also evaluate our method on two application specific datasets. In Section 7 we evaluate our method on the task of detecting of parking spaces on the side of the road, while in Section 8, we deal with the problem of detecting traffic lights at both day and night time.

CamVid The CamVid dataset [17, 18] contains sequences of color images recorded from a moving car at daytime and at dusk. Most of the sequences are recorded at a low frame rate of roughly 1 frame per second. While computing structure from motion information from those sequences is principally possible [135], here we only rely on single color images. We use the same evaluation protocol as in [17, 85, 135] by downscaling the images to a resolution of 320×240 and taking the same 367 images for training and 233 for testing. Similar to [17] we consider only the 11 classes with most instances.

MSRC-21 The MSRC-21 dataset [128] is also a well-known and very challenging dataset, which contains nature and indoor images of which 276 are used for training and 256 for testing [128]. We consider the most common 21 classes (the classes MOUNTAIN and HORSE are commonly ignored) which range from trees, grass and sky to people, buildings and bicycles.

Stanford Background The Stanford Background dataset [51] focuses on outdoor scenes that have some particular objects in the foreground. The commonly used testing procedure for this dataset [51] is to perform a 5-fold cross validation on a split of 572 training and 143 testing images.

eTRIMS The eTRIMS image database [77] contains images of building facades divided in 8 semantic classes. Since the dataset contains only 60 images, we use the same testing procedure as in [45] by performing the evaluation on 10 different random splits of 40 images for training and 20 for testing.

Daimler Urban Segmentation The Daimler Urban Segmentation dataset [119] contains 500 grayscale stereo image pairs with corresponding dense disparity maps. The images

a) Image | b) Unary classification | c) Local geodesic neighborhood | d) Global geodesic neighborhood | e) With geodesic smoothing

Figure 5.4: Visualization of the results of the different stages of our method.

have a resolution of 1024×440 pixels and are labeled in the classes: GROUND, VEHICLE, PEDESTRIAN, BUILDING, SKY and BACKGROUND. For evaluation, we use the same split of 300 training images and 200 testing images as suggested by the authors.

KITTI The KITTI dataset [46, 47] is a comprehensive benchmark for various computer vision problems related to autonomous driving like stereo and optical flow computation, object detection, visual odometry and others. We use the annotated set of 60 images as in [84]. The images have a resolution of 1240×376 and are equally split into a training and a testing set. Furthermore, the KITTI dataset provides stereo image pairs, which allows us to compute depth information.

5.2.2 Analyzing the Method

In this section we analyze how the different stages of our method deal with different segmentation problems and how they contribute to the final performance. In Figure 5.4 we visualize the results at the different method stages on one example image from the Stanford Background dataset and use it in order to explain different effects through the section. In parallel, we also present a full evaluation of the same method steps on the whole CamVid, MSRC-21 and Stanford Background datasets in order to measure the same effects quantitatively (see Table 5.2). More detailed results and comparison to other methods on multiple datasets are presented in Section 5.3.

Unary Classification

The unary classification step of our method uses relatively simple, but very fast features and therefore the segmentation results are quite noisy and with big error regions. Looking at Figure 5.4 b) the windscreen of the car is wrongly classified as WATER, the bumper of the car as ROAD and there are multiple wrong patches on the ground as well. While employing more powerful features would lead to better unary classification performance at the cost of increased runtime, in this paper we focus on developing a universal higher-level method that is able to correct errors by integrating context from multiple image regions.

Local Neighborhood

By using local neighborhoods, our method is able to correct many errors from the unary classifier by using context information from the vicinity of each pixel. Looking at Figure 5.4 c), we see that many of the smaller errors are filtered and the shape of the car is much

Method	CamVid			MSRC			Stanford Background		
	Global	Average	Pascal	Global	Average	Pascal	Global	Average	Pascal
Unary	70.6	56.4	35.7	61.4	50.4	33.4	66.6	64.2	46.0
Superpixels	74.5	60.6	39.6	67.2	59.2	40.7	69.5	66.8	48.8
Mean-shift	75.6	61.7	40.5	71.7	63.7	46.0	71.1	68.2	50.4
Euclidean	76.0	62.3	41.2	73.0	67.5	49.1	70.4	68.0	49.5
Geodesic (local)	76.5	63.1	41.9	71.9	66.2	47.8	71.5	68.4	50.5
Geodesic (global)	**77.9**	**64.0**	**43.6**	**76.3**	**70.7**	**54.5**	**73.2**	**70.1**	**52.4**

Table 5.2: Quantitative evaluation of the different local and global neighborhoods on CamVid, MSRC-21 and Stanford Background.

better defined. This is also confirmed by the quantitative evaluation in Table 5.2 with all of the local neighborhoods delivering significantly better results than the unary classifier. Comparing the different local neighborhoods between each other we see that the local geodesic neighborhood achieves the best results. While the difference in some cases may seem small, the geodesic neighborhood has the important property of aligning the segmentation borders to the object boundaries that have a strong gradient, which is usually not directly visible in the quantitative analysis.

In order to better understand why this is the case, we visualize the voting histogram features computed over different neighborhood variants in Figure 5.5. Here, we show the normalized values of the histogram bins for the classes TREE, ROAD and OBJECT. Note that even though the images can be interpreted as a probability distribution for each class, these are actually the features that serve as the input to the neighborhood classifier as described in detail in Section 5.1.4. We see that there are some significant differences in the histogram features for the different neighborhoods and therefore also in the resulting segmented images shown in the last row of Figure 5.5. The Euclidean neighborhood simply covers all pixels in a certain radius and therefore pixels close to object edges get responses from other pixels on both side of the edge. Therefore, the features in those regions are blurred, resulting in segmentation borders that do not align well to the object edges. The superpixels can better align to the image structure, but they are either too big to fit to the small details of the car, or too small to capture enough local context. The mean-shift segmentation does not have the size constraints of the superpixels and can therefore adopt better to the objects. However, as with any unsupervised segmentation method, it is extremely difficult to create a disjoint image segmentation that captures all object edges. This problem is somewhat mitigated by creating multiple segmentations with multiple parameters as described in Section 5.1.2, but we can see that the results are still not as good as with the geodesic neighborhood. The main advantage of the geodesic neighborhood is that it generates a neighborhood with a different shape for each individual pixel, which allows it to adapt well to small structure details, while in the same time being big enough to capture enough local context.

Global Neighborhood

While the local neighborhoods are able to correct smaller errors, their limited range makes it more difficult to handle bigger errors like the bumper of the car or the "water" patches on the ground in Figure 5.4 c). Adding the global neighborhood to the neighborhood

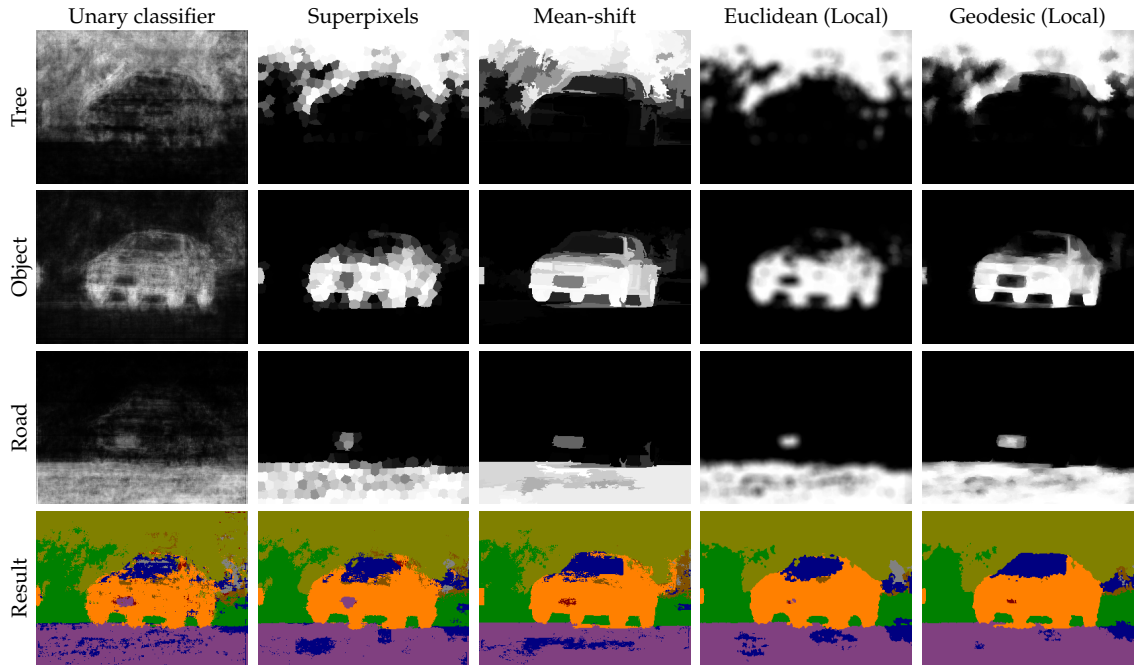
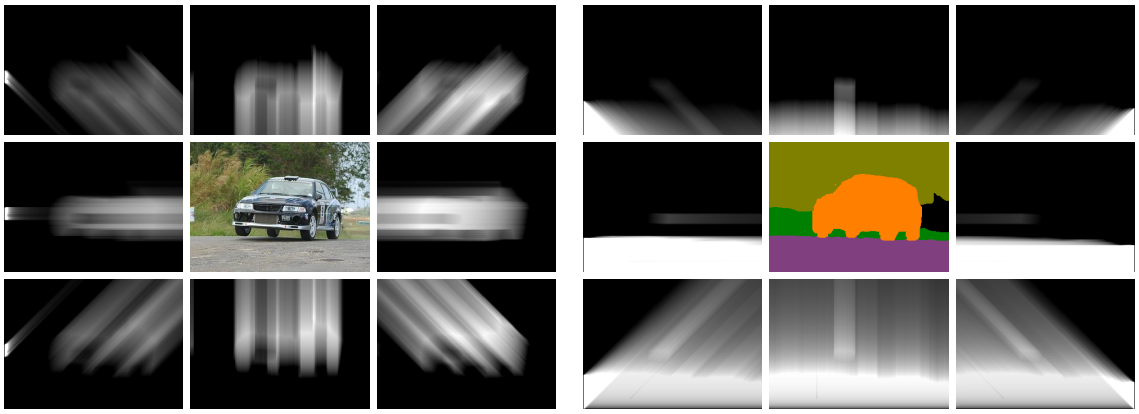


Figure 5.5: Comparison between the voting histogram features computed on different local neighborhoods. The images show the normalized values of the histogram bins for the corresponding classes and the resulting segmentation.

classifier feature vector allows it to learn higher-level context information that resolves most of these problems.

To explain this, in Figure 5.6 we visualize the features of the eight rays of the global geodesic neighborhood for the classes ROAD and OBJECT. The images can be interpreted in the following way: a high value for a pixel in the image for the ray pointing to the left for the class OBJECT means a high probability that there is a region of the class OBJECT on the left of the chosen pixel. If we take as example one of the pixels on the ground that were wrongly classified as WATER, it will get very high responses for the class ground from the rays pointed downwards and sideways, while from the rays pointed up it will get responses for the classes OBJECT, TREE and SKY. Based on this information of the objects in the image surrounding the pixel, the classifier is able to learn the respective context relations and correct most of the errors (see Figure 5.4 d)).

From the quantitative evaluation in Table 5.2 we can see that the usage of the global geodesic neighborhood gives a significant advantage over the local variants. It is interesting to note that the increase in performance is much bigger for the MSRC-21 dataset. Analyzing the images in more detail shows that in the Stanford Background and in the CamVid dataset most of the images contain instances of all classes together. In this case, the global neighborhood learns spatial context relations between objects, e.g. cars are on top of the road and the sky is above the buildings and the cars. In MSRC-21, however, the images show a variety of different situations that usually contain only 2 or 3 of the 21 classes. In this case, the global neighborhood is able to also learn the co-occurrence between the classes, e.g. cows and trees appear in images with grass, while birds and



Rays for class object | Rays for class road

Figure 5.6: Visualization of the 8 rays of the global geodesic neighborhood for the classes object and road.

airplanes appear in images with sky. Further examples of this can be seen in Figure 5.13.

Geodesic Smoothing

While the segmentation based on the global neighborhood is already good, we can see that there are many small pixel errors everywhere in the image. This is due to the fact that the neighborhood classifier takes the decisions about the classes of the pixels independently of each other. By using our geodesic smoothing method (see Section 5.1.5) we get smoother segmentation (see Figure 5.4 e)) and an improvement in the overall classification performance.

5.2.3 Integrating 3D Data

Our framework is not limited to handling 2D images, but it can also naturally handle 3D data in all stages: feature computation, neighborhood computation and geodesic smoothing. The Daimler Urban dataset and the KITTI segmentation dataset contain stereo image pairs from which depth data can be computed. While the Daimler Urban dataset already comes with precomputed dense disparity maps of high quality, for the KITTI dataset, we compute the disparity maps by using the efficient method of [155] which is one of the top ranked methods on the KITTI stereo benchmark. For performance reasons, we do not classify each pixel individually, but we use cells of 4×4 pixels. Note, however, that this is not equivalent to downscaling the images 4 times, because we still compute the features at full resolution and compare the result to the ground truth data on the pixel level.

For both datasets, we use the same error measures as for the 2D datasets. The authors of the Daimler Urban dataset use a somewhat different measure, which is based on the average intersection over union, but excludes the class BACKGROUND. Furthermore, they also report the average only on the two dynamic classes VEHICLE and PERSON. We report the same measures as in [120] in addition to the others in order to allow for comparison.

Method	Global	Average	Pascal (all)	Pascal (dyn)	Ground	Vehicle	Person	Building	Sky
Unary Texture	60.6	64.8	50.6	33.6	68.8	41.1	26.1	68.0	48.9
Unary Depth	61.1	64.2	44.2	37.7	71.0	45.2	30.2	25.3	49.1
Unary Combined	66.2	70.6	57.7	46.6	72.6	54.9	38.3	68.9	53.6
Geodesic Texture (global)	70.2	**76.4**	65.2	62.0	73.8	**71.0**	53.0	72.2	55.8
Geodesic Depth (global)	70.2	75.5	64.3	60.1	73.2	69.0	51.2	71.3	56.9
Geodesic Combined (global)	**71.5**	76.2	66.1	61.9	75.5	70.9	52.8	**74.2**	56.9

Table 5.3: Quantitative evaluation of the neighborhood classifier using different 2D and 3D data on the Daimler Urban segmentation dataset.

Method	Global	Average	Pascal	Road	Building	Sky	Tree	Sidewalk	Car	Grass	Column	Fence
Unary Texture	66.1	60.4	38.8	74.2	66.7	92.1	64.7	59.8	57.0	69.9	7.8	**51.8**
Unary Depth	51.7	53.5	31.2	79.4	54.3	85.4	30.6	57.5	65.7	49.8	**14.2**	44.4
Unary Combined	72.4	65.5	45.3	79.3	75.4	92.3	69.6	66.1	71.5	77.9	10.5	47.3
Geodesic Texture (global)	75.2	**68.4**	48.7	**84.8**	**85.0**	**93.6**	66.8	64.8	**79.6**	**84.9**	13.2	42.7
Geodesic Depth (global)	74.0	66.0	46.5	84.1	81.1	88.0	67.8	63.2	74.5	83.3	1.4	50.5
Geodesic Combined (global)	**76.1**	68.3	**49.3**	84.2	82.7	93.4	**70.9**	**67.5**	78.8	82.5	11.5	43.4

Table 5.4: Quantitative evaluation of the neighborhood classifier using different 2D and 3D data on the KITTI segmentation dataset.

Unary Classification

For the unary classification step we evaluate three different unary features sets (as described in Section 5.1.4): only texture features, only depth features or both together. From the quantitative evaluation in Table 5.3 and Table 5.4 we can see that for both datasets using each modality separately already gives meaningful results. This also shows that our novel formulation of the 2D Walsh-Hadamard transform in 3D provides discriminative features. Combining both modalities in the classifier results in a noticeable improvement. Therefore, all of the following evaluations of the neighborhood classifier are based on the output of the combined texture and depth unary classifier.

Neighborhood Classification

Having access to the depth data, gives us the possibility to define the geodesic distance not only on the image gradients but also in the 3D space (see Section 5.1.3). Therefore, we can compare the performance of the geodesic neighborhood when defined for different image modalities. From the quantitative evaluation on both of the datasets we see that the two variants perform similarly well, which shows that our method can be naturally adapted to different image modalities (in fact we are even using the same unary features).

However, looking at the shapes of the neighborhoods in more detail (see Figure 5.7) we can see that they can be quite different in some regions. The geodesic neighborhood defined in the image space can overflow from the dark area on the cars to the dark areas on

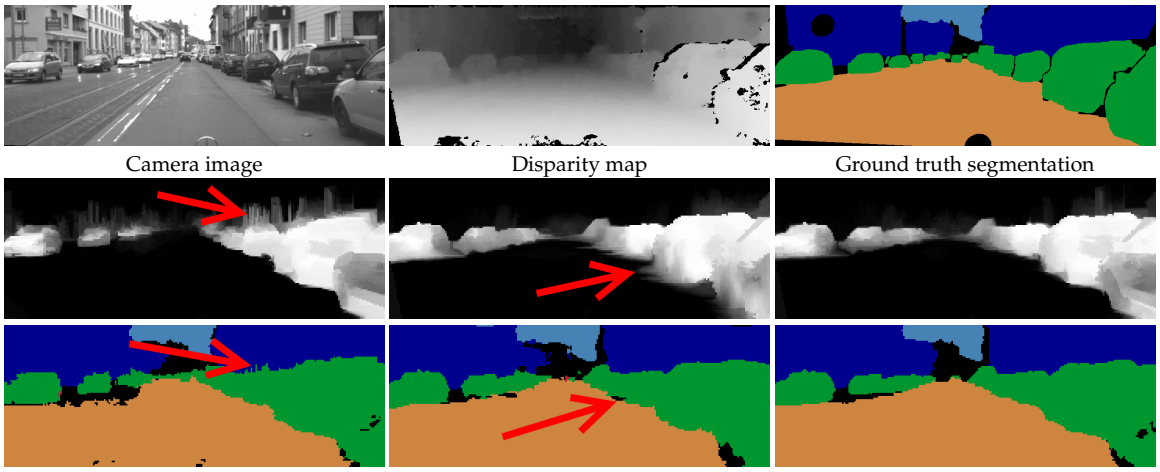


Figure 5.7: Visualization of the geodesic neighborhood defined in the 2D texture space, the 3D metric space and combined. The red arrows point to some typical problems of the geodesic neighborhood when defined in the texture or depth information alone that can be resolved when both are combined.

the building because of the lack of contrast. While this is not the case for the 3D geodesic neighborhood, because there is a big difference in the depth between the cars and the building, the geodesic neighborhoods can flow out of one object into another one if they are touching. This can be observed around the tires of the cars touching the street. Combining both distances in the neighborhood definition as described in Section 5.1.3, with $\alpha = 10$, allows us to use the advantages of both modalities, which leads not only to big qualitative, but also to significant quantitative improvements. This is also the case for the global geodesic neighborhood, where the rays can better adapt to the image structure if they use the edge information in both image modalities.

5.2.4 Implementation Details and Parameters Evaluation

In this section we give more details about setting the model parameters. Our method generalizes well enough so that we can use the same parameters for the computation of the features and for the classifier training over all datasets.

Classifier Parameters

For both the unary and the neighborhood classifiers, we employ the JointBoost method [142] that can efficiently handle multi-class classification problems. We use some standard techniques [83, 129] to speed up the training by not taking the entire image, but sub sampling it on a grid of size 5 instead. Furthermore, at each iteration step we test a random 30% of the features by comparing them to 100 thresholds that are sampled uniformly over the values of each feature. In fact, the only difference between the unary and the neighborhood classifiers is in the number of iterations, which is only for performance reasons. While for the unary classifier we need 5000 iterations in order to get the maximum out

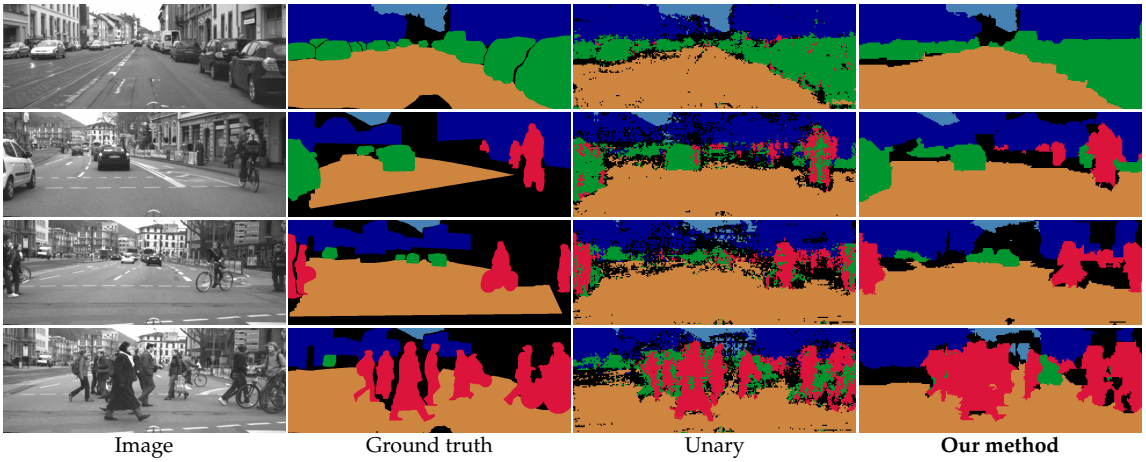


Figure 5.8: Result images on the Daimler Urban Segmentation dataset.

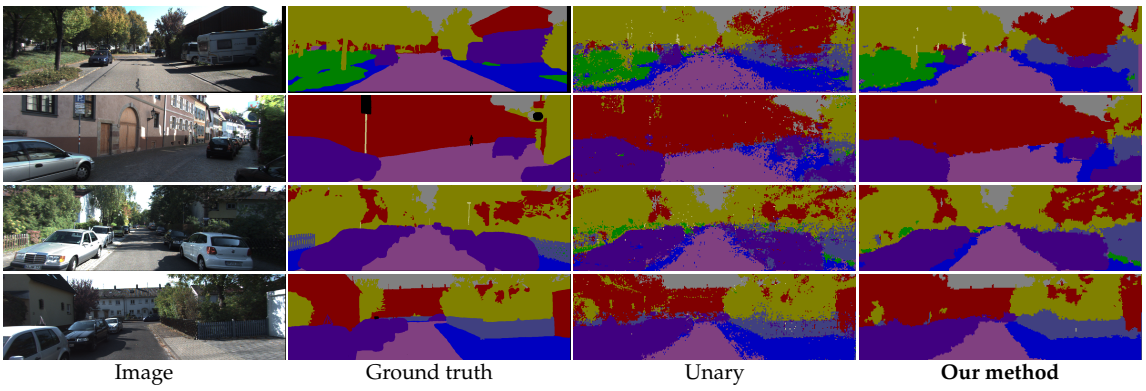


Figure 5.9: Result images on the KIITI dataset.

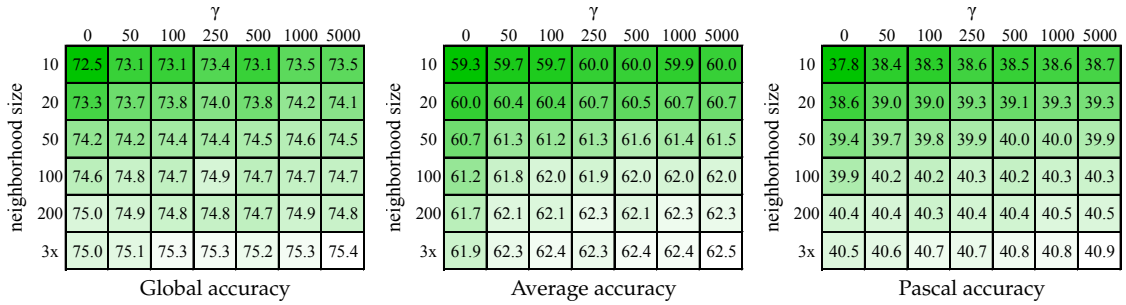


Figure 5.10: Influence of the two most important parameters controlling the size and the shape of the geodesic neighborhood on the performance of our method. The neighborhood size 3x indicates the usage of 3 neighborhoods of size 10, 50 and 200 pixels.

of the relatively big feature space (see Section 5.1.4), the neighborhood classifier, which operates on our compact voting histogram features, saturates after 1000 iterations. The parameters of the classifiers have influence mainly on the training time of the method.

Neighborhood Computation

Here we evaluate the influence of the geodesic neighborhoods computation parameters on the performance of our method. Since the results are comparable for all datasets, here we show detailed numbers only from the CamVid dataset.

Neighborhood Size and Geodesic Weight The size of the neighborhood and the weighting parameter γ are the most important parameters for the segmentation results, because they influence the size of the region taken into account and its shape. In Figure 5.10 we show an evaluation of our method using the local geodesic neighborhood for different combinations of the values. We see that making the neighborhood bigger also gives better results, but the effect saturates after a size of 100. Smaller neighborhoods work better for smaller objects, because they do not go over the object boundary, but for bigger objects, they do not cover enough pixels to capture enough local context. Therefore, a combination of several neighborhoods is beneficial. In our case, we use three neighborhoods of sizes 10, 50 and 200 (denoted as 3x). Note that the three neighborhoods can be computed efficiently, by just computing the neighborhood of size 200 and taking the first 10 and 50 pixels for the others. As discussed above, the local geodesic neighborhood achieves better results than the local Euclidean neighborhood, which can also be seen in Figure 5.10. This effect saturates at high values of γ, because the contribution of the Euclidean distance becomes negligible. For our experiments, we use $\gamma = 5000$ in order to maximize the effect of the geodesic term.

Neighboring Pixels Grid Size Another parameter that influences the shape of the neighborhood is the grid size at which neighboring pixels are sampled by the Dijkstra algorithm (see Section 5.1.3 for details). Higher values of the parameter allow the neighborhoods to cover bigger parts of the image and lead to some improvement in performance especially

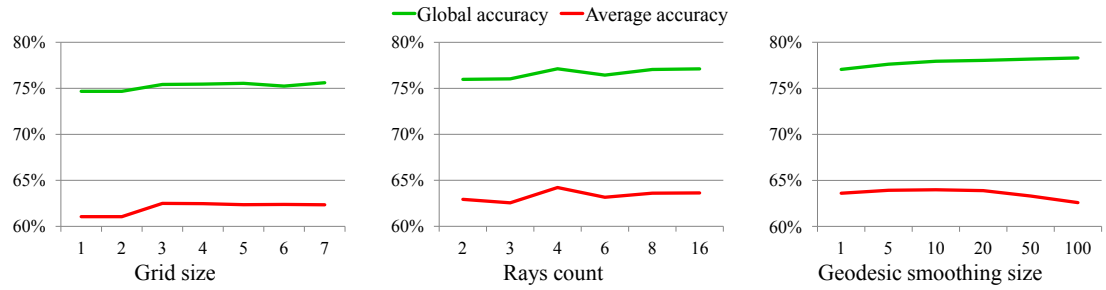


Figure 5.11: Evaluation of the parameters for the sampling grid size, the number of rays in the global geodesic neighborhood and the size of the neighborhood used for geodesic smoothing.

for the average accuracy (see Figure 5.11). However, this effect quickly saturates for after a grid size of 3.

Rays Count For the global geodesic neighborhood, the number of rays determines from which directions information from the image is gathered. As can be seen in Figure 5.11, more directions give the neighborhood classifier more information about the scene and lead to higher accuracy. Adding more than 8 rays, however, does not lead to any further improvement.

Geodesic Smoothing As discussed in Section 5.1.5, the size of the neighborhood used for geodesic smoothing should be small in order to not smooth over smaller objects. This is confirmed by the results in Figure 5.11, where the global accuracy increases with bigger neighborhood sizes, but for larger values, the average accuracy starts to decrease. This is due to the fact, that large neighborhoods flow out of smaller objects like poles or signs and consequently they tend to be lost in the image resulting in a drop of the accuracy for the smaller classes and therefore also the average per class accuracy.

5.2.5 Runtime Evaluation

A lot of computer vision systems like driver assistant systems, autonomous vehicles or mobile robots need to operate and react in the real world. This means that the runtime of such methods is an important factor to consider. Furthermore, using specialized hardware like a GPU or a FPGA may not always be possible. In this section we evaluate the runtime of our method and show how we can adjust the trade-off between speed and accuracy.

In Table 5.5 we show the runtime of our method for two of the datasets used for evaluation - one using camera images only (CamVid) and one with both 2D and 3D data (Daimler Urban). For the Daimler Urban dataset we evaluate two variants of our method using cells of size 4×4 pixels, like in the quantitative evaluation presented above and 8×8 pixels. All tests are performed on a desktop machine equipped with two Intel Xeon X5690 processors with 6 cores each running at 3.46 GHz. All parts of our method are parallelized using OpenMP, but no other specific optimizations, like usage of SSE instructions, have been performed. All of the code is running on the CPU only and no GPU is used.

Method stage	**CamVid** 1×1 pixel	**Daimler Urban** 4×4 pixels	**Daimler Urban** 8×8 pixels
Unary features	544 ms	159 ms	90 ms
Unary classifier	1106 ms	53 ms	11 ms
Geodesic Neighborhood	2016 ms	402 ms	52 ms
Voting Histogram Features	137 ms	149 ms	15 ms
Neighborhood classifier	290 ms	187 ms	35 ms
Geodesic Smoothing	13 ms	5 ms	1 ms
Total	**4106 ms**	**955 ms**	**204 ms**

Table 5.5: Runtime of our method on two of the datasets used in the evaluation.

We can see that the performance of our method is highly dependent on the pixels or cells our method has to classify and compute the features for. While for the CamVid dataset, we have 76,800 pixels per image, when we use cells of size 4×4 pixels for the Daimler Urban dataset we need to process only 21,960 cells and this explains the big difference in the runtime between the two datasets. Around 40% of the time is spent computing the geodesic neighborhoods for every pixel. This can be further optimized by taking into account the nodes found for the neighboring pixels, because adjacent pixels tend to have very similar neighborhoods.

One useful property of our method is that one can easily control the trade-off between segmentation accuracy and runtime by adjusting the size of the cells. We show this on the Daimler Urban dataset by increasing the cell size to 8×8 pixels and by this effectively making the method more than 4 times faster. As expected, the accuracy decreases, but not by very much - the faster method is still able to achieve 65.2% average Pascal accuracy for all classes instead of 66.1% and 59.5% instead of 61.9% for the dynamic classes.

Since in our method one and the same operation is applied to multiple pixels simultaneously, the code can be easily sped up further by vectorizing it on a suitable hardware (like a GPU or DSP) or by using SSE instructions. Another option for optimization would be to adapt the computation of the geodesic neighborhood for a given pixel to use the results of neighboring pixels that have already been computed.

In Chapter 7 we show how our method is used in a real-time application as a part of a system for the detection of parking spaces from a driving car.

5.3 Comparison to Other Methods

We perform a detailed comparison of our method using the global geodesic neighborhood to six closely related state-of-the-art methods. Because most semantic segmentation methods are strongly dependent on the features that are computed from the images, we try to compare methods in a setting where the same image features are used.

First, we compare to the CRF based Robust P^n model [74] and the classifier-based auto-context method of [145]. Here, we use the unary features as described in Section 5.1.4. For the Robust P^n model [74] we use the implementation provided by the authors, while for auto-context [145] we use our own implementation.

For the state-of-the-art method of [83] we adopt the features and the unary classifier from the implementation that is published online by the authors. The overall accuracies

Method	CamVid			MSRC			Stanford Background		
	Global	Average	Pascal	Global	Average	Pascal	Global	Average	Pascal
Unary	70.6	56.4	35.7	61.4	50.4	33.4	66.6	64.2	46.0
Geodesic (local)	76.5	63.1	41.9	71.9	66.2	47.8	71.5	68.4	50.5
Geodesic (global)	**77.9**	**64.0**	**43.6**	**76.3**	**70.7**	**54.5**	**73.2**	**70.1**	**52.4**
Auto-context [145]	74.5	61.7	40.5	72.5	67.3	49.4	72.0	69.4	51.5
Robust P^n [74]	**77.9**	56.0	40.5	73.4	65.0	47.7	71.9	67.9	50.8

Table 5.6: Quantitative comparison to auto-context [145] and the Robust P^n [74] method on CamVid, MSRC-21 and Stanford Background.

with those features are much higher because the authors [83] use more sophisticated, but also much slower image features.

We use the Daimler Urban dataset to compare to the multi-cue segmentation method of [119] and the Stixmantics method [120], which combine texture and depth data to create a fast method for real-time semantic segmentation. Since the authors do not provide access to their code, we cannot use the same features during evaluation.

The Iterative Context Forests (ICF) method [44, 45] is also related to our method since it learns context relations. For comparison, we evaluate our model using similar features on the eTRIMS dataset. We also compare to a version of ICF that was extended by the authors of [120] to handle depth data and was also evaluated on the Daimler Urban dataset.

For the last comparison, we take the output probability maps from the deep learning method SegNet [9] as input to our neighborhood classification stage. This means that we effectively apply our method on top of the SegNet results.

5.3.1 Robust P^n Model

The Robust P^n model is a powerful higher level CRF model that relies on multiple segmentations (based on mean-shift) of the input image in order to better align the labeled segments to the image gradients. While our idea is similar, we take a completely different approach for modeling the relationships between pixels and segments. While the authors of [74] rely on a CRF to model these relations, in our model they are encapsulated in the voting histogram features and learned by the classifier. For our experiments, we use the inference implementation provided by the authors, but using our image features and classifier for the unary potentials of the CRF.

The Robust P^n model performs very well with regards to the global accuracy on the CamVid dataset, but is outperformed by our method on the other datasets and for the other evaluation measures as can be seen in Table 5.6. If we look at the result images in Figure 5.12, 5.13 and 5.14 we can see that this method produces well-defined label segments that align good to image structures due to the mean-shift segmentation, but they may be wrong or smooth over smaller objects and therefore get worse results on the average per class accuracy and the Pascal accuracy.

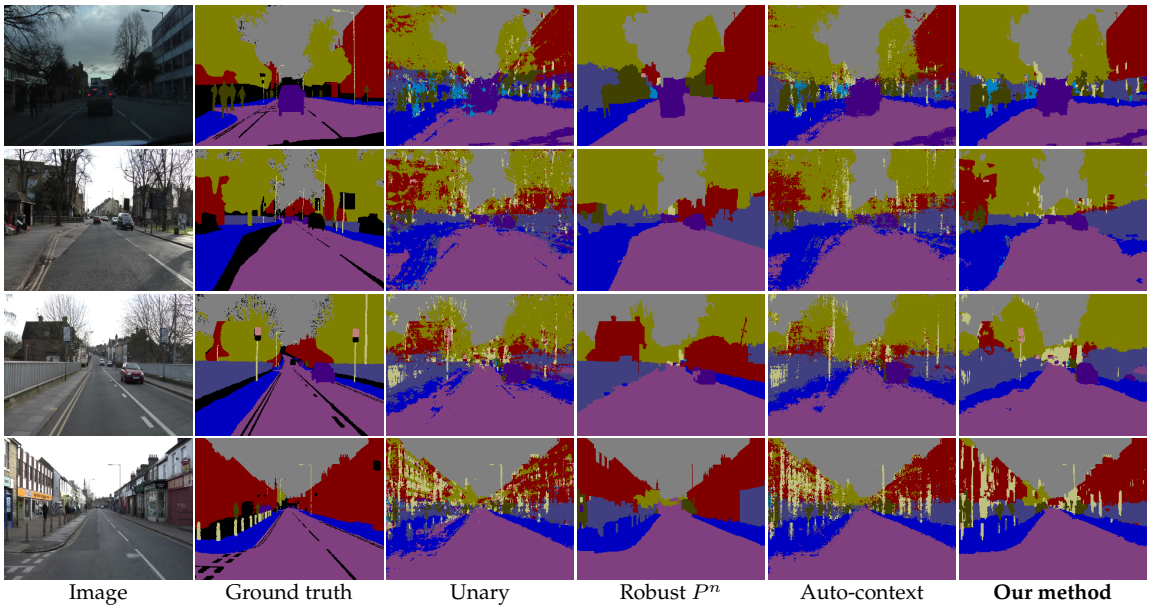


Figure 5.12: Result images on the CamVid dataset and comparison to auto-context [145] and the Robust P^n [74] model.

5.3.2 Auto-Context

The auto-context method is related to our model, because it uses a chain of classifiers, but with a different idea behind the higher-level features. While we use the adaptive geodesic neighborhoods to summarize the output of the unary classifier around the pixels of interest and along 8 rays that provide more global context, auto-context uses fixed points along 8 similar rays to sample the image features and the responses of the previous classifier. Therefore, the feature vectors for the classifiers in auto-context are much bigger than those in our method and the sampling is not adapted to the image structure, but is always done at the same offsets. However, we can easily implement auto-context as a use-case of our method by using the same sampling structure as in [145] and regarding each sampling point as a separate neighborhood containing only one pixel. We perform two iterations of auto-context because we also train two classifiers for our method and the analysis in [145] shows that the performance of auto-context quickly saturates after the second iteration.

From the qualitative results in Figure 5.12, 5.13 and 5.14, we can see that the sampling structure of auto-context is able to capture more context than the local version of our method on the MSRC-21 dataset. This is due to the fact that co-occurrence plays a very important role in the MSRC-21 dataset, since it contains different situations in which the same 2-3 types of objects appear together. Since, auto-context samples pixels at long distance around the pixel of interest, it can incorporate those co-occurrence relations. However, this is also true for our global method, which in addition also provides better results around the object boundaries and therefore achieves superior results overall (see Table 5.6). On the other datasets, where the global context relations are not that strong even the local version of the geodesic neighborhood performs better.

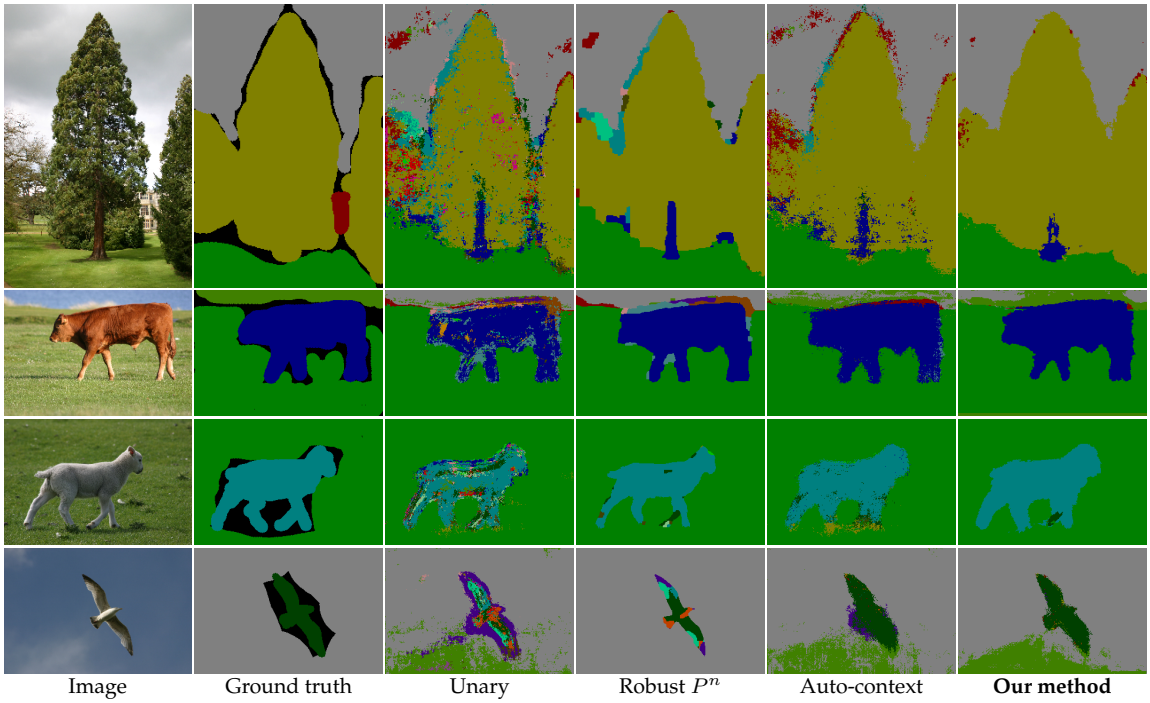


Figure 5.13: Result images on the MSRC-21 dataset and comparison to auto-context [145] and the Robust P^n [74] model.

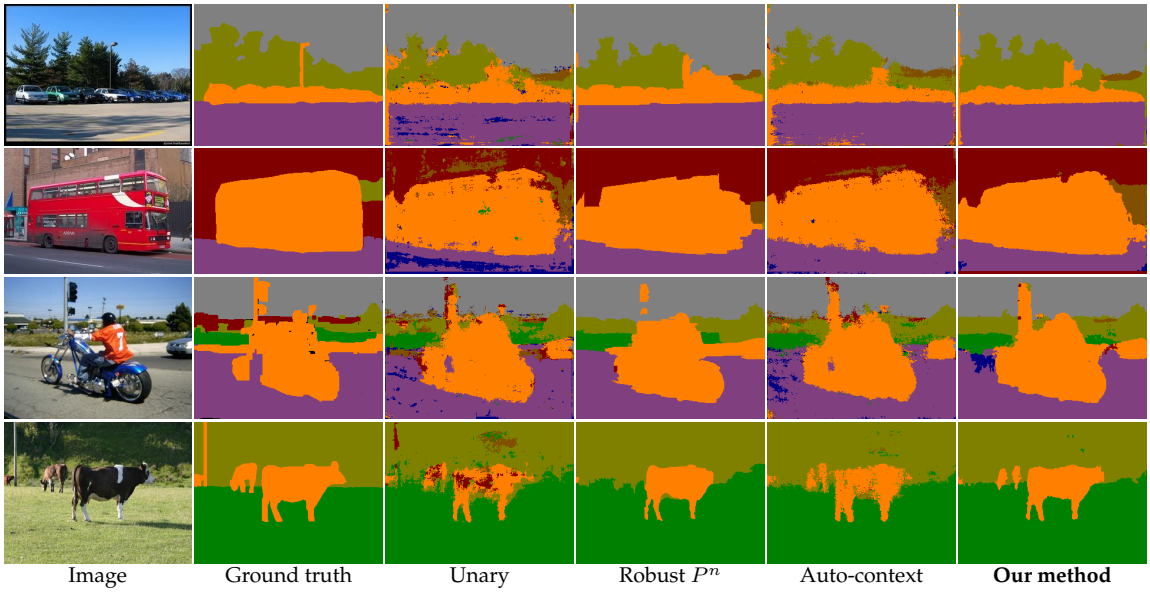


Figure 5.14: Result images on the Stanford Background dataset and comparison to auto-context [145] and the Robust P^n [74] model.

Method	CamVid			MSRC		
	Global	Average	Pascal	Global	Average	Pascal
Unary from Section 5.1.4	70.6	56.4	35.7	61.4	50.4	33.4
Unary from [83]	77.7	60.8	42.4	84.3	77.7	65.7
Geodesic (global)	81.0	**61.9**	44.8	84.5	**79.5**	66.4
Associative Hierarchical RF [83]	**85.1**	59.9	**50.4**	**87.8**	77.5	**69.7**

Table 5.7: Quantitative evaluation on the CamVid and MSRC-21 datasets based on the unary potentials from [83].

5.3.3 Associative Hierarchical Random Fields

The associative hierarchical random fields of [83] are originally based on the Robust P^n model, but extend it by adding more complex higher level reasoning based on additional classifiers acting on higher level nodes in the graph that are organized in a hierarchical structure. We use the implementation and parameters provided by the authors to run their method on the CamVid and MSRC-21 dataset. For our method, we substitute the output of our unary classifier with the output of the unary classifier of [83].

As we can see from the results presented in Table 5.7, the unary classifier already provides very good segmentation due to the powerful image features used in [83]. Note, however, that those features are much slower to compute. While the 2D Walsh-Hadamard features can be computed in 544 ms per image for the CamVid data set (see Table 5.5), the features used in [83] take 1627 ms per image on the same machine and with a similar level of code optimization.

The results show that the method of [83] delivers better results, according to the global and intersection-over-union measure, while our method is better at the average accuracy measure. Overall, even though our results are slightly worse than [83], our method runs twice as fast. Both our method and the method of [83] are optimized to take advantage of parallel processing on multiple processors and cores, but the parallelization is done on different levels. In our method multiple pixels or cells are processed in parallel so that the system can be used in an online setup, while in the publicly available code of [83], multiple images are processed in parallel. In order to eliminate the effects of the different parallelization approaches, we also performed the timing experiments on only one processor core, with comparable results.

5.3.4 Stixmantics

The multi-cue segmentation method of [119] and the Stixmantics method [120] are evaluated on the Daimler Urban dataset (see Table 5.8), with both being real-time systems with the same goal as our paper - combine fast texture and 3D features for semantic segmentation. Our global geodesic neighborhood method is able to clearly outperform the method of [119]. In comparison to the Stixmantics method, our results are very similar for the evaluation measure of [120]. The method of [120] performs slightly better for dynamic objects, however, it is worth noting, that we process each image separately and do not use temporal information as done in [120], which is especially helpful for dynamic objects.

Method	**Global**	**Average**	**Pascal (all)**	**Pascal (dyn)**	Ground	Vehicle	Person	Building	Sky
Geodesic Combined (global)	**71.5**	76.2	66.1	61.9	75.5	70.9	52.8	**74.2**	56.9
Multi-cue segmentation [119]			56.6	58.8	82.8	63.9	53.6	29.0	53.8
Stixmantics [120]			66.7	**64.0**	**87.6**	68.9	**59.0**	57.6	60.2
Depth-enabled ICF [120]			52.7	44.2	86.2	53.5	34.9	35.1	53.9

Table 5.8: Comparison to related methods on the Daimler Urban segmentation dataset.

Method	**Global**	**Average**	**Pascal**	Building	Car	Door	Pavement	Road	Sky	Vegetation	Window
Unary	75.6	71.6	51.7	70.3	**59.9**	**64.7**	48.5	72.8	97.2	88.6	71.1
Geodesic (global)	**79.4**	**73.5**	**55.8**	**75.9**	57.5	61.7	**52.8**	**74.5**	**97.6**	**89.4**	**78.5**
ICF [44]	70.8	68.6									
ICF (best) [45]	77.2	72.2									

Table 5.9: Quantitative evaluation on the eTRIMS image database.

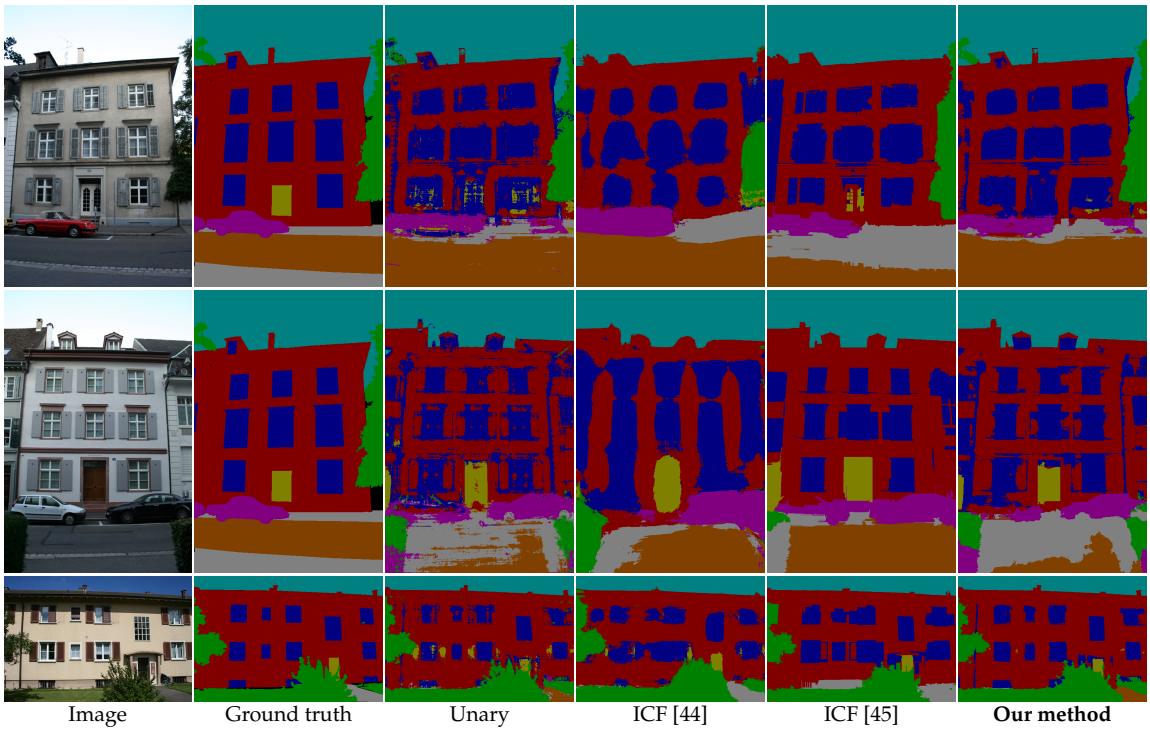


Figure 5.15: Result images on the eTRIMS image database and comparison to the ICF method of [44, 45].

Method	Global	Average	Pascal	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Bicyclist
Associative Hierarchical RF [83]	85.1	59.9	50.4	**86.4**	73.1	**96.6**	74.2	35.5	**96.0**	28.5	53.7	6.5	82.3	25.4
SegNet [9]	87.1	71.1	54.1	84.3	85.7	91.2	90.8	30.2	89.7	68.2	60.5	43.8	**92.4**	44.9
Geodesic Global	**87.9**	**75.6**	**56.1**	79.4	**87.3**	94.0	**92.9**	**46.7**	93.0	**79.0**	**67.6**	**49.7**	92.3	**49.5**

Table 5.10: Comparison between SegNet, Associative Hierarchical RF and our global geodesic method trained on the SegNet probability maps.

5.3.5 Iterative Context Forests

In order to compare to the Iterative Context Forests (ICF) method of [44, 45] we run our method on the eTRIMS image database [77] which is also used for evaluation by the authors of ICF. The authors of [45] make use of the powerful Geometric Context method of [64] to create a set of base features. In order to allow for a better comparison we integrate the geometric context features in the feature vector of the unary classifier, by taking the probability of each of the 8 geometric classes for each pixel directly as a feature.

The quantitative results of our method are shown in Table 5.9 and some example result images are shown in Figure 5.15. Our base unary features cannot compete with the ICF method delivering inferior results. By running our complete pipeline, however, we get clearly better results than those reported in [44, 45] showing that our approach is indeed able to capture context relations well.

Using the Daimler Urban dataset, we can compare to ICF as well. The original method is extended by [120] to also handle depth data, so that it can take advantage of the disparity maps in the Daimler Urban dataset. In Table 5.8 we show the numbers reported in [120]. Even though, the ICF is more flexible in learning context relations, our local and global neighborhoods are able to summarize the context information well enough and again deliver better results.

5.3.6 Deep Convolutional Neural Networks

Recent development in Deep Learning methods have led to significant increase in performance on semantic segmentation problems, mainly due to the ability of convolutional neural networks to learn expressive features from data instead of relying on hand crafted features. However, many methods lack the ability of using global context or work on a smaller scale image as discussed in detail in Section 3.3. Here, we want to explore if our method can be used to compensate problems of deep learning methods by learning context relations on top of the class predictions of existing methods.

In the experiment, we use the SegNet [9] network and the CamVid dataset. We use the pre-trained model and implementation as provided by the authors. We process each image with SegNet and store the class probability distribution of each pixel from the last layer of SegNet. This distribution is then used in our method as a unary potential, similar to the approach used in Section 5.3.3.

The results are summarized in Table 5.10, where we also added the results of [83] for comparison. We can see that our global geodesic model is able to provide a significant

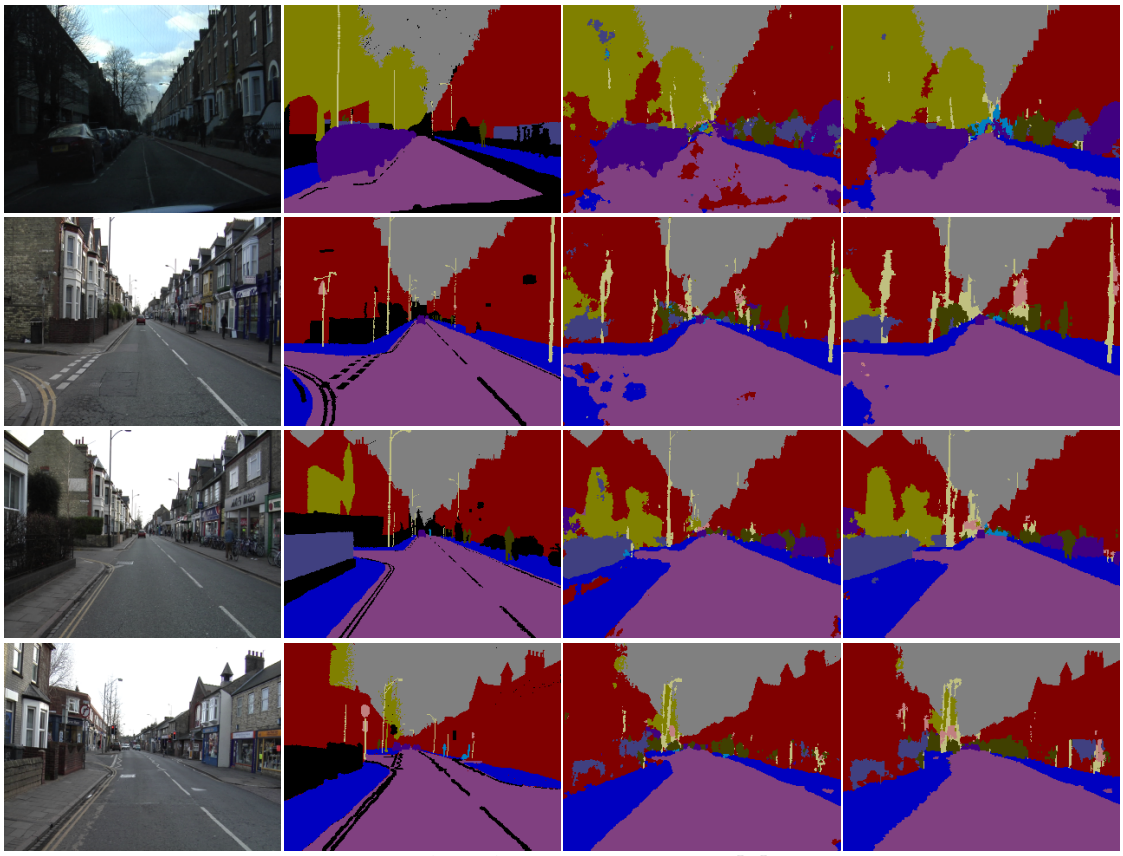


Figure 5.16: Comparison of results on images from SegNet [9] and our global geodesic method trained on the SegNet class predictions on the CamVid dataset. Our method is able to better recover the details roof tops.

increase of up to 4.5% on all 3 evaluation measures and especially on many of the under-represented classes. We can also see that the features learned by the deep convolutional network are much more powerful than the state-of-the-art hand crafted features from [83].

Looking at the resulting segmentation images in Figure 5.16 we can observe that the improvement comes indeed mostly from problems that can be resolved by using context. Our method is able to filter out many large regions in the middle of the road which were wrongly classified as building or sidewalk by SegNet, because of the ability of our method to learn global context. The advantage of our method with respect to local context relations can be observed around object boundaries and especially on the roof tops of the buildings, where our method can better recover the fine details.

5.4 Conclusion

In this chapter we presented a two-stage classification framework for semantic image segmentation of both 2D and 3D images. We showed how to define local and global pixel neighborhoods based on geodesic distance that are able to model local and global context relation between image regions. We also introduced a compact histogram feature, which summarizes context information in those neighborhoods which allows for fast training and evaluation.

We evaluated our method on six challenging datasets containing both 2D and 3D information and gave detailed insights how different parts of our model work. Our method achieves competitive results comparable to related state-of-the-art methods, while at the same time being very fast. We showed that the proposed framework can automatically learn local and global context as well as co-occurrence relations, which enables it to improve the segmentation quality even on a recent deep learning method.

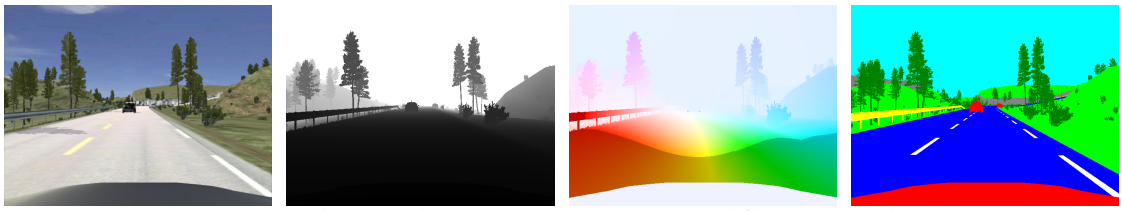
6 Framework for Generating Synthetic Data

As discussed in the chapters before, enabling semantic segmentation methods to learn context is important for achieving good segmentation results. For this, however, we also need to train our models on data that accurately represents the situations we are interested in and contains enough samples of all relevant context relations that need to be learned. The same applies also for the test set, which is typically used to verify that the method has indeed learned to recognize all relevant scenarios. Furthermore, high-quality data is also important for a thorough and objective evaluation and comparison of different methods. In this chapter we focus on the problem of generating synthetic ground truth data for semantic segmentation.

Nowadays, recording large amount of high-quality images and video is a relatively easy task, because cameras are very cheap and easily accessible. However, collecting ground truth data can be very challenging. Manual labeling of bounding boxes, lane markings or multi-class pixelwise annotations is possible, but very time consuming and expensive. For some image modalities, different reference sensors can be used. Infrared projector devices, like the Microsoft Kinect, or multi-camera systems can be used to acquire high-precision depth maps indoors. For outdoor scenes stereo cameras or laser scanners can be used, however they typically lack in precision or resolution respectively. Collecting ground truth data for optical flow or scene flow is even more difficult, because there is no sensor that can directly measure it and manual labeling is practically impossible.

One way to easily obtain large amounts of high-precision ground truth data is to generate it synthetically. Synthetic images are often used during the development of new methods in order to better understand some of their properties. For example [66] use a synthetic rotating sphere for the evaluation of scene flow and [147, 151] use synthetic traffic sequences rendered with POV-Ray to evaluate stereo and optical flow algorithms for driver assistance. Furthermore, benchmarks like the popular Middlebury dataset [10] or the Syntel dataset [20, 153] employ synthetic images for the evaluation of optical flow and [19] use synthetic scenes for the evaluation of background subtraction algorithms. Synthetically generated human models in different poses are used in [126] as training data for a successful human pose estimation method. In the field of semantic segmentation [113] showed that synthetic images from a video game can be used to improve the performance on a real world dataset.

In this chapter we present an open-source framework for generating synthetic data in multiple image modalities with corresponding pixelwise semantic annotations with focus on driver assistance and autonomous driving (see Figure 6.1). The framework is based on the open-source driving simulator VDrift [1], which provides a realistic rendering engine and physics simulation. Having full access to the source code and the 3D models of the simulator, we are able to modify it, so that we can generate not only camera images, but also high-precision depth and optical flow maps, pixelwise semantic annotations and to record the exact camera pose for every frame. The proposed framework also allows the



a) Camera image b) Depth image c) Optical flow d) Segmentation

Figure 6.1: Example images in 3 different modalities and the corresponding pixelwise annotations generated with the proposed framework. In the flow image, pixels with flow of more than 10 px are shown in darker colors. More examples are available in the supplementary materials.

user to define specific traffic scenarios that can be of particular interest for driver assistance applications. The framework is available for download at `http://campar.in.tum.de/Main/VladimirHaltakov`.

Using the proposed framework, we show how to generate images in 3 different image modalities and evaluate a pairwise CRF model for semantic segmentation. We show how the multiple modalities allow us to explore different feature types. Furthermore, we use the framework to generate ground truth data for two specific scenarios - a front looking camera and a parking camera looking on the side. We show how these synthetic images can be used to train a semantic segmentation model, which is then evaluated on a real world dataset. This model reaches comparable performance to the same method trained on real images, that were annotated manually.

6.1 Publicly Available Datasets

Since we are particularly interested in semantic segmentation of outdoor scenes (and especially in the context of ADAS and autonomous driving), in this section we give an overview of some of the more popular publicly available datasets. Table 6.1 gives a brief overview of the existing datasets and we discuss their advantages and disadvantages in detail below.

Most of the multi-class segmentation datasets, like Sowerby [62], Corel [62], MSRC-21 [128], eTRIMS [77] and the Stanford Background dataset [51], consist only of separate camera images with their corresponding semantic annotations. Therefore, they can only be used in segmentation methods that rely solely on texture information. Furthermore, all of the datasets mentioned above provide a small amount of relatively low resolution images (with the exception of eTRIMS, less than 320×240 pixels).

The PASCAL VOC 2012 [37], SUN [154] and the recent ADE20K [162] contain significant amounts of high-resolution ground truth images labeled in many categories. However, those datasets do not focus on street scenes and also do not contain any depth data.

The CamVid [17, 18] and the Dynamic Scenes Dataset [152], include video sequences which can be used to compute optical flow or structure from motion point clouds. However, these methods do not deliver high-quality depth information. This could be a problem for the evaluation, because it is difficult to tell which errors are caused by the input

Dataset	Image resolution	Video sequences	Depth data	Traffic scenes	Annotated images
eTRMIS [77]	768×512	-	-	-	60
Corel [62]	180×120	-	-	-	100
MSRC-21 [128]	320×213	-	-	-	591
Stanford Background [51]	320×240	-	only 3 classes	-	715
NYU Depth V2 [131]	640×480	yes	Kinect	-	1449
NYU Depth V1 [130]	640×480	yes	Kinect	-	2347
PASCAL VOC [37]	500×333	-	-	-	4369
SUN database [154]	various	-	-	-	16873
ADE20K [162]	various	-	-	-	22210
Leuven [86]	316×256	yes	partially	yes	70
City [36]	640×480	yes	stereo	yes	95
Sowerby [62]	96×64	-	-	yes	104
Dynamic scenes [152]	752×480	yes	-	yes	221
KITTI [8, 46, 47]	1384×1032	yes	stereo, laser	yes	400
CMU RGB-D [99]	600×402	-	laser	yes	372
Daimler Urban [119, 120]	1024×440	yes	stereo	yes	500
CamVid [17, 18]	960×720	yes	motion stereo	yes	700
Cityscapes [28]	2040×1016	yes	stereo	yes	5000
GTA5 [113]	1914×1052	yes	-	yes	24966
Our framework	2560×1600	yes	3D models	yes	> 15905

Table 6.1: Summary of the most important properties of 12 publically available multi-class segmentation datasets and our proposal.

data and which by the model itself. Both datasets do not provide any other ground truth data apart from the pixelwise annotations.

The Leuven [86] dataset and the City [36] dataset provide not only video sequences of road scenes and semantic annotations, but also stereo camera images. While the quality of the depth maps computed with stereo matching algorithms is in general better than with structure from motion methods, the depth data can still contain many errors around object boundaries or in the presence of reflections. Furthermore, the images provided by both datasets have a relatively small resolution and only less than 100 frames are annotated.

In order to deal with the problem of providing high-quality 3D data, several works introduce datasets that use an additional high-precision depth sensor along with the camera: the NYU Depth V1 [130], the NYU Depth V2 [131] and the CMU Driving RGB-D [99] datasets. The first two datasets use the Microsoft Kinect to record synchronized camera and range images in indoor environments. Additionally, many of the frames have pixelwise annotations. However, outdoor scenarios pose significantly different challenges than indoor scenes and therefore those datasets cannot be used to develop or evaluate methods that are required to work in outdoor environments, like in the case of autonomous driving. The CMU RGB-D driving dataset of [99] employs a laser scanner instead of the Kinect, which allows recording of outdoor scenes. However, the images provided in the dataset are recorded at a relatively low frame rate, which makes the computation of optical flow or the usage of temporal information almost impossible. Furthermore, the used laser scanner operates in push broom mode and therefore the depth maps are sparse and do not always overlap with the camera images.

The popular KITTI [46, 47] dataset consists of a large amount of stereo camera video

sequences of traffic scenes synchronized with the output of a 360 degree laser scanner providing precise, but sparse ground truth depth and flow data for the lower part of the camera's field of view. The authors also provide extensive benchmarks for the evaluation of stereo, optical flow, object detection and visual odometry methods. Unfortunately, only a limited amount of images (200 for training and 200 for testing) have been labeled with pixelwise annotations [8].

There are two semantic segmentation dataset that explicitly focus on the domain of ADAS and autonomous driving - the Daimler Urban dataset [119, 120] and Cityscapes dataset [28]. Both datasets contain high-resolution stereo images pairs that allow the computation of depth data. The Cityscapes dataset [28] has become one of the standard benchmarks for semantic segmentation. It contains 5000 images with fine pixelwise annotations and 20000 images with coarse annotations.

The HD1K Benchmark is also worth mentioning, since it provides more than 1000 high-resolution frames (2560×1080) in realistic driving scenarios with ground truth depth and optical flow data, but it doesn't provide pixelwise semantic labels.

The dataset presented by [113] is a synthetic dataset generated using the video game Grand Theft Auto V. The game features a very realistic rendering engine that can simulate different weather conditions. While the authors were able to extract a very large amount of images, the labeling process is not fully automated. Since GTA V is a closed source product, no direct access and modification of the internal models of the game is possible. The authors of [113] propose a method of intercepting the data stream to the GPU and develop a special labeling tool exploiting this information in order to speed up the labeling process significantly to 7 seconds per image on average. While the GTA V rendering engine, being a high-budget commercial game, is superior to the rendering engine of VDrift, we are able to access and modify the VDrift code because it is open-source. Therefore, our framework is able to generate the ground truth pixelwise annotations completely automatically and also deliver depth and optical flow annotations. Furthermore, our framework allows the generation of specific scenarios by placing vehicles at specified locations and recording and replaying multiple vehicle trajectories.

6.2 Framework

We considered several driving games and simulators and we chose VDrift [1] because it features a realistic rendering engine, a sophisticated physical simulation engine and a lot of different track and car models. In addition, we have full access to the source code. As we show below, the last point is essential for the generation of some of the image modalities.

6.2.1 Image Modalities

Since our main goal is to generate data for different image modalities, we adapted the rendering pipeline of the simulator to suit our needs. We are able to control different rendering settings like lighting and shadow generation and we can access the 3D structure of the scenes. We are also able to control the camera pose relative to the car and in this way to simulate cameras mounted at different positions and orientations. We developed a set of OpenGL shaders that generate different image types efficiently and we are

able to run the simulation in real time. The rest of the section describes in detail each of the image modalities that can be generated by our framework. An example can be seen in Figure 6.1 and video sequences are provided on `https://vimeo.com/haltakov/synthetic-dataset`.

Camera images For the camera images, we use the default rendering pipeline of the simulator, which is based on OpenGL. It employs several rendering techniques like anisotropic filtering, anti-aliasing, motion blur, ambient occlusion, shadows and reflections. The textures of the 3D models in the simulator are also of relatively high quality. This results in camera images with resolution of up to 2560×1600 pixels.

Depth maps From the 3D models of the scene, we are easily able to generate depth images by directly accessing the z coordinate (in the camera frame) of each pixel. Since our implementation is based on an OpenGL shader, we encode the depth information into all 3 color channels of the output image, which leads to a precision of 24 bits per pixel. If the maximum viewing distance in the simulation is set to 1000 meters the resolution of the ground truth depth map is 0.06 mm.

Optical flow maps The generation of ground truth optical flow is more challenging than that of the depth maps, since the required information is not directly available. In OpenGL there are two matrices that have an effect on where a 3D point should be rendered in the image - the model matrix and the projection matrix. While the camera is moving around the scene, the model matrix of each object is updated in order to account for the camera motion, while the projection matrix is usually fixed.

At each frame, we provide the optical flow shader with the current model matrix for each object and the model matrix from the previous frame. With this information, we can compute the 3D movement vector of each point from the previous frame to the current one. By projecting this 3D vector onto the image plane of the camera, we get the corresponding optical flow for each 3D vertex, but we still have to compute the flow value for the points of each triangle defined by 3 vertices. Interpolating the flow between the vertices in 2D would lead to wrong flow values for the triangle's points. Instead, we interpolate the 3D position of each triangle point in the current and in the previous frame and then compute the correct flow value for each pixel.

As for the depth maps, we encode the flow into the 3 color channels of the output image with a precision of 24 bits per pixel. The flow values in x and y direction are encoded into separate images, from which the original flow can be easily reconstructed. This means that in the case of sequences with a maximum flow value of 100 pixels, the resolutions of the ground truth data is 0.6×10^{-5} pixels.

Pixelwise annotations In the case of pixelwise semantic annotations, each of the pixels in the image should be assigned a class from a set of predefined classes $\mathcal{L}$ that we are interested in. Here, we define the set $\mathcal{L}$ that consists of 7 classes: SKY, TREE, GRASS, ROAD, MARKINGS, BUILDING and CAR. Unfortunately, VDrift does not support different semantic information for all of the object types listed above. For example, in the 3D model of the track, trees and buildings are indistinguishable, because they are simply modeled as 3D

meshes. The road markings do not even have separate 3D structure at all and are just painted on the ground texture. Therefore, the best way to distinguish the different object types is by their texture. We assigned a unique RGB color value to each of the classes above and modified the textures of several tracks by painting them uniformly with the corresponding colors. Rendering the scene with those textures results in a scene where each object is painted in the color specifying its class. In this case, it is very important to switch off all visual effects like lighting, shadows, reflections, anti-aliasing and mip-mapping in order to get the raw color value for each pixel.

6.2.2 Scenario Generation

Since VDrift was originally created as a racing simulator, we modified the game engine in several ways in order to allow for the generation of scenarios that simulate real traffic conditions. Using the default AI settings, the cars would drive as fast as possible around the track. Therefore, we modified the replay system of the simulator so that we are able to manually record the movement of several cars separately and then combine all the replays into one. In this way, we can create arbitrary scenarios of multiple cars driving like in normal traffic.

We also added the possibility to put stationary vehicles at arbitrary positions on the map. In this way, we can simulate parked cars for more realistic scenes.

6.3 Modalities Comparison for Semantic Segmentation

The ground truth data that we can generate using our framework gives us the possibility to explore the effects of using different image modalities for semantic segmentation. This is not possible with any of the existing datasets because they either do not include all 3 image modalities, the quality of the data is relatively poor or the amount of the training data is limited.

For the experiments, we created 25 sequences on 5 different tracks simulating a car driving on country and urban roads. Of those sequences, 12 were used for training and 13 for testing. The virtual camera was mounted behind the windshield of the car like a typical camera used for driver assistance systems in current vehicles. For each frame, we generated images in all 3 possible modalities: camera images, depth maps and optical flow maps and the ground truth pixelwise semantic annotations. Running the simulation at 30 frames per second we generated 7905 images. However, since at this frame rate consecutive images are very similar, we took only each 5th image, which results in a training dataset of 669 images and an evaluation dataset of 912 images.

6.3.1 CRF Model

For the multi-class image segmentation task we use a simple pairwise CRF model, similar to the one we presented in Section 4.1. Each pixel of the image is first classified with a JointBoost classifier [142] based on various features extracted from the image and then inference is performed using the α-expansion algorithm [15] on the pairwise graph with the pairwise potentials based on the contrast sensitive Potts model as in [15]. We use a fairly

Model	Global	Average	Road	Marking	Building	Grass	Tree	Car	Sky
T	89.0	70.5	79.5	**20.0**	61.9	79.2	73.6	88.4	**90.9**
D	80.6	58.8	88.4	0.0	54.2	68.0	25.8	92.3	83.1
F	75.0	49.3	82.3	0.0	52.8	19.4	16.7	91.7	82.5
T+D	**91.2**	**72.2**	88.1	10.0	72.5	82.0	70.3	92.5	90.2
D+F	79.8	57.9	84.5	0.0	55.3	73.5	15.1	92.1	84.6
T+F	90.1	70.9	84.7	10.4	64.0	79.4	**74.3**	93.1	90.6
T+D+F	91.0	71.2	**88.8**	3.2	**71.3**	**82.7**	68.2	**93.2**	90.9

Table 6.2: Pixelwise accuracies from the evaluation of the CRF segmentation model trained on 7 different combinations of features.

simple CRF setup because the goal is to focus on the effect of incorporating different information like texture, depth and optical flow in the features and to evaluate the contribution of the different image modalities to the segmentation accuracy.

Similarly to Section 4.1.3, for the **texture features** we transform the image into Lab color space and for each color channel we compute the mean and the variance of the first 16 coefficients of the 2D Walsh-Hadamard transform [63] at several scales of 8, 16 and 32 around each pixel. The **depth features** consist of the 3D coordinates of the corresponding 3D point, the coordinates of the surface normal at this point and the Fast Point Feature Histogram (FPFH) of [117]. For the **flow features**, we use the 2D vector of the flow directly. In order to incorporate local context information, we also include the 2D coordinates of the pixel.

The segmentation is performed on cells of 8×8 pixels instead on each pixel, but the evaluation is done at the pixel level.

6.3.2 Results

Here we analyze the importance of different image modalities for the segmentation performance. We train and evaluate 7 variants of the CRF model described above by giving it access only to the features of different subsets of image modalities: texture (**T**), depth (**D**), flow (**F**), texture and depth (**T+D**), depth and flow (**D+F**), texture and flow (**T+F**) and all three together (**T+D+F**). The location features are used in every configuration. In Table 6.2 we report the global segmentation accuracy over all evaluation images, for each class individually and the average precision over all classes. In Figure 6.2 we show an example of the segmentation computed by all 7 models. A video sequence with comparison of the results can found on `https://vimeo.com/haltakov/synthetic-dataset-results`.

Using only the texture features (**T**) the CRF model is already able to achieve very good results - 89.0% overall accuracy. However, for classes with big variations in appearance, like BUILDING or CAR (compared to the other configurations), the performance is not that good, which can also be seen on the result images. Using the depth (**D**) or the flow features (**F**) alone leads to relatively poor results - 80.6% and 75.0% respectively. While the numbers may not seem too bad, we can see from the resulting images that the segmentation is much worse than in case of using only the texture features. Traffic scenes often have similar structure - sky in the upper part of the image, road in the lower part and combination of

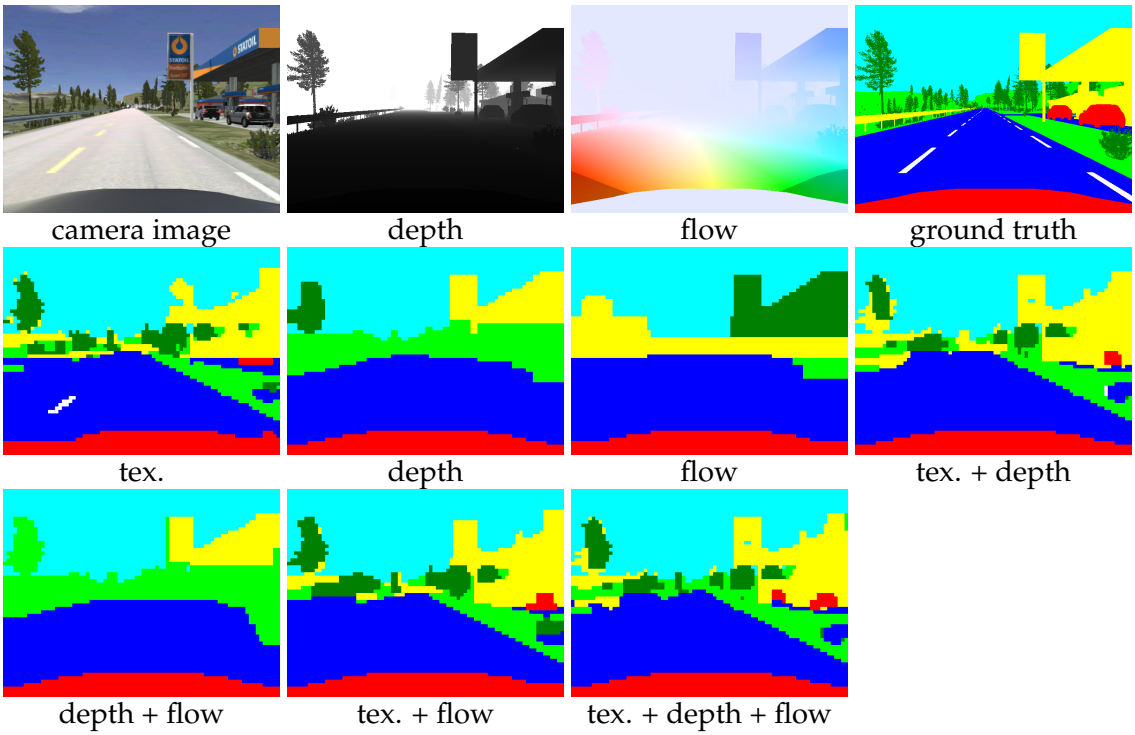


Figure 6.2: Input images in 3 different modalities, the ground truth annotation and the output segmentation of all 7 models for one frame from our evaluation dataset.

buildings, trees and cars in the middle part. This allows the CRF model to achieve high accuracy rates relatively easy relying on the location features, but in order to evaluate the real quality of the segmentation, one should look into the details. For example, we can see both from the quantitative analysis and from the result images, that when using only depth or flow, the model has difficulties distinguishing between flat surfaces like ROAD and GRASS and the road markings cannot be detected at all. The average performance of those models over all classes is therefore also low.

It is not surprising to see that combining the texture features with either depth (**T+D**) or flow (**T+F**) features results in increased overall segmentation accuracy - 91.2% and 90.1% respectively. In most of the classes we see significant improvements, especially for ROAD, BUILDING and CAR. This is the case because those classes vary a lot in their appearance, but have well defined shapes, which can be better captured by the depth or flow features.

Combining all 3 feature types (**T+D+F**) gives overall accuracy of 91.0% that is only slightly worse (by 0.2%) than when using texture and depth features. This suggests that the information encoded by the depth and flow features is similar. This is also confirmed by the model that uses them together (**D+F**). When combining the two modalities, no improvement is observed. However, in the case of more dynamic scenes this could change.

In conclusion, we can say that it makes sense to use depth or flow features along with texture features extracted from the camera images, because they can improve the recognition of objects that have relatively well defined shapes like for example cars. When using relatively simple features for the depth and flow modalities, as in our case, it is not ben-

eficial to include all three image modalities, but this may change if more sophisticated features are used.

6.4 Using Synthetic Data for Training

The presented framework gives us the ability to easily generate large amounts of training data in specific scenarios. In this section we generate an application specific ground truth dataset. We use the synthetic data to train a segmentation method and then evaluate its performances on real world data. We focus on the parking space detection application from a sideways looking camera that is described in more detail in Chapter 7. Here we focus on the accuracy and ignore any real-time requirements in order to explore the limits of using synthetic data for training. In Chapter 7 the focus is on developing a method that can perform in real-time on the road and therefore the results may differ.

6.4.1 Dataset

We have generated a real world dataset by recording images for a sideways looking camera of a moving car and labeled 220 images per hand (see Chapter 7 for more details). Using our framework, we generated similar scenarios automatically positioning randomly generated vehicles along streets in the simulation and driving past them. The positions and orientations of the parked vehicles and the gaps between them were varied randomly in order to achieve more realistic scenarios. Note that this is not possible without having full control over the whole simulation.

Since our real world camera is calibrated, we adjust the simulated camera to deliver exactly the same images. Furthermore, we use the image sequences to generate stereo disparity maps by using the stereo method of [155]. In total, we generated 24 different sequences with about 9300 ground truth images. A visual comparison between the real and the synthetic images is shown in Figure 6.3.

6.4.2 Evaluation Setup

To evaluate the ability to train on synthetic data, we train two different models - one using only real data and one using only synthetic data. The test dataset, however, contains only real images. Since the amount of real world labeled data is small we perform 5-fold cross validation in each experiment.

For the segmentation method, we use our global geodesic model from Chapter 5, in which we employ both texture and 3D information from the computed disparity maps. We also present results on training only the unary classifier with different data sources: only texture images, only depth images and both combined.

6.4.3 Results

The results are summarized in Table 6.3. As expected, just training a unary classifier on synthetic data and testing on real data delivers worse results than training on real data, even though the amount of training images is smaller. The difference is, however, smaller when using only 3D data, since the difference between real and synthetic data there is not

Method	**Global**	**Average**	**Pascal**	Road	Marking	Vehicle	Background
Training on Real Data							
Unary Dep	91.5	69.5	63.9	96.9	1.0	87.3	92.7
Unary Tex	90.2	84.4	75.8	93.3	68.2	85.8	90.2
Unary Tex Dep	94.8	89.4	83.2	96.5	73.0	93.3	94.7
Geodesic Global	95.4	93.1	85.0	95.8	85.5	94.6	96.3
Training on Synthetic Data							
Unary Dep	87.4	66.4	58.4	91.5	0.0	89.4	84.6
Unary Tex	75.1	54.0	43.2	91.6	0.4	45.9	78.2
Unary Tex Dep	84.2	63.5	54.8	90.6	0.5	81.5	81.3
Geodesic Global	95.2	91.9	84.1	96.2	81.7	93.9	95.9

Table 6.3: Results from different versions of our method trained on real and on synthetic data. The evaluation was done on real data only.

as big. Using the real texture images gives a significant boost of performance both in the case where only texture data is used and when both modalities are combined.

However, training our full global geodesic model provides interesting results. In this case, the difference between training on real world and synthetic data is almost none: 95.4% vs 95.2% global accuracy, 93.1% vs 91.9% average per class accuracy and 85.0% vs 84.1% Pascal accuracy. This can be explained by the fact that even though the unary classifier makes much more mistakes when trained on synthetic data, the ability of our model to learn context can compensate almost all of those problems. This can also be observed in the resulting segmentation images in Figure 6.4. Here, we see very big differences in the segmentation quality of the unary classifiers, but almost identical results from the global geodesic method, which is able to filter out even very large problematic regions.

The results align well with other reports of synthetic data being used successfully to improve the segmentation performance in real world scenarios [113]. While the image quality of VDrift cannot match the quality of a high-budget game like GTA V, it offers full control over the whole simulation pipeline, which allows us to quickly generate application specific datasets that closely match the real stations.

6.5 Conclusion

In this chapter we introduced a driving simulation framework for creation of ground truth data. This framework allows us to automatically generate large amount of camera images, depth maps, optical flow maps and pixelwise semantic annotations. We also provide a large dataset that can be used to evaluate the performance of segmentation methods using different modalities.

We also show how our framework can be used to generate an application specific dataset that can be used as training data and delivers results comparable to the same method trained on real data.

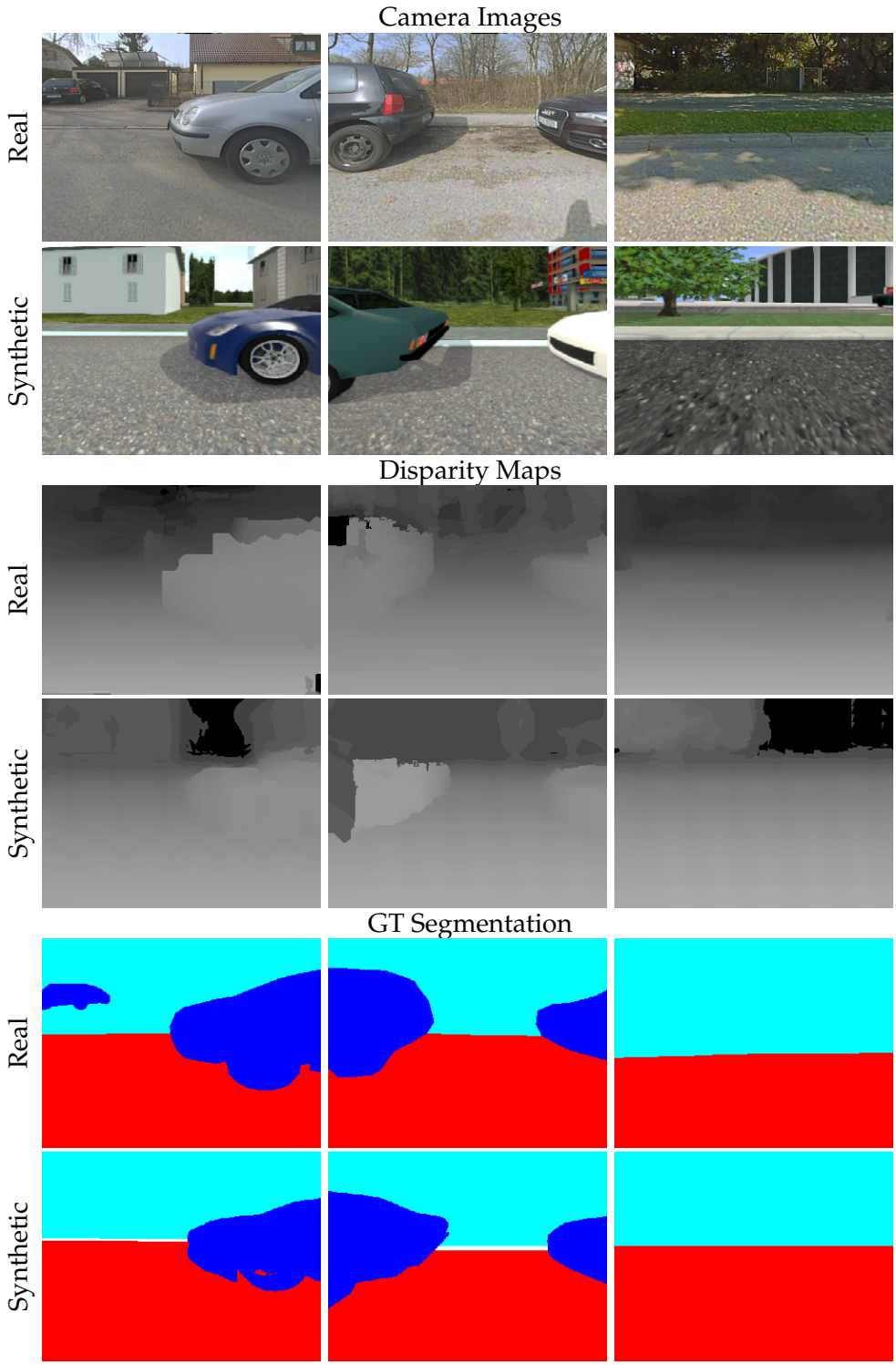


Figure 6.3: Comparison between the camera images, computed disparity maps and ground truth semantic segmentation of real and synthetic images.

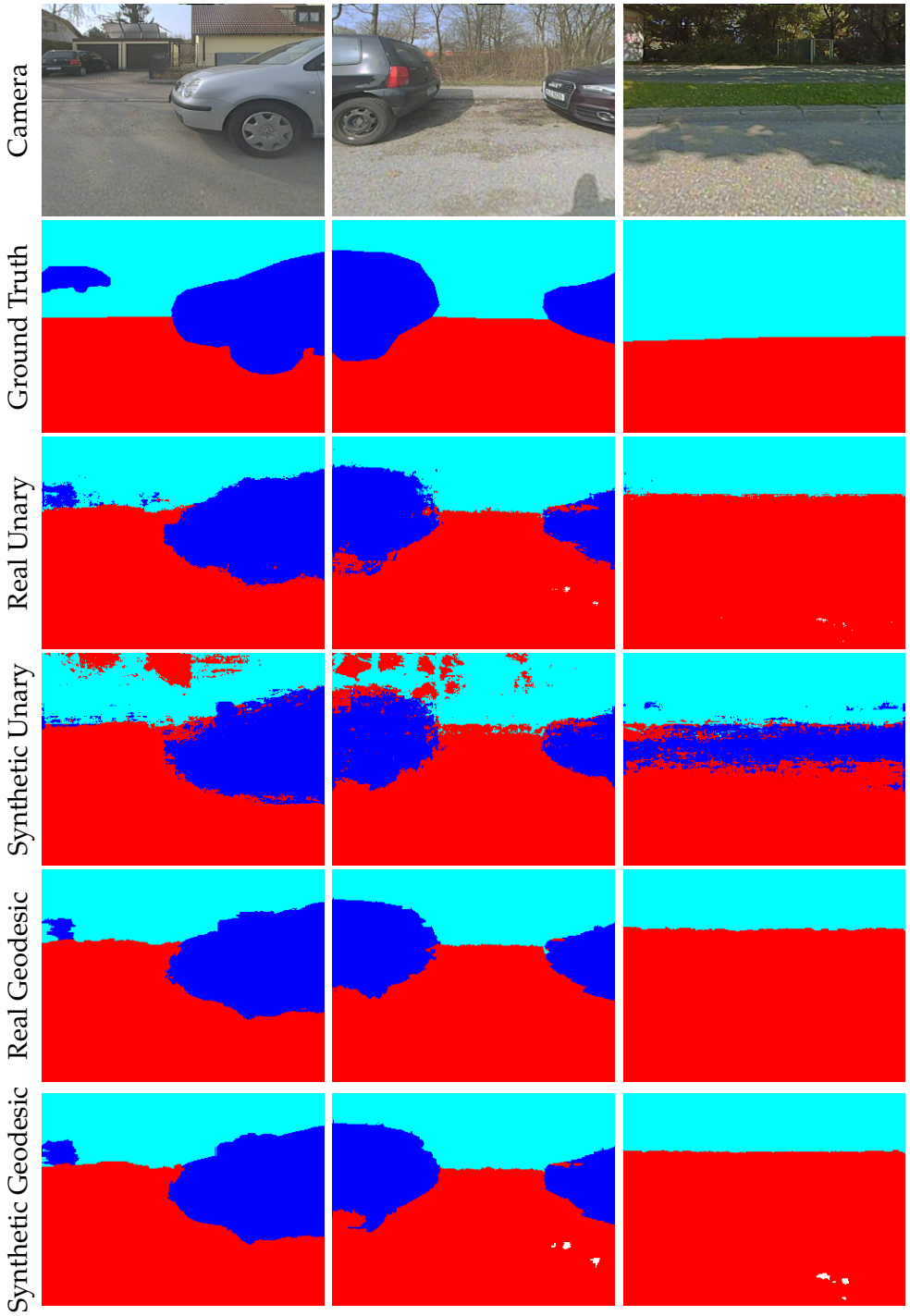


Figure 6.4: Comparison of results from our method trained on real and on synthetic data.

7 Parking Space Detection

In this section we present a real world application for detection of free parking spaces on the side of the road using a side-viewing camera. Fisheye cameras in the vehicle's mirrors are already widely used for parking systems on a wide range of production vehicles. We use a camera in the right mirror to observe the side of the vehicle while driving in order to detect free parking spaces and parked cars. This information can then be used by various advanced driver assistant systems like for example automated parking or autonomous vehicles.

7.1 Dataset

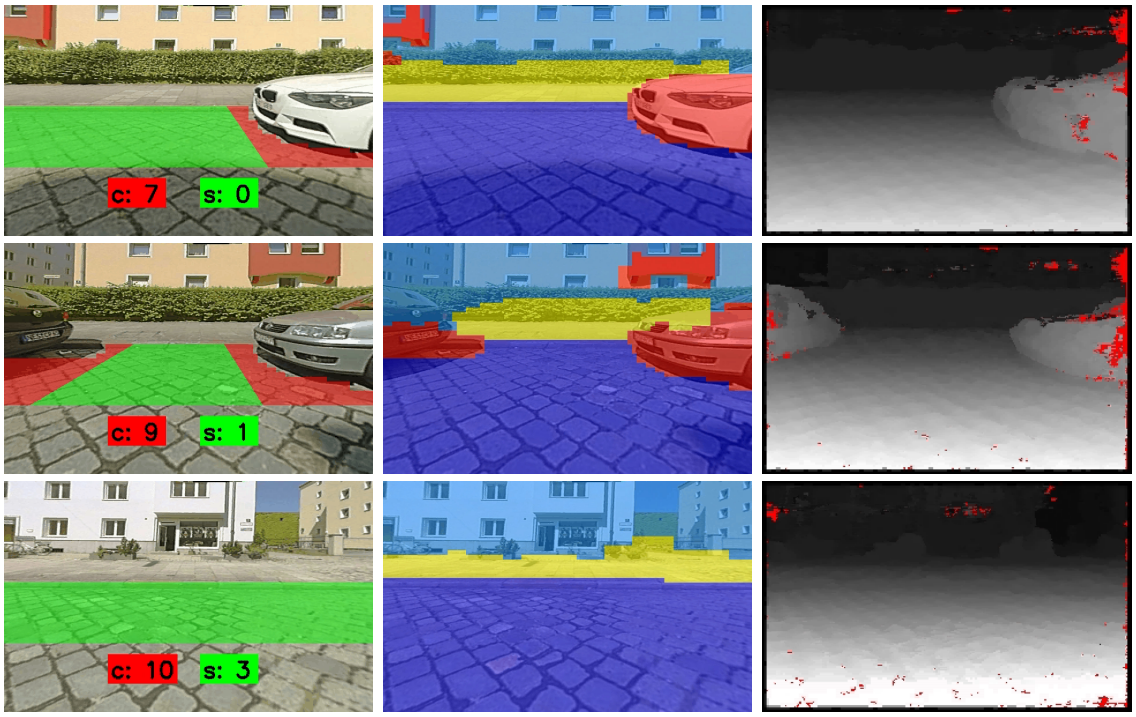
For training and evaluation of the system we recorded a dataset of around 30 km of driving in the city under different environment conditions: cloudy weather, sunny weather and rain. All images are rectified, cropped to the relevant region of interest and scaled down to a resolution of 320×240 (see Figure 7.1 for example images). We labeled 220 of the images pixelwise in 6 semantic classes: ROAD, VEHICLE, LANE MARKING, HORIZONTAL BACKGROUND (like sidewalk, grass areas) and VERTICAL BACKGROUND (like buildings and sky). Since the amount of data is relatively small, we perform 5-fold cross validation using 80% of the data for training and 20% for testing in each run.

Along with the camera images, we also compute 3D information using structure from motion. Because the environment on the side of the car in smaller streets is predominantly static, this works well in most situations. We calibrated the camera and rectified all images to be perpendicular to the direction of travel. For the computation of the disparity map, we employ the stereo fusion method of [146] that uses multiple frames in order to generate robust disparity maps even in challenging outdoor conditions (see Figure 7.1).

7.2 Detection of Parking Spaces and Parked Vehicles

The first step in the parking space detection pipeline is to apply our global geodesic method from Chapter 5 to a camera image taken each 500m. The resulting segmentation provides us with the limits of the drivable area (free space) and also with the location of other parked cars.

After that, we extract the border of the ground plane (the class ROAD), which defines a boundary for the free space around the vehicle. Since we calibrated the camera with respect to the ground plane, we can now compute the 3D position of each point belonging to the ground under the assumption that it is flat. This allows us to measure the distance of each point on the border of the ground plane to the vehicle. Furthermore, from the semantic segmentation we also know the type of object that is beyond that border. We



Detected parking spaces | Semantic Segmentation | Motion stereo disparity map

Figure 7.1: Parking space detection application build on the proposed semantic segmentation method.

Method	Global	Average	Pascal	Road	Lane Marking	Car	Back. (hor.)	Back. (vert.)
Unary Texture	78.3	73.8	50.3	84.8	**92.4**	82.8	27.4	81.7
Unary Depth	78.2	56.0	46.2	**95.5**	1.2	73.7	36.7	73.1
Unary Combined	80.2	75.7	53.2	87.5	90.4	83.7	35.3	81.3
Geodesic Texture (global)	81.7	80.0	56.2	80.7	90.7	**90.8**	53.6	84.5
Geodesic Depth (global)	79.9	75.9	53.9	79.5	76.2	86.4	**53.9**	83.7
Geodesic Combined (global)	**82.0**	**80.0**	**56.5**	81.7	90.0	90.7	53.1	**84.5**

Table 7.1: Quantitative evaluation on the Parking Space Detection dataset.

split the space next to the car in 20 cm wide sections that can be one of the following 3 classes:

- PARKED CAR - if the segment is limited by another segment of class vehicle,
- FREE - if there are at least 2 m to the ground plane border,
- NOT FREE - if there are less than 2 m to the ground plane border.

Using the vehicle movement data, we can fuse the information from multiple frames in order to find the number of free parking spaces and the number of parked cars, assuming that a space is free if it is at least 6 m long.

In order to achieve real-time performance, we make several optimizations to speed up the segmentation method. As for the Daimler Urban dataset and the KITTI dataset, we do not classify each pixel, but cells of 8×8 pixels. Furthermore, we compute the geodesic neighborhood for a smaller number of pixels - 100 instead of 200. With this configuration, we are able to segment one image in 38 ms on a CPU, which is enough for the method to run in real-time in our test vehicle.

In Figure 7.1 we present some example results images, showing the camera images, the compute depth maps, the segmentation results and the result of the parking space detection application. More results can be seen as a video on `https://vimeo.com/haltakov/parking-space-detection`.

7.3 System Performance

We evaluate the performance of the system on two levels: segmentation accuracy and application accuracy. For the segmentation accuracy we use the same measures as for the evaluation of the datasets in Section 5.2 - global pixelwise accuracy, average per class accuracy and the average per class intersection over union. The quantitative results are summarized in Table 7.1. Here, we can again observe the same effects as for the other datasets: combining texture and depth information gives a significant boost over each of the modalities used separately.

Additional to the segmentation accuracy, we also evaluate the ability of the system to recognize parking spaces, which is directly related to the performance of the segmentation method. We compare the output of our detection system to a human counting the parking spaces and the parked cars over several sequences of total length of approximately 2.5 km.

The average recall rate for parked cars is 98.2% at 1.6 false positive detections per 1000 m, while the average detection rate of the free parking spaces is 83.6% at 0.4 false positives per 1000 m. The parking space detection system is slightly biased towards detecting cars, because the classifier tends to detect unknown objects like trash cans or bus stops as cars. This problem can be resolved by adding more training images to the dataset.

7.4 Conclusion

In this chapter we demonstrated how our semantic segmentation method can be applied to a real world automotive application for detecting free parking spaces and parked cars on the side of the road. We also show that our method can be easily tuned to perform in real time on a CPU, while sacrificing little performance.

8 Traffic Lights Detection

Another practical application we address in this thesis is traffic light detection. The position and the current state of the traffic lights in front of the vehicle is a valuable information for many safety and comfort driver assistance functions. Red light running is a major safety problem, with estimated 165,000 motorists, cyclists and pedestrians injured in the USA every year, a lot of which fatal [67, 2]. Similar studies in Germany [3] show that 7,356 incidents with people or property damaged happened in 2013 because of disregarding traffic signals at intersections. Furthermore, traffic lights detection is needed in order to enable autonomous driving in cities and on country roads. While most other objects can be detected by more than one sensor in the vehicle or looked up in a high-definition map, the state of the traffic light can only be observed by a camera.

In this chapter we focus on the problem of detecting the presence and the state of traffic lights from camera images both at day and at night. Day and night scenes pose fundamentally different challenges for visual traffic light recognition. At day, the structure of the traffic light is well visible, but the light source can be difficult to detect due to the presence of many other bright image regions especially in sunny weather. Furthermore, traffic lights are relatively small in size compared to traffic signs or other road users, which makes the detection at large distances difficult. In contrast, at nighttime, light sources are visible from a very high distance, but since the traffic light box is usually not visible in the camera image (or only at very short distance), there is only a limited textural support to distinguish the traffic lights from other light sources like street lamps, vehicle brake lights and advertisements. Figure 8.1 shows examples of such difficult situations.

Our method is based on our semantic segmentation method from Chapter 5 and can handle both day and night situations in a unified manner. We train the classifier on different images for day and night, but all the other parts of the pipeline remain unchanged. While there is a vast amount of literature on traffic light recognition, only very few vision methods deal with both day and night situations [73, 134]. A detailed overview of the

Figure 8.1: The same scene recorded on a sunny day and at night showing the challenges for a traffic light detection system that needs to operate in all conditions.

related work is given in Section 8.1.

Our method consists of two main stages. First, we use a pixelwise semantic segmentation to find image regions that are potential traffic light candidates. While similar image segmentation steps, usually based on color thresholding, are used by many other systems, we show that more advanced machine learning methods like our semantic segmentation approach can provide more robust candidates, because they can extract more context information from the scene. In a second step, we compute multiple color and geometric features on the regions found in the first step, which are then used by another classifier to confirm or reject the candidates and to also determine the current color of the traffic light. Additionally, a tracking algorithm is used to enforce temporal consistency.

The proposed system is evaluated on two datasets with 57 intersections recorded both at day and at night in order to show that we can handle both scenarios using the same approach. Furthermore, we also present our results on the publicly available dataset of [30], which contains only daytime recordings.

Our main contribution is a unified framework for real-time traffic light detection both at day and at night based on semantic segmentation to generate traffic lights proposals and the subsequent classifier used to confirm or reject those candidates based on geometric and color features.

8.1 Related Work

We divide the related methods in three groups based on the situations they operate in: at day, at night or both. While there are works that rely on high-accuracy maps as a prior for the traffic lights position in the camera image [38, 42], here we focus on purely vision based systems, because they pose their own unique challenges.

For a more detailed review and comparisons between different methods, we refer to two recent survey papers [33, 70] that provide a good overview over the state-of-the-art.

8.1.1 Detection at Day

Most of the related works focus on traffic light detection at day. Many methods rely only on pure image processing by applying color segmentation followed by geometric and visual filters [21, 34, 49, 103, 124, 140]. Those methods may deliver good results if the light shape is clearly visible, but they don't scale well to various traffic light shapes and night conditions. The evaluation provided on those methods is also very limited and sometimes only qualitative results are presented.

The methods described in [30, 149] rely on template matching in addition to image processing techniques, which increases the robustness of the system in some situations, but they work only during the day. Both methods are evaluated on the dataset or part of it that is introduced in [30], which we also use for the quantitative evaluation of our method.

More powerful machine learning methods are employed by [25, 50, 69, 92] in order to learn the appearance of the traffic lights at day. However, at night most parts of the traffic light are not visible, so it is not clear if those methods can be extended to also work in all situations.

Most of the works above provide very limited evaluation based on short sequences of couple of minutes or done only qualitatively, which makes comparison of performance difficult.

The method of [108] is also based on classification using an AdaBoost classifier and is evaluated on one of the recent large public datasets for traffic light detection - the VIVA traffic light detection dataset [70, 108].

Following the great success of deep convolutional networks for object detection and classification, most of the recent methods for traffic light detection are based on deep convolutional networks [11, 118, 150]. While the methods vary in the way that the deep networks are used, they are all focused on day time detection only.

8.1.2 Detection at Night

The big challenge for traffic light detection methods at night is to filter out light emitting objects that are not traffic lights. Several works exist that explicitly focus on the night detection problem either by using template matching methods [39, 88], support vector machines classification [72] or just image processing [32]. However, due to the lack of a publicly available datasets for traffic lights detection with night recordings, those methods are tested only qualitatively or on small non-public datasets.

8.1.3 Detection at Day and Night

The methods that are most strongly related to ours are those that are designed to deal both with day and night conditions [31, 73, 134].

The authors of [73] use a pipeline consisting of image adjustments in the RGB space, thresholding and applying a median filter to detect traffic lights in different weather and illumination conditions. However, the scenarios where this method is applied are limited, because only traffic lights suspended above the road are detected, while many smaller intersections are regulated only by traffic lights mounted on posts on the side.

Another system designed to handle both day and night situations as well as adverse weather conditions is presented by [134]. The authors employ a color pre-processing step, followed by a fast radial symmetry transform to extract candidates and a spatio-temporal consistency check to reduce false positives. While the detection at day is quantitatively evaluated on the dataset of [30], the night detection is evaluated only qualitatively, which makes comparison of the performance in different situations impossible.

The method of [31] is based on a fuzzy clustering to better separate the traffic light colors. During tracking a chain of rules is used to filter out spots that are not real traffic lights. The system is validated during the Public Road Urban Driverless test in Italy [16].

8.2 Challenges for Traffic Light Detection and Recognition

There are many challenges in building a robust image processing pipeline for traffic light detection and recognition that can be used in a driver assistance application or in an autonomous vehicle. In this chapter of the thesis we address the problem of detection and color classification under different lighting conditions, but in this section we also discuss all other important aspects.

Figure 8.2: Examples of different variants over the world.

Robust detection The detection of traffic lights is the basis of the pipeline. The challenge here is to detect the traffic lights robustly under different lighting and weather conditions. Cloudy weather is usually the best case scenario for traffic light detection, because the contrast between the traffic light spots and the surrounding environment is good. On the other hand, sunny weather, especially low sun scenarios, lead to many other bright structures and reflections in the image which usually reduces the detection rate and detection distance. As discussed in the beginning of the chapter, the appearance of the traffic lights change significantly between day and night conditions.

Color classification While in the general case, the difference between the red, yellow and green colors is significant, the lack of standardization of traffic light leads to many different variants of the colors used in different countries, like for example very red yellow lights or bluish green lights.

Variance The difference in appearance of the traffic lights across different countries is huge. While traffic lights in Europe are usually vertical and contain 3 light spots, traffic lights in the USA can be both vertical and horizontal and can contain any number of spots ranging from 1 to 7 in different configurations. Furthermore, different manufacturers use different colors and shapes of the traffic light box and also use different lighting technology. Some examples of traffic lights over the whole world are presented in Figure 8.2.

LED technology LED traffic lights pose a specific challenge to camera systems, because they are usually pulse modulated and it can often happen that the light spots appear darker or completely off in one or more consecutive frames. A robust traffic light detection system should be able to deal with this problem and continue tracking the traffic light.

Distance estimation The distance to the traffic light is an important cue for many applications. The difficulty in the distance estimation is that traffic lights are relatively small in comparison to other objects like traffic signs or vehicles. Therefore, even small errors in the estimation of the position of the traffic light in the image lead to big errors in distance. Furthermore, since traffic lights don't lie on the ground, the vertical position in the image is not a strong cue for the distance of the traffic light to the camera.

Arrows classification Classifying the arrow shape of the traffic light spot is important in order to interpret the direction for which the traffic light is relevant. However, this usually requires a high-resolution camera, because the details of the shape are usually very small. Furthermore, the variation of the light shapes is usually very big across different countries.

Relevance Detecting and classifying the traffic light is an important first step, but in order for a vehicle to react on it, it is important to know which traffic light is relevant for the vehicle. This requires a certain higher level understanding of the scene, like matching traffic lights to lanes, classifying the arrows in the traffic light and interpreting the direction of each lane. Another challenge is to distinguish between traffic lights that are relevant for vehicles from traffic lights relevant for pedestrians, bicycles and public transport, especially at higher distances.

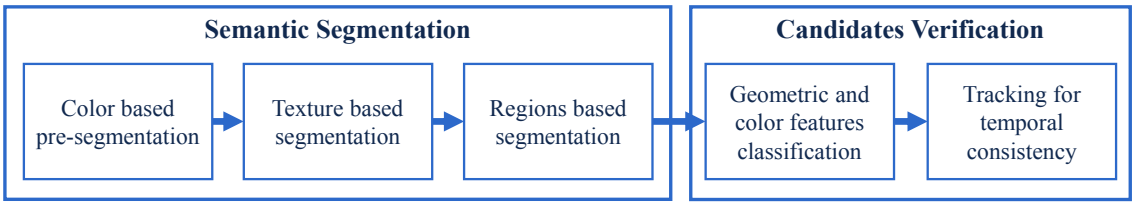


Figure 8.3: Overview of the method pipeline.

Position Autonomous vehicles need to recognize traffic light form a high distance in order to react early enough, but also at the very close distance, when the vehicle is standing first-in-line at the junction. While some countries have traffic lights on both sides of the junction, many others put traffic lights only on the closer side. Cameras with a small field of view provide many pixels per degree and allow traffic lights to be detected at higher distances, but they usually lose the traffic light when it is closer to the camera. On the other hand, cameras with a large field of view (typically fisheye cameras) are essential for the recognition in the close range, but object quickly become very small with increasing distance.

8.3 Method

The general method pipeline is illustrated in Figure 8.3. A semantic segmentation algorithm is first used to label each image pixel and find potential candidate regions in the image. Those regions are then verified by a classifier based on several color and geometric features, which are also used to determine the state of the traffic light. The verification stage also includes a tracking step, which helps to enforce temporal consistency on the detections.

8.3.1 Semantic Segmentation Based Candidates

The goal of this stage is to find regions in the image that are potential traffic light objects. In this stage, having false positives (e.g. candidates that are not traffic lights) is not critical, since the subsequent verification stage is designed to filter them out. The number of false negatives, on the other hand, needs to be low, because missed traffic lights will not be evaluated in the next steps. Nevertheless, a segmentation method that has few false positives is desirable since the verification stage will be both more accurate and more efficient. It is also important to note that our goal is to design a method that will be applicable both at day and at night.

The semantic segmentation problem aims to divide the image into semantically meaningful regions. In our case, we need only two semantic labels: BACKGROUND and TRAFFIC LIGHT CANDIDATE. Our goal here is to label only the light spot of the traffic light and not the whole box, because in most cases the box is not visible in the camera image at night. Furthermore, there are many variants of traffic light boxes worldwide, which also may have a number of spots ranging from 1 to 6 or more.

We employ a three-step semantic segmentation method based on the method described in Chapter 5. Each step follows the same approach: for every pixel, we compute features

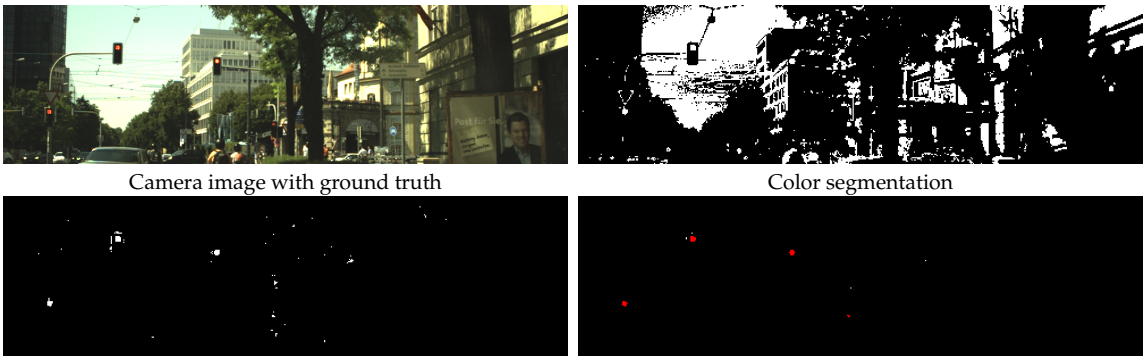


Figure 8.4: Intermediate results of our method at the different stages of the pipeline. At every step the number of traffic light candidates (in white) is reduced. The candidate regions confirmed by the classifier in the last stage are marked in red.

from the image and from the result of the previous steps. Each pixel is then classified based on those features by a JointBoost [142] classifier. The three steps are described in detail below, while in Section 8.4.2 we show how they contribute to the final detection performance.

Color Segmentation

The first step is a simple color segmentation used to improve the runtime of the method. Instead of tuning the color thresholds by hand, we employ a classifier that uses only the color of the pixel in the Lab color space as input and is biased to have few false negatives on traffic light candidates by giving the traffic light pixels a very high weight. Figure 8.4 shows an example output of this step. The subsequent steps ignore all pixels that were labeled as background.

Texture Segmentation

In the second step, the pixels are classified based on the texture in their surrounding area. For this we compute a feature vector $f(x_i)$ based on the 2D Walsh-Hadamard Transform [63] as already presented in Section 5.1.4.

While the classifier trained on texture features is already able to provide good results, the shape of the regions may not be very robust due to small pixel errors around the borders (see Figure 8.4). This happens because the classifier takes the decision about the class of each pixel individually and independently of the labels of the neighboring pixels. This problem is addressed in the next step.

Region Segmentation

In this step, we apply our neighborhood classification method from Chapter 5. The region classifier considers not only the pixel of interest itself, but also a set of related pixels

Feature	Values	Description
Mean (RGB)	3	Mean of the region pixels computed separately for each color channel.
Mean (Lab)	3	
Std. deviation (RGB)	3	Standard deviation of the region pixels computed separately for each color channel.
Std. deviation (Lab)	3	
Image position	2	The pixel coordinates of the center of the region.
Area	1	Area of the region.
Orientation	1	Angle between the x-axis and the major axis of the region.
Aspect ratio	1	Aspect ratio of the two sides of the region's bounding box.
Ratio of areas	1	Ratio of the areas of the region and its bounding box.
Y-coordinate-area ratio	1	Ratio between the region's center y-coordinate and its area.
Solidity	1	Ratio between the areas of the region and its convex hull.
Eccentricity	1	Ratio of the distance between the foci and the major axis length of an ellipse that has the same second moment as the region.

Table 8.1: Geometric and color features used for the classification of candidate regions.

called neighborhood. Since the geodesic distance is slower to compute than the Euclidian distance, here we define the neighborhood N_i of pixel i to contain all pixels in a circle of radius 3 around each pixel, which is much more computationally efficient. This formulation allows the classifier to model local context relations, which leads to better segmentation performance and better candidate regions (see Figure 8.4 for an example result).

8.3.2 Candidates Verification

The semantic segmentation method introduced in the previous section learns texture features and label interactions that are characteristic for traffic lights. However, since the classifiers from the segmentation stage classify each pixel individually, it is difficult to model geometric features that describe whole regions, like for example, if the region has a circular shape. Therefore, in the verification stage we train another classifier based on the region geometry and color features. The classifier does not take decisions on the pixel level anymore, but on the region level. Furthermore, we introduce a simple tracking by detection algorithm in order to enforce temporal consistency of the detections.

Traffic Lights Classifier

Each candidate region coming from the semantic segmentation method is classified in the classes BACKGROUND, GREEN, YELLOW or RED traffic light. The input to the classifier is a set of 21 geometric and color features described in Table 8.1. Here, we again make use of the JointBoost classifier, which now operates on regions instead of pixels. The result of the classification is shown visually in Figure 8.4, where only the candidates that were classified correctly are painted in the corresponding color, while the white candidates are rejected.

Tracking

Since the verification method described above operates on individual frames, one can often observe sporadic false detections that last only one or two frames or detected traffic lights can be missed for several frames mainly due to motion blur or due to LED traffic lights flickering in some frames.

To deal with this problem we introduce a simple tracking by detection algorithm to enforce temporal consistency. The traffic lights are detected separately in each frame and then the detections from two subsequent frames are matched based on the distance between them. This allows us to determine the number of frames each traffic lights has been tracked and only traffic lights that were already seen in at least three frames are counted as detected.

8.4 Results

We evaluate our method on two challenging datasets in both day and night situations and present a comparison with two related works. Furthermore, we evaluate the influence of the different steps of our method and its runtime.

8.4.1 Datasets

We use the publicly available dataset of [30] which is recorded at day in Paris and has a length of around 17 minutes and manually labeled bounding boxes in each frame. Since there is no fixed training and testing split we perform a 3-fold cross-validation. In the rest of the section we refer to this dataset as *France Day*.

We also created two additional datasets in order to analyze the performance of our method at day and nighttime. We used a 1 megapixel camera taking images at 16 frames per second mounted behind the windscreen of a vehicle. We defined a city route in Munich, Germany with a length of around 17 km, which contains 57 intersections with traffic lights ranging from side streets to big multi-lane streets. The recordings were done both on a sunny day and at night. All traffic lights have been labeled with bounding boxes around the 3 lights for the day scenes and around the illuminated light only for the night scenes, if the light source is bigger than 5 pixels in the camera image. We refer to these datasets as *Germany Day* and *Germany Night* respectively. Because of the small number of yellow traffic lights in all of the datasets, they are ignored during evaluation.

8.4.2 Method Analysis

Here, we analyze the performance of our method at all steps in the method pipeline and show that every step is important to achieve the final performance.

Semantic Segmentation

For the training of the semantic segmentation method all bounding boxes are first converted into pixelwise labels on the active light spot of the traffic light. The performance of the three segmentation steps is measured according to the percentage of pixels labeled

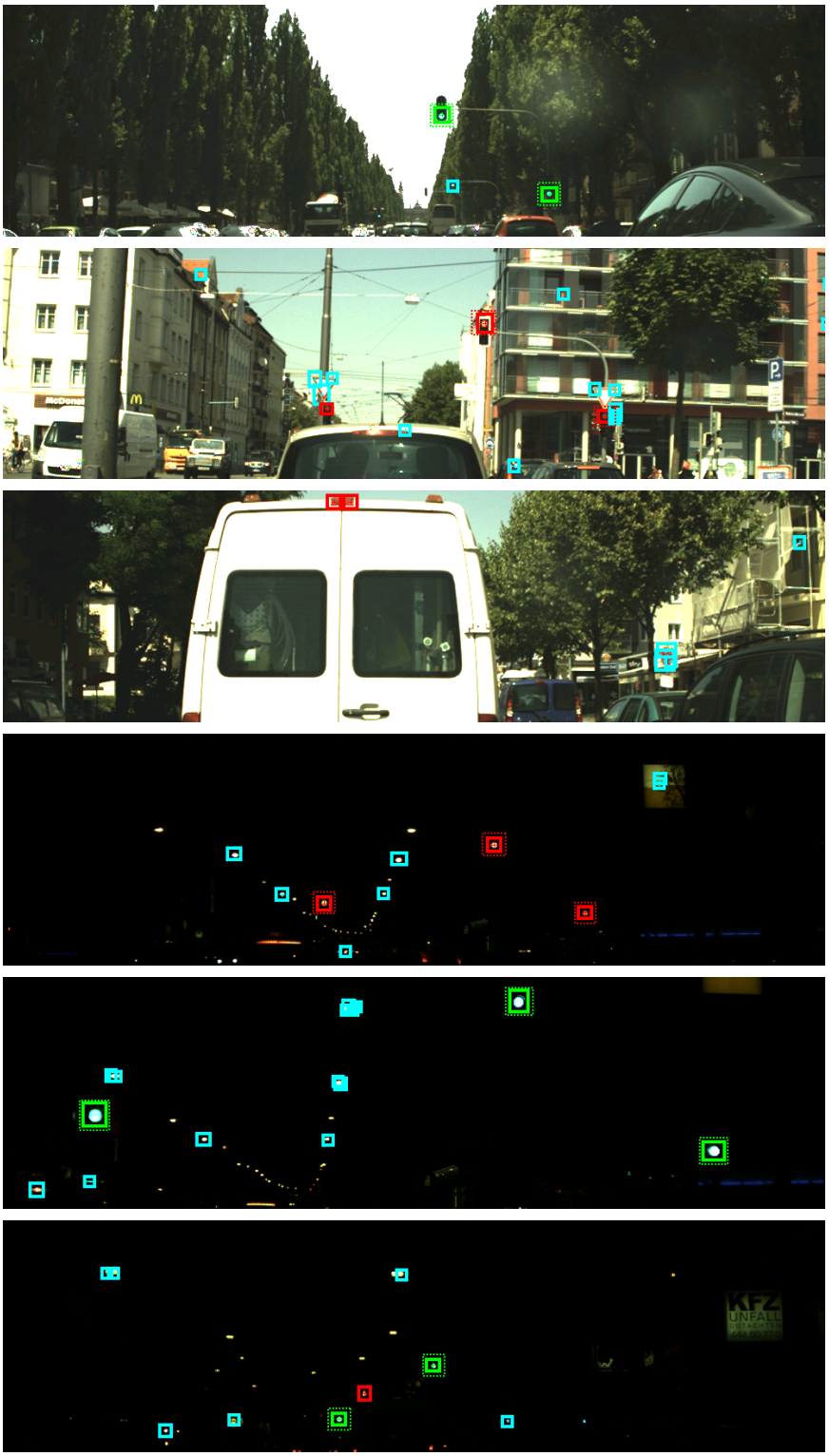
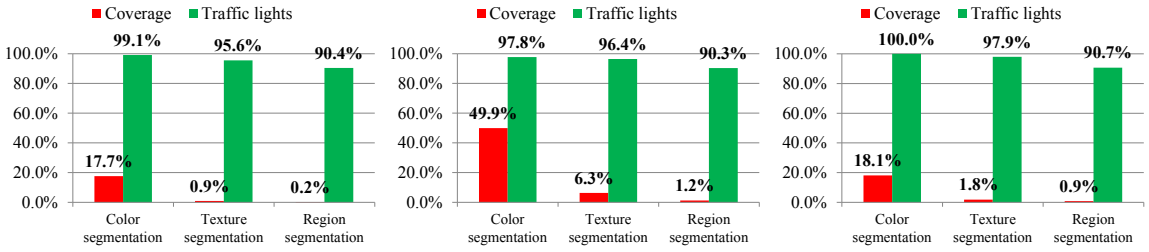


Figure 8.5: Results from *Germany Day* and *Germany Night*. Candidates are shown in cyan, confirmed detections in red or green and ground truth with a dashed box. The last rows for the day and for the night scenes show typical false positives.

Figure 8.6: Results from *France Day* [30]. Candidates are shown in cyan, confirmed detections in red or green and ground truth with a dashed box.



France Day Germany Day Germany Night

Figure 8.7: Results of the 3 steps of the segmentation stage showing the percentage of all image pixels labeled as candidates and the correctly classified traffic light pixels.

Stage	France Day		Germany Day		Germany Night	
	Recall	Precision	Recall	Precision	Recall	Precision
Without tracking	76.1%	63.3%	91.6%	61.3%	91.5%	57.4%
With tracking	71.7%	73.2%	84.3%	71.5%	84.4%	73.8%

Table 8.2: Quantitative results based on frame-wise recall and precision.

correctly as BACKGROUND or CANDIDATE. While this does not directly translate to detection rate for the traffic lights, because some traffic light regions could be only partially segmented, it is a very good indicator of how good each step in the pipeline performs.

From the quantitative results shown in Figure 8.7 we see that with every step in the pipeline the number of pixels labeled as traffic lights ("Coverage") decreases significantly, while our semantic segmentation method is able to retain almost all of the real traffic lights ("Traffic lights"). While the *Germany Day* dataset is more challenging for the simple color segmentation due to the big variety of traffic lights and illumination conditions because of the sunny weather, the region segmentation step achieves results similar to those of the other datasets.

An interesting observation is that after the region segmentation, the segmentations of images at day and at night look very similar. This means that our region segmentation method is able to find a representation of the candidates that is largely invariant to the lighting conditions. This effect can be observed in Figure 8.8 and in the result videos available on `https://vimeo.com/haltakov/traffic-light-detection`.

Candidates Verification

The tracks based recall measure used by the authors of the *France Day* dataset [30] is not suitable for many functions that need a stable tracking of the traffic lights while approaching the intersection, like for example red light warning or autonomous breaking. Therefore, we employ a frame based measure of recall and accuracy, which are more natural for the mentioned functions.

The quantitative results on all three datasets are summarized in Table 8.2 with our method achieving similar performance in all scenarios. Figure 8.5 and Figure 8.6 show some example detections. The tracker is an essential step to reduce the amount of false

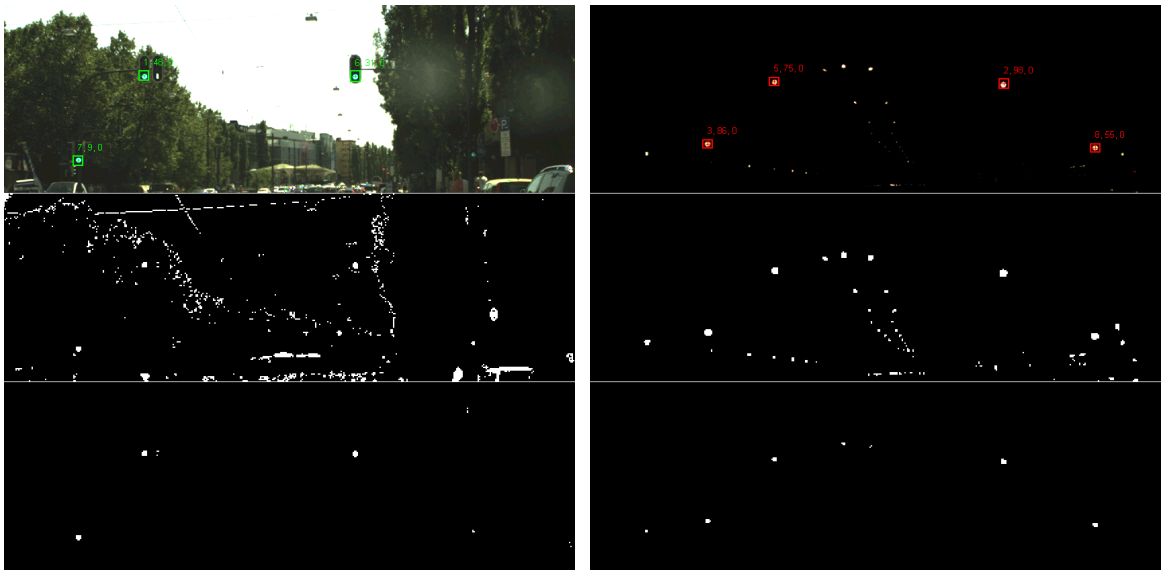


Figure 8.8: Two frames taken in different lighting conditions and their corresponding segmentations after the texture and region steps. Looking at the region segmentations of both frames, it is not possible to tell which was recorded at day and which one at night.

positives both at day and at night, because they tend to appear only for short periods of time.

System Runtime

Semantic segmentation methods can be slow in general, since they need to classify each image pixel. Our three-step semantic segmentation approach, however, filters out many of the pixels in the first step, so that the more expensive texture analysis is performed only on the relevant image parts.

The total runtime of our method is 65ms per frame, with the semantic segmentation accounting for 92% of it. All experiments were performed on a machine with 2 Intel Xeon X5690 processors running at 3.5 GHz. The code is written in C++ without the use of SSE instructions and is only partially parallelized.

8.4.3 Comparison to Related Methods

Two of the related methods [30, 134] have published results on the complete *France Day* dataset so that we can perform a quantitative comparison.

Although the authors of [30] use precision and recall as benchmark measure, they define a computation rule based on temporal tracks instead of frames. This means that one physical traffic light is counted as correctly detected if it is detected in at least one frame during its lifetime. Since our method is able to detect 33 of the 34 traffic lights in at least one of the frames (we also consider the partially occluded ones), the recall of our method is 97.1%, while the authors of [30, 134] report 97.7% and 93.8% respectively. Unfortunately,

the authors of [30] do not give precise description of how they compute the precision measure.

8.5 Conclusion

In this chapter we presented a unified machine learning framework for traffic light detection at different lighting conditions. The used powerful semantic segmentation method is able to provide robust candidates both at day and at night by analyzing the image structure. We also describe several geometric and color features that are used to reject false candidates and to classify the color of the traffic light. An additional tracking by detection step is important for enforcing consistency of the results over time and reducing the amount of false positives.

We showed that our method runs in real-time and delivers good results on three challenging datasets recorded in different illumination conditions and containing data from more than 100 intersections with multiple traffic lights.

In the future we plan to perform further evaluation on two recent datasets: the VIVA traffic light detection dataset [70, 108] and the Bosch Small Traffic Light Dataset [11].

9 Conclusion

In this thesis we explored different ways to enable semantic segmentation methods to learn context relations and we show that considering more context leads to improved segmentation results. We argue that this is due to the fact that images usually contain many ambiguities, which cannot be resolved without using additional information.

First, we introduced a new formulation for the pairwise potentials in a CRF framework, which are based on a classifier instead on the classical contrast sensitive Potts model. We show that this classifier is able to learn to classify edges in the image and discriminate between transitions from the road to road markings or from trees to the sky for example. Therefore, the classifier is able to learn local context relations which are then considered in the optimization step of the CRF, giving the model the possibility to provide smooth segmentation images, but not smooth over smaller objects.

We approached the problem of learning both local and global context by another model, which moves away from the CRF formulation and employs a chain of classifiers instead. Here, we introduced a novel way to define the neighborhood of a pixel by using the geodesic distance, which helps the classifiers to capture local context and better adjust the segmentation to the object boundaries in the image. Furthermore, the same concept is extended to allow the method to learn global context relations as well. We also demonstrated that our method can naturally deal with both texture and 3D information. Another important aspect of this model is that it can be easily tuned to real-time performance on a CPU without much loss in accuracy.

Along with exploring methods to extract more information from the image, we also explored ways to generate new images with ground truth data that can be used to evaluate new methods or to augment existing databases. We presented a generic framework based on the driving simulator VDrift that can be used to easily generate large amounts of high-quality ground truth data like camera images, pixelwise semantic labels, depth data and optical flow data. We show that this synthetic data can be used to train a method that performs as good as a method trained on real data that needs to be labeled by hand.

Finally, we showed how the presented ideas and methods can be used in two real world automotive applications. First, we demonstrated an application for automatically detecting free parking spaces and parked cars on the side of the road. We showed that using context greatly improves the performance of the application. Second, we deal with the problem of detecting traffic lights. Here, the semantic segmentation method allows us to efficiently generate candidate objects both at day and at night, even though the challenges of the traffic light detection problem change a lot with the illumination of the scene. Therefore, we are able to apply the same recognition method independent of the time of day.

Ultimately, we show that learning context is a key factor in achieving good segmentation performance in real world scenarios and that semantic segmentation is a powerful method that can serve as a basis for many driver assistance and autonomous driving applications.

Bibliography

[1] http://www.vdrift.net.

[2] *Traffic Safety Facts 2008*. National Highway Traffic Safety Administration, 2008.

[3] *Fachserie. 8, Verkehr. 7, Verkehrsunfälle.* Statistisches Bundesamt Wiesbaden, 2013.

[4] Federal motor vehicle safety standards; rear visibility. 79 Federal Register 19177, April 2014.

[5] Euro NCAP 2020 Roadmap. http://www.euroncap.com/en/for-engineers/technical-papers/, March 2015.

[6] Euro NCAP Assessment Protocol - Safety Assist. http://www.euroncap.com/en/for-engineers/protocols/safety-assist/, January 2016.

[7] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. In *PAMI*, 2012.

[8] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *BMVC*, 2017.

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *PAMI*, 2017.

[10] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *IJCV*, 2011.

[11] K. Behrendt and L. Novak. A deep learning approach to traffic lights: Detection, tracking, and classification. In *ICRA*, 2017.

[12] J. Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society.*, 1975.

[13] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. *ICCV*, 2001.

[14] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *CVPR*, 1998.

[15] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001.

[16] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti. PROUD - Public Road Urban Driverless-Car Test. In *ITS*, 2015.

[17] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008.

[18] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008.

[19] S. Brutzer, B. Höferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *CVPR*, 2011.

[20] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[21] Z. Cai, Y. Li, and M. Gu. Real-time recognition system of traffic light in urban environment. In *CISDA*, 2012.

[22] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.

[23] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *PAMI*, 2016.

[24] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. In *arXiv:1706.05587*, 2017.

[25] C.-C. Chiang, M.-C. Ho, H.-S. Liao, A. Pratama, and W.-C. Syu. Detecting and recognizing traffic lights by genetic approximate ellipse detection and spatial texture layouts. In *International Journal of Innovative Computing, Information and Control*, 2011.

[26] W. W. Cohen and V. R. Carvalho. Stacked sequential learning. In *IJCAI*, 2005.

[27] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. In *PAMI*, 2002.

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[29] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[30] R. de Charette and F. Nashashibi. Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates. In *IV*, 2009.

[31] M. Diaz, P. Cerri, and P. Medici. Robust real-time traffic light detection and distance estimation using a single camera. In *Expert Systems with Applications*, 2015.

[32] M. Diaz-Cabrera and P. Cerri. Traffic light recognition during the night based on fuzzy logic clustering. In *Computer Aided Systems Theory - EUROCAST 2013*, 2013.

[33] M. Diaz-Cabrera, P. Cerri, G. Pirlo, M. A. Ferrer, and D. Impedovo. A survey on traffic light detection. In *ICIAP Workshops*, 2015.

[34] M. Diaz-Cabrera, P. Cerri, and J. Sanchez-Medina. Suspended traffic lights detection and distance estimation using color features. In *ITS*, 2012.

[35] O. Dictionaries. "context". Oxford Dictionaries. http://www.oxforddictionaries.com/definition/english/context, January 2016.

[36] A. Ess, T. Mueller, H. Grabner, and L. V. Gool. Segmentation-based urban traffic scene understanding. In *BMVC*, 2009.

[37] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) challenge. *IJCV*, 2010.

[38] N. Fairfield and C. Urmson. Traffic light mapping and detection. In *ICRA*, 2011.

[39] B. Fan, W. Lin, and X. Yang. An efficient framework for recognizing traffic lights in night traffic images. In *CISP*, 2012.

[40] P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with convolutional neural networks: a comparison to SIFT. In *CoRR*, 2014.

[41] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 1956.

[42] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich. Making Bertha see. In *ICCV Workshop on Computer Vision for Autonomous Driving*, 2013.

[43] B. Frey and D. MacKay. A revolution: Belief propagation in graphs with cycles. In *In Neural Information Processing Systems*, 1997.

[44] B. Fröhlich, E. Rodner, and J. Denzler. As time goes by - anytime semantic segmentation with iterative context forests. In *DAGM*, 2012.

[45] B. Fröhlich, E. Rodner, and J. Denzler. Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In *ACCV*, 2012.

[46] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 2013.

[47] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[49] A. E. Gomez, F. A. R. Alencar, P. V. Prado, F. S. Osorio, and D. F. Wolf. Traffic lights detection and state estimation using hidden markov models. In *IV*, 2014.

[50] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, and H. Chen. The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles. In *IV*, 2010.

[51] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009.

[52] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 1989.

[53] S. Gupta, R. Girshick, P. Arbelàez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014.

[54] V. Haltakov, H. Belzner, and S. Ilic. Scene understanding from a moving camera for object detection and free space estimation. In *IV*, 2012.

[55] V. Haltakov, J. Mayr, C. Unger, and S. Ilic. Semantic segmentation based traffic light detection at day and at night. In *GCPR*, 2015.

[56] V. Haltakov, C. Unger, and S. Ilic. Framework for generation of synthetic ground truth data for driver assistance applications. In *GCPR*, 2013.

[57] V. Haltakov, C. Unger, and S. Ilic. Geodesic pixel neighborhoods for multi-class image segmentation. In *BMVC*, 2014.

[58] V. Haltakov, C. Unger, and S. Ilic. Geodesic pixel neighborhoods for 2D and 3D scene understanding. *CVIU*, 2016.

[59] B. Hariharan, P. Arbelàez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[60] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[61] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[62] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[63] Y. Hel-Or and H. Hel-Or. Real time pattern matching using projection kernels. *ICCV*, 2003.

[64] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005.

[65] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. *Wavelets, Time-Frequency Methods and Phase Space*, 1989.

[66] F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *ICCV*, 2007.

[67] Insurance Institute for Highway Safety (IIHS). Status Report, Vol. 42, No. 1. Rep. IIHS, 2007.

[68] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fields - an efficient, non-parametric approach to image labeling problems. In *CVPR*, 2012.

[69] C. Jang, C. Kim, D. Kim, M. Lee, and M. Sunwoo. Multiple exposure images based traffic light recognition. In *IV*, 2014.

[70] M. B. Jensen, M. P. Philipsen, A. Møgelmose, and T. B. Moeslund. Vision for looking at traffic lights: Issues, survey, and perspectives. In *ITS*, 2016.

[71] S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. In *PAMI*, 2013.

[72] H.-K. Kim, Y.-N. Shin, S. Kuk, J. H. Park, and H.-Y. Jung. Night-time traffic light detection based on SVM with geometric moment features. In *World Academy of Science, Engineering and Technology*, 2013.

[73] Y. K. Kim, K. W. Kim, and X. Yang. Real time traffic light recognition system for color vision deficiencies. In *ICMA*, 2007.

[74] P. Kohli, L. Ladický, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. In *IJCV*, 2009.

[75] V. Kolmogorov and M. Wainwright. On the optimality of tree-reweighted max-product message passing. In *21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[76] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. GeoF: Geodesic forests for learning coupled predictors. In *CVPR*, 2013.

[77] F. Korč and W. Förstner. eTRIMS Image Database for interpreting images of man-made scenes. Technical report, University of Bonn, 2009.

[78] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2011.

[79] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[80] S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *NIPS*, 2003.

[81] S. Kumar and M. Hebert. A hierarchical field framework for unified context-based classification. In *ICCV*, 2005.

[82] L. Ladický, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical CRFs for object class image segmentation. In *ICCV*, 2009.

[83] L. Ladický, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical random fields. In *PAMI*, 2013.

[84] L. Ladický, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, 2014.

[85] L. Ladický, P. Sturgess, K. Alahari, C. Russell, and P. H. S. Torr. What, where and how many? Combining object detectors and CRFs. In *ECCV*, 2010.

[86] L. Ladický, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. H. S. Torr. Joint optimisation for object class segmentation and dense stereo reconstruction. In *BMVC*, 2010.

[87] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[88] J. Li. An efficient night traffic light recognition method. In *Journal of Information & Computational Science*, 2013.

[89] S. Li and A. Chan. 3D human pose estimation from monocular images with deep convolutional neural network. In *ACCV*, 2014.

[90] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.

[91] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[92] F. Lindner, U. Kressel, and S. Kaelberer. Robust recognition of traffic signals. In *IV*, 2004.

[93] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 1989.

[94] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. In *CoRR*, 2015.

[95] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional models for semantic segmentation. In *CVPR*, 2015.

[96] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi. Entangled decision forests and their application for semantic segmentation of CT images. In *Information Processing in Medical Imaging*, 2011.

[97] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[98] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *ECCV*, 2010.

[99] D. Munoz, J. A. Bagnell, and M. Hebert. Co-inference for multi-modal scene analysis. In *ECCV*, 2012.

[100] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 1980.

[101] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[102] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *ICCV*, 2011.

[103] M. Omachi and S. Omachi. Traffic light detection with color and edge information. In *ICCSIT*, 2009.

[104] S. Papert. The summer vision project. http://hdl.handle.net/1721.1/6125, June 1966.

[105] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[106] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters – improve semantic segmentation by global convolutional network. In *CVPR*, 2017.

[107] D. Pfeiffer and U. Franke. Towards a global optimal multi-layer stixel representation of dense 3D data. In *BMVC*, 2011.

[108] M. P. Philipsen, M. B. Jensen, A. Møgelmose, and T. B. Moeslund. Traffic light detection: A learning algorithm and evaluations on challenging dataset. In *ITS*, 2015.

[109] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017.

[110] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[111] J. Redmon and A. Farhadi. YOLO9000: Better, faster, stronger. In *CVPR*, 2017.

[112] ReelSEO.com. 500 hours of video uploaded to YouTube every minute. http://www.reelseo.com/hours-minute-uploaded-youtube/, November 2015.

[113] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.

[114] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[115] S. Ross, D. Munoz, M. Hebert, and J. A. Bagnell. Learning message-passing inference machines for structured prediction. In *CVPR*, 2011.

[116] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

[117] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *ICRA*, 2009.

[118] S. Saini, S. Nikhil, K. R. Konda, H. S. Bharadwaj, and N. Ganeshan. An efficient vision-based traffic light detection and state recognition for autonomous vehicles. In *IV*, 2017.

[119] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth. Efficient multi-cue scene segmentation. In *GCPR*, 2013.

[120] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth. Stixmantics: A medium-level model for real-time semantic scene understanding. In *ECCV*, 2014.

[121] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[122] R. Shapovalov, D. Vetrov, and P. Kohli. Spatial inference machines. In *CVPR*, 2013.

[123] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional models for semantic segmentation. *PAMI*, 2016.

[124] Y. Shen, U. Ozguner, K. Redmill, and J. Liu. A robust video based traffic light detection algorithm for intelligent vehicles. In *IV*, 2009.

[125] M. J. Shensa. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on Signal Processing*, 1992.

[126] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. In *PAMI*, 2012.

[127] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[128] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[129] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. IJCV, 2007.

[130] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *ICCV - Workshop on 3D Representation and Recognition*, 2011.

[131] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[132] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[133] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[134] G. Siogkas, E. Skodras, and E. Dermatas. Traffic lights detection in adverse conditions using color, symmetry and spatiotemporal information. In *VISAPP*, 2012.

[135] P. Sturgess, K. Alahari, L. Ladický, and P. H. S. Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC*, 2009.

[136] C. Sutton and A. McCallum. Piecewise training for undirected models. In *UAI*, 2005.

[137] C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *International Conference on Machine Learning*, 2007.

[138] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[139] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *ECCV*, 2008.

[140] H. Tae-Hyun, J. In-Hak, and C. Seong-Ik. Detection of traffic lights for vision-based car navigation system. In *Advances in Image and Video Technology*, 2006.

[141] P. J. Toivanen. New geodesic distance transforms for gray-scale images. In *Pattern Recognition Letters 17*, 1996.

[142] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. In *PAMI*, 2007.

[143] Digital Trends. Facebook reveals we upload a whopping 350 million photos to the network daily. http://www.digitaltrends.com/social-media/according-to-facebook-there-are-350-million-photos-uploaded-on-the-social-network-daily-and-thats-just-crazy/, September 2013.

[144] Z. Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008.

[145] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3D brain image segmentation. In *PAMI*, 2010.

[146] C. Unger, E. Wahl, P. Sturm, and S. Ilic. Stereo fusion from multiple viewpoints. In *DAGM*, 2012.

[147] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn. Differences between stereo and motion behaviour on synthetic and real-world stereo sequences. In *IVCNZ*, 2008.

[148] P. Viola and M. Jones. Robust real-time object detection. In *Second International Workshop On Statistical And Computational Theories Of Vision - Modeling, Learning, Computing, And Sampling*, 2001.

[149] C. Wang, T. Jin, M. Yang, and B. Wang. Robust and real-time traffic lights recognition in complex urban environments. In *International Journal of Computational Intelligence Systems*, 2011.

[150] M. Weber, P. Wolf, and J. M. Zöllner. DeepTLR: A single deep convolutional network for detection and classification of traffic lights. In *IV*, 2016.

[151] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers. Stereoscopic scene flow computation for 3D motion understanding. *IJCV*, 2011.

[152] C. Wojek and B. Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *ECCV*, 2008.

[153] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. Lessons and insights from creating a synthetic optical flow benchmark. In *ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation*, 2012.

[154] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN database: Large scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[155] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, 2014.

[156] L. Yatziv, A. Bartesaghi, and G. Sapiro. O(N) implementation of the fast marching algorithm. In *Journal of Computational Physics 212*, 2006.

[157] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[158] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based R-CNNs for fine-grained category detection. In *ECCV*, 2014.

[159] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[160] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.

[161] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014.

[162] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

[163] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.