2017

Extracting Parking Areas from Remote Sensing Imagery and Spatiotemporal Traffic Data

Julian Glaab
University of Rhode Island, glaab.julian@gmail.com

EXTRACTING PARKING AREAS FROM REMOTE SENSING IMAGERY AND SPATIOTEMPORAL TRAFFIC DATA

BY

JULIAN GLAAB

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

INDUSTRIAL AND SYSTEMS ENGINEERING

UNIVERSITY OF RHODE ISLAND

2017

MASTER OF SCIENCE THESIS

OF

JULIAN GLAAB

APPROVED:

Thesis Committee:

Major Professor Jyh-Hone Wang

Christopher Hunter

Valerie Maier-Speredelozzi

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2017

ABSTRACT

The increasing world-wide demand for mobility caused urban parking space to become a scarce resource. Searching for parking is not only annoying to drivers, it also represents a serious cause of urban traffic. A previous study found that vehicles searching for parking contribute as much as 30 percent to the total traffic [1]. To solve this problem, a comprehensive Intelligent Transportation System (ITS) is necessary to supply drivers directly with information about open parking spaces in their surrounding area, so they do not need to "go and look" themselves.

However, in order to create an ITS for urban parking, a detailed map with accurate parking area positions and additional parking information, like cost or parking restrictions, is required. Currently, no database provides this information above the level of individual municipalities. For mapping parking areas fast, cost efficient and reliably, a highly scalable process is required, which only relies on existing and disseminated technologies instead of exhaustive manual data collection methods. Remote sensing imagery and vehicle trajectories from probe vehicles, so-called "floating car data" (FCD) were identified as an appropriate data source, as these sources provide all necessary information and are available for most urbanized regions in standardized form.

This study uses machine learning techniques to extract and aggregate parking area information from remote sensing imagery and floating car data. The multistage process for detecting parking area positions on images is based on GIS (Geographic Information System) data, a convolutional neural network for detection of parking vehicles and a recursive algorithm to derive parking areas from the position of individual parking vehicles.

Subsequently, metadata about cost, user group restrictions and opening hours is added to every detected parking area. Metadata is obtained by analyzing char-

acteristic temporal patterns in local parking behavior, which are retrieved from floating car data. In order to sense these patterns, machine learning classifiers are applied to a set of different features. To achieve higher accuracy, FCD data is enriched by context data, like demographics or points of interest.

Results are validated using OpenStreetMap (OSM) data and through manually collected reference data in a test area in the city of Braunschweig in Germany. The developed process for parking area detection is robust and achieved a detection accuracy above 95 percent with respect to parking area capacity in fully exposed image areas. However, the process is not able to sense parking areas, which are hidden by objects like roofs or trees. The idea of metadata extraction from floating car data is very promising, as the average classification accuracy ranges already above 80 percent in the used training data set. The remaining error is mainly due to the fact that the amount of used training data currently does not allow a spatial classifier resolution high enough to capture parking restrictions of small individual parking areas.

In future work it is planned to use more floating car data to improve classification accuracy further and to extract even more parking information from this data source. Parking area mapping based on remote sensing imagery may be improved by using a specialized convolutional network architecture for vehicle detection. Additionally, images from autumn and winter season, where more ground is exposed to the classifier, could be used in order to increase overall detection accuracy.

ACKNOWLEDGMENTS

I would first and foremost like to thank my academic advisor Dr. Jyh-Hone Wang, Professor for Advanced Statistical Methods from the Department of Mechanical, Industrial and Systems Engineering at the University of Rhode Island (URI). He consistently allowed this thesis to be my own work, but always supported me regarding my research and writing.

I would also like to thank Dr. Christopher Hunter, Professor for Civil and Environmental Engineering and Dr. Valerie Maier-Speredelozzi, Associate Professor at the Department of Industrial and Systems Engineering at URI. As members of my thesis committee they both provided me with valuable input throughout this study.

Furthermore, I would like to thank the German Academic Foundation (Studienstiftung des Deutschen Volkes) for the gracious financial support of my research. Without their support, I would not have been able to obtain the data required to conduct this study.

I would also like to acknowledge the Open Data Science Conference (ODSC) for granting me their scholarship to attend the ODSC East conference in Boston in May 2017. I was able to gain a lot of new inspiration at ODSC pushing my research even further. In this context, I would also like to thank the Graduate Student Association, the International Engineering Program and the Graduate School at URI for their additional financial support in this matter.

This thesis is part of the AIPARK project. The Federal Ministry for Economic Affairs and Energy of Germany awarded AIPARK with admission to the EXIST program for university-based entrepreneurship, which provides generous funding for this project. I am very thankful for this recognition.

Finally, I must express my very profound gratitude to my parents and to my

girlfriend for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Julian Glaab

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Rationale

With the increasing demand for individual mobility worldwide, parking space has become a scarce resource in urban areas and represents a serious problem today. While the total traffic volume is continuously increasing in many places, local governments often can not catch up with extending infrastructure [2]. Nadereh et al. found that drivers in urban areas often spend more than 20 minutes each time they are trying to park their car. Hence, it is estimated that vehicles searching for parking contribute as much as 30 percent to the total traffic [1]. Parking-caused traffic not only consumes time, it is also a source of substantial amounts of greenhouse gas emissions and extensive air pollution and therefore creates a potential health risk for urban citizens [3].

The general issue with parking-caused traffic is the asymmetric distribution of information among drivers and infrastructure. If all motorists are only aware of the parking occupancy along their individual trajectory they are missing information at a more global scale, which is necessary to make efficient decisions when navigating [4]. To overcome the problem of asymmetric information distribution among vehicles, several solutions have been developed in the past in order to mitigate the parking problem. These approaches may be categorized in stationary and crowd-based systems.

Stationary systems Stationary parking guidance systems are most commonly used for off-street parking areas like car parks. These systems consist of two parts, a mechanism for counting the number of occupied parking spots and some kind of visualization to provide drivers with information about the current occupancy.

Measuring the number of open parking spots is usually achieved by cameras, ultrasonic or radar sensors or ticketing systems. Occupancy is often visualized through roadside adjustable traffic signs [5]. More advanced stationary systems also stream their information to a server, making it accessible for third-party services like mobile apps [6]. While quite accurate, these systems are expensive to install and maintain and naturally only cover a very limited share of the total parking spots in an area, as they typically are operated by property owners only.

Crowd-based systems More recent parking solutions are based on the idea of **crowdsensing**. Instead of retrieving data from stationary sensors, moving vehicles are used as a source of information. For instance, Mathur et al. have successfully tested ultrasonic sensors on a fleet of taxicabs to measure on-street parking occupancy. By analyzing the sensor's distance readings and mapping them to road sections they achieved an average occupancy estimation accuracy above 90 percent [7].

Two other research groups, Nawaz et al. and Salpietro et al. tried to use smartphones to track parking events and notify other users about them. In all crowd-based systems, the collected information is either processed and re-distributed by a central server or directly shared via vehicular ad-hoc-networks [4, 6]. All mentioned crowd-based approaches show promising results, as they are able to cover much larger areas at lower cost compared to stationary systems. However, sensor-based solutions require high levels of technological dissemination of the used hardware, e.g. ultrasonic sensors and communication technology in vehicles [8]. Although previously introduced smartphone based crowdsensing systems avoid these issues they require strong user participation, which these systems were unable to reach. Users needed to interact with other users and inform them about open parking spaces. Due to the strong driver involvement and the fact that neither approaches

reached a critical mass of users, they failed to achieve acceptable levels of accuracy.

It can be concluded that none of the existing solutions is able to address the problem of asymmetric parking information sufficiently. To resolve this issue, one would need to supply drivers with global information about currently available parking spaces without relying on expensive hardware or a large base of users. Additionally, any parking guidance system that works at global scale requires a consistent and comprehensive database with detailed information about on- and off-street parking spaces as a foundation.

However, this can become challenging, as it is difficult to obtain high quality parking area information at larger scales [9]. Public parking spaces are most commonly regulated by local municipalities, which usually do not keep a record of parking area locations and restrictions.

This thesis is part of the development of AIPARK, a new kind of parking guidance system. The study strives to provide a new method for mapping parking spaces and corresponding metainformation by extensively mining floating car data (FCD) and remote sensing imagery.

1.2 AIPARK system

A group of graduate students in the fields of Computer Science and Industrial Engineering at TU Braunschweig in Germany is working on a parking guidance system. The developed system is called **AIPARK**, which is short for "Artificial-Intelligence-Based-Parking". When finished, AIPARK will be able to guide drivers on a straight route to an open parking spot near their destination. In order to do so, AIPARK consistently analyzes very large amounts of traffic data with multiple artificial neural networks (ANN) that are able to draw conclusions about the parking situation at any requested location in German cities. Drivers will receive parking directions at high accuracy directly inside their GPS system or mobile

navigation application.

The theory behind AIPARK is based on findings from several research groups, which prove that human mobility behavior is predictable to a large extents. In 2010, Song et al. analyzed mobility patterns from 50,000 individuals over three months. They showed that most people follow very regulated, repeating mobility patterns. Considering the entropy of each individuals' trajectory, they found the potential predictability of user mobility at an average value of 93 percent. Song et al. obtained this value by applying the concept of "Fano's inequality", which is a theorem in information theory stating the lower bound on the error probability in any information signal. Thus, a 93 percent level of user mobility predictability can be seen as the theoretical maximum any model will be subject to [10]. In 2008, Gonzalez et al. published their article "Understanding individual human mobility patterns" in Nature Journal. By investigating trajectories from 100,000 individuals over six months, they found that human mobility is characterized by very high spatial and temporal regularity [11]. Obviously, road traffic is determined by human mobility patterns. Therefore, it may deduced that the rules governing human mobility also apply to parking.

The architecture of AIPARK is designed to sense these patterns and derive parking occupancy in spatial and temporal dimensions from them. The architecture is shown in Figure 1 and is split into input data, internal processing within the AIPARK system and user front-end. This design is based on the DIN CEN ISO/TS standard 18234-7, which provides a reference data and communication scheme for the delivery of parking information to a variety of receivers [12].

The gray box in the upper left corner outlines the role of this study within the overall system: A process for extracting information increments from remote sensing imagery, road data from GIS (geographic information systems) and floating

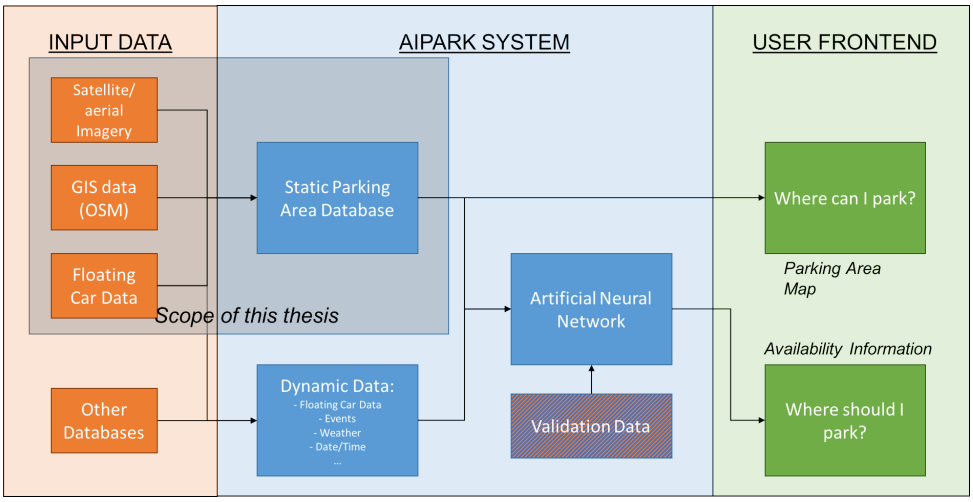


Figure 1. AIPARK System

car data are merged and integrated into one single parking area database.

1.3 Objective and limitation of study

Objective This master's thesis will contribute to the AIPARK project by providing a highly scalable method for mapping parking areas at a (inter-)national scale and storing them in a single standardized database. The objective of this study is to develop this process by solely relying on widespread technological standards to ensure its high scalability. All results are going to be tested and compared against real-world-data within the city of Braunschweig, Germany.

Limitations The limitations of this study may be split into three major dimensions: Data quality and coverage, capabilities of state of the art techniques in computer vision and data science and lastly, limited hardware capabilities.

As in every data analysis problem, output quality is determined to a very large extent by the type of input to the model. If the input data does not suit the problem in terms of content or quality requirements, the model output will be insufficient. Also, the proposed solution builds on a multistage process, which is composed of several different cutting edge techniques in computer vision. Their technological

limitations also directly influence the accuracy of parking area detection. The entire system can not perform better than the worst of its components.

Lastly, hardware capabilities also present a notable limitation to this study, due to the high computational cost of computer vision algorithms. For computation, a setup consisting of an Intel quad-core i5 processor (CPU) at 2.9 Giga-Hertz, 8 gigabyte (Gb) RAM and an NVIDIA GeForce GT 640 graphics card (GPU) equipped with 1 Gb memory is used. Although the hardware does not directly influence the quality of this study's results, it has strong impact on performance. Especially when extracting complex features from large datasets, system memory becomes a bottleneck during data preparation and classification tasks. Since the parking area detection algorithm partially relies on state of the art convolutional neural networks (CNNs), the comparatively low graphics memory also slows down training and classification phases quite strongly. CNNs for object detection can only efficiently be trained on GPUs, as they handle matrix operations much faster than regular cores.

CHAPTER 2

Literature review

In order to develop a process for mapping parking areas and their metadata, the current state of the art in all related fields of technology is aggregated. For this purpose, the main literature regarding parking area databases in general, floating car data fundamentals and object detection methods focusing on remote sensing imagery will be reviewed.

2.1 Existing parking area databases and surveying methods

The need for parking area databases and highly accurate traffic information in general arose fairly recently with the technological dissemination of the mobile internet. Before, parking space data was mostly collected by surveyors and managed by municipal administrations for the purpose of city planning [13, 14]. This resulted in a highly decentralized allocation of information with great differences in terms of data quality and structure [15].

OpenStreetMap The development of actual databases containing parking area data started with the public introduction of digital maps and geographic information systems (GIS). Today, OpenStreetMap is the most comprehensive public digital map and the only GIS that contains considerable amounts of parking area information. Founded in 2004 by British entrepreneur Steve Coast at University College London, OpenStreetMap has grown a remarkable user base of more than three million registered members [16]. The idea behind OpenStreetMap is called “Crowdsourcing” and describes the collaborative and voluntary collection of geographic information. OpenStreetMap provides a data format (“OSM-XML”), consisting of standardized building elements. Different graphical user-interfaces

(GUIs), such as websites or mobile applications provide a convenient way for users to utilize these elements to enrich OpenStreetMap with further geographic information. The goal of OpenStreetMap is to create and provide an open and editable map of the world [14].

Among a vast variety of other geographic information, OpenStreetMap also includes data about parking areas, including location, geometric shape and - at varying degree of completeness - meta information. OpenStreetMap may hold meta information such as parking area capacity, cost, opening hour restrictions or user group restrictions. An analysis conducted within the scope of this study resulted in a total count of about 300,000 parking area entries within Germany and more than one million entries world wide. About 50,000 parking areas additionally include some kind of meta information [17].

Commercial solutions Besides OpenStreetMap as an open source project, there is also a small number of commercial parking data providers. These companies can be categorized in two groups depending on their method of data acquisition. Companies like Inrix, Streetline or ParkingHQ merge parking information from multiple sources like parking garage operators, municipal traffic information systems or physical parking sensors for real-time data into one single database [18, 19, 20]. While the quality of data provided by operators themselves is in general very high, the process of combining the vast number of information providers into one platform is very time-consuming. Another issue with this approach is caused by the fact that these platforms only include data from managed parking areas, but not from on-street parking spots or free off-street parking areas, which represent with the largest share of urban parking infrastructure, at more the 85 percent.[2].

The second group of commercial data providers is aware of this issue and tries

to address it with a different kind of data acquisition: Companies like Parkopedia are relying on data contributed by their own user base, which makes the system a proprietary form of crowd sourcing [21]. Users of Parkopedia's mobile apps can add new information about parking area positions, opening hours, restrictions and prices to the database. Hence, the data collection method of Parkopedia is also capable of obtaining data for free on- or off-street parking areas. However, data quantity, actuality and spatial coverage are strongly dependent on the size of Parkopedia's user base. This "chicken or the egg" dilemma, where the product quality depends on the user base and the user base is attracted by product quality makes it difficult to scale up the data acquisition system of Parkopedia to new cities or countries. In order to succeed with such a model and to reach the required critical mass, typically strong marketing efforts or additional product features are necessary to attract new users [22]. In conclusion, it can be stated that no for-profit organization has found a promising comprehensive and scalable approach for acquiring parking data.

2.2 Floating car data

Floating car data (FCD), also referred to as "probe data", is a term that is now used for a very broad range of traffic data acquisition types. This section presents and compares the different sources of floating car data, which are available today. Furthermore, FCD data representation approaches are introduced.

2.2.1 Sources of floating car data

FCD may be obtained from various different sources. Table 2.2.1 provides a summary of all FCD types. Principal literature, such as Lorkowski et al. often distinguishes between passive and active floating car data [23]. Passive FCD is generated by tracking single vehicles from stationary points along the road network.

Vehicle identification is either achieved through automatic number plate recognition (ANPR) or by passive onboard-transponders that respond to the signal of a stationary measurement facility. Passive FCD generates very accurate traffic data at rather limited geographic scope. Because the required equipment must withstand severe weather conditions and obtaining approval for installation can be a lengthy process, passive FCD is a very expensive method of FCD generation.

Active FCD is characterized by each individual's ability to collect data and send it to a central control station for analysis. The three main sources of active FCD are celluar networks ("Floating Phone Data"), fleets of probe vehicles and smartphone data. Any type of active FCD extraction needs a wireless communication system and positioning unit. Position data is frequently sent to a central server where it gets mapped onto the road network [23].

One of the most commonly known methods is "Floating cellular data". This FCD approach utilizes cellular networks like the Global System for Mobile Communication (GSM) and triangulation for locating a driver's mobile phone [24]. Although it has low geographic accuracy of between 100 meters to 35 kilometers, cellular network FCD is very well developed and has many applications in real-time traffic information systems today.

Fleets of probe vehicles were the first kind of active FCD that has been described in the literature [25]. Many different research projects equipped taxi fleets with GPS receivers and some kind of communication unit to turn them into probe vehicles. More recent projects even utilized the many assistance systems in modern cars (e.g. ultrasonic sensors) for measuring traffic density in their surrounding. This approach is called Extended Floating Car Data (XFCD) [26]. Car manufacturers like BMW or Daimler also turned parts of their fleets into probe vehicles [27]. Fleets of probe vehicles are able to generate traffic data in much larger ar-

eas than passive FCD. However, the cost for equipping thousands of vehicles with the necessary hardware is expensive. Spatiotemporal data coverage represents another problem: Real-time detection of traffic congestion requires high density or frequency of probe vehicles, which cannot be guaranteed at any time. Chen et al. found that probe vehicles must contribute as much as one percent to the total traffic in order to generate FCD at the same quality as passive FCD extraction methods [8]. Especially in urban areas with less traffic volume, like residential zones, data from vehicle fleets is therefore often less significant [23].

With the spread of smartphone technology, a new and more accurate source of FCD was provided. Since modern smartphones are equipped with both GPS receivers and GSM modules, they are able to stream their position steadily to a remote server. Interestingly, only very few papers like Krieg et al. or Salpietro et al. are considering smartphones as a source of floating car data [28, 4]. The reason for this is grounded on the fact that smartphone FCD is mostly kept undisclosed by mobile app developers and hardware manufacturers (OEMs). Google Maps is a prominent example of a company that collects and uses smartphone FCD in their own mobile application, but refuses publication of its data due to privacy and strategic reasons [29].

Nevertheless, when available, smartphone FCD is a valuable source of information for traffic analysis applications at an entirely new level of accuracy and spatiotemporal coverage. Table **??** provides a summary of all presented FCD acquisition approaches.

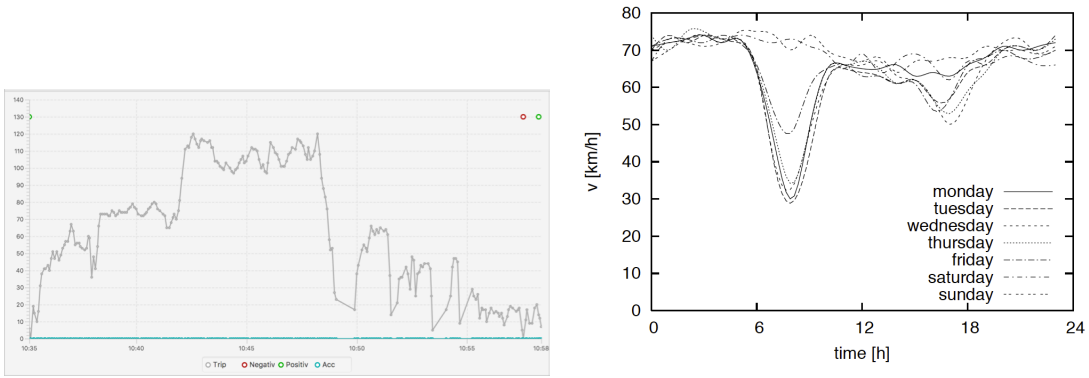
2.2.2 Data representations

In many applications, FCD is used to determine travel speed in certain road sections or - at larger scale - in entire road networks. Figure 2 illustrates the usual appearance of FCD. The left image shows the “raw” speed profile of the probe

Data source	Accuracy	Advantages	Disadvantages
Cellular networks	100 m to 35 km	Mature technology Large number of probes	Low accuracy
Probe vehicle fleets	1-15 m (GPS)	High accuracy	High cost scattered coverage
Stationary traffic sensors	< 1 m	High accuracy	High cost Measurements only for selected spots
Smartphone data	1-15 m (GPS)	High accuracy High coverage Mature technology	Restricted access

Table 1. FCD sources

vehicle. In this case, data is obtained by smartphone FCD. The right image by Lorkowski et al. depicts the travel speed at a certain location as an aggregated form of FCD.



(a) Vehicle speed profile from smartphone FCD (b) Diurnal course of speed on Cannstatter Strasse, Stuttgart, by Taxi-FCD [23]

Figure 2. Common FCD representations

Another FCD representation, which is particularly useful to this study are "parking events". By analyzing the speed profile in relation of the geographic trajectory of a smartphone, it is possible to find out when and where drivers parked or left their parking space. Parking events have already been described and used by Salpietro et al. and Nawaz et al. in their attempts to create a real-time parking vacancy map based on crowd-sourced smartphone sensor data [4, 6]. But the distinction in "successful" and "leaving" parking events allows to extract much more information about temporal parking attractiveness of urban

areas. A research group at Chinese Wuhan university has already been able to localize - although still inaccurately - parking areas solely based on the spatial distribution of parking events using the DBSCAN clustering algorithm [30]. This study is attempting to go one step further and extract additional information about cost, parking restrictions and opening hours from spatiotemporal parking event distributions.

2.3 Image classification and object detection

The science of computer vision presents two major challenges to researchers: Image category classification and object detection. The classification task strives to determine which object class from a finite list of given classes is shown in an image. Object detection goes one step further and also tries to locate objects within the image. Figure 3 illustrates the desired output of these two problems. Object detectors mostly consist of two main components, an algorithm for generating region proposals and a classifier whose task is to describe the content of each region proposal. Thus, to locate objects in an image, multiple classifications are performed on different image regions. Since this study is partly based on vehicle detection within remotely sensed images, computer vision fundamentals as well as state of the art object detection methods are introduced in this section. Furthermore, a summary of the previous work on vehicle detection in aerial images is provided.

2.3.1 Object detection in images

From an mathematical perspective, a digital two-dimensional image $I(x, y)$ can be expressed as a 2D array of intensity samples at limited precision for each sample, according to Shapira et al. [31]. Following this definition, mathematical operations on images can be defined very straight-forward, since the image is now nothing but a matrix. Each matrix element is called a “pixel”, which is the smallest

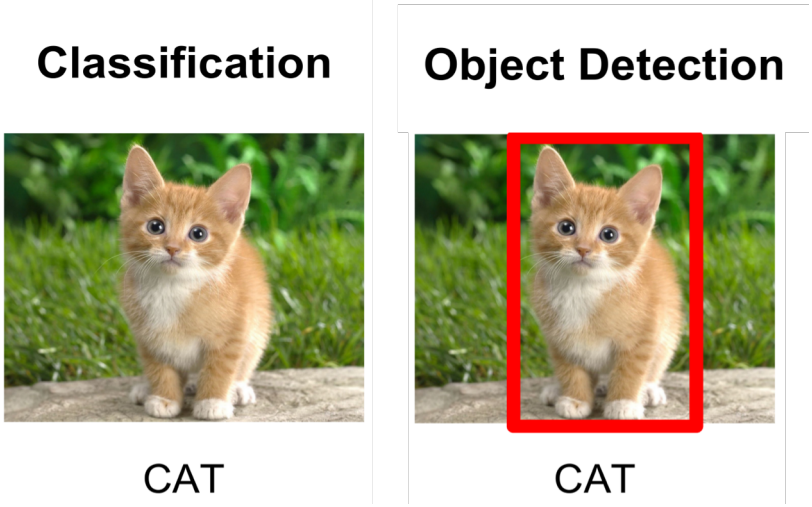


Figure 3. Classification and detection: general problems of computer vision

increment for storing intensity values.

If an image only has pixel intensity values of either 0 or 1, the image is called a "binary image". Gray scale images are described by having only one single intensity value per pixel within a $[0, 1]$ range.

A multispectral image $M(x, y)$ has a vector at each pixel. If the image is a colored image in RGB-colorspace, the vector has three elements, where each specifies an intensity value for red, green and blue. This study mainly works with gray scale and color images. Binary representation is going to be used for masking certain image parts. Figure 4 illustrates the standard coordinate system, which is used in image processing to describe positions, areas or to access individual pixels or pixel subsets.

With a common understanding of images established, methods of object detection (OD) can now be explained. Computer vision distinguishes between three major OD techniques: Mathematical morphology and segmentation, deformable part models and convolutional neural networks.

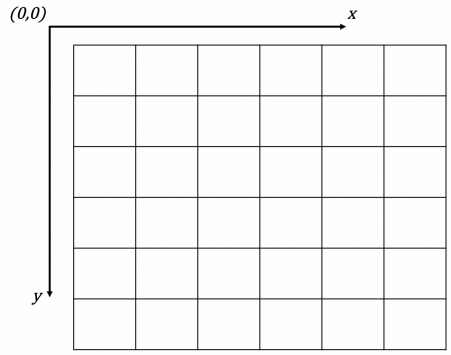


Figure 4. Standard coordinate system for image processing and single pixel access

Mathematical morphology in image processing

Mathematical morphology was the first approach to the processing of digital images, mainly developed during the 1980s. Major contributions were made by French mathematicians Matheron and Serra, who worked on the fundamentals of mathematical morphology since 1964 [32]. Morphology in image processing became popular with the increasing computational capabilities of processors. The method can be defined as a shape-based approach for analysis of structures in images. It simplifies images by preserving main shape characteristics and eliminating irrelevant image contents [33]. If the method is used for dividing images into clusters with similar content, it is called "image segmentation". Mathematical morphology in its standard form is performed on binarized images. With binary morphology, object detection is accomplished by a standard workflow: First the image is binarized, then a combination of four standard operations is performed on the pre-structured binary image. The four morphological standard operators dilation, erosion, opening and closing are explained in the next sections.

Binarization Images need to be converted into grayscale images and then be binarized before mathematical morphology can be applied on them. A pixel may have a finite intensity value between $[0, 1]$ in a gray-scale image. During bina-

rization, every pixel is assigned a value of either 0 or 1, depending on a given threshold, which separates these two classes. Many different methods for thresholding have been developed, which either compute one threshold for the entire image or adaptively consider changing levels of illumination in different image regions [34]. The most simplistic form of binarization is setting the value of all pixels with intensity ≤ 0.5 to zero and accordingly set the value of all remaining pixels to one. Depending on the thresholding method, binarization may produce significantly different results, which affect subsequent morphological operations. Figure 5 provides results of two different thresholding methods, the global "Otsu" method and an approach using adaptive thresholds.

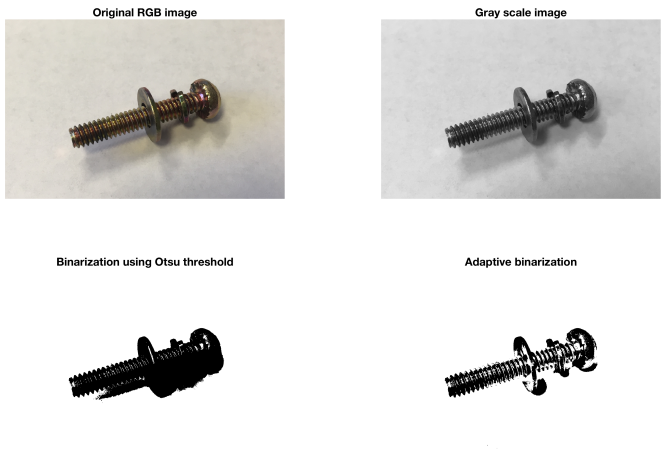


Figure 5. Image of a screw after applying different binarization methods

Structuring element A structuring element (SE) is a shape used to probe the input image. It is expressed as a matrix that defines a neighborhood to certain pixels surrounding the pixel in the image. The pixel and its neighbors are treated as one unit, when applying morphological operators. Structuring elements may have different shapes, like circles, disks, rectangles or lines. Usually, the structuring elements type and size are chosen according to the geometry of the object which

needs to be processed [35]. The mathematical definition of structuring elements helps to illustrate the concept: If a digital image I is a subset of a discrete 2D-space $I \subset \mathbb{Z}^2$, then a 3x3 structuring element $S_1 \subset \mathbb{Z}^2$ is defined around the pixel of origin $\otimes$ [36]:

$$S_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \otimes & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1}$$

and a 5x5 diamond shaped structured element S_2 as:

$$S_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & \otimes & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2}$$

While in S_1 only directly surrounding pixels are treated as neighbors, the structuring element S_2 considers additional pixels to be neighbors due to its diamond-like form. As pointed out in section 2.3.3, structuring elements were among the first techniques used for vehicle detection in remote sensing images.

Morphological operators Mathematical morphology uses four operators, erosion, dilation, opening and closing. By transforming certain subsets of a binary image, these operators are able to separate objects from the background.

Erosion of an image I by using structuring element S is written as $I \ominus S$ and removes pixel structures at an objects' boundaries. **Dilation** $I \oplus S$ is the dual operation of erosion. It has the effect of "thickening" or merging object parts.

$$I^c \ominus S = (I \oplus S)^c \tag{3}$$

I^c is the complement of I.

The **opening** operation is performed by eroding I followed by dilating it using the same SE. The operation is defined as:

$$I \circ S = (I \ominus S) \oplus S \tag{4}$$

Opening an image serves the purpose of removing small object increments while maintaining the overall structure of the object. Lastly, the **closing** operation is defined vice versa as $I \bullet S$ due to the duality of erosion and dilation. Geometrically, closing helps smoothing rough object structures. Figure 6 provides examples for each of the four operators.

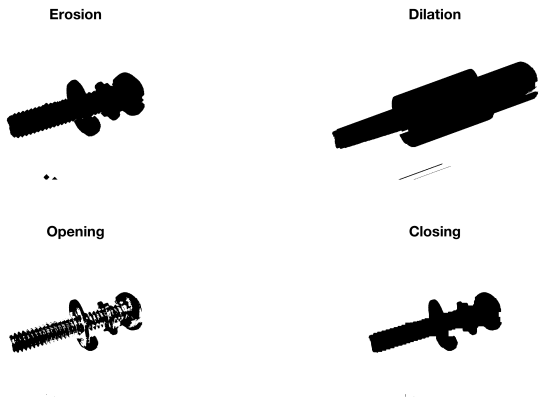


Figure 6. Morphological operations: erosion (diamond SE), dilation (line SE), opening (square SE), closing (square SE)

Applications With appropriate parametrization of binarization and suitable choice of structuring elements and morphological operators, objects may be separated quite well from image background. Mathematical morphology has many applications like industrial optical quality control, robotics or document processing.

However, this method also has several downsides: First of all, it only considers rough information about object size and shape but cannot incorporate any information about object color or structure. Secondly, mathematical morphology is size and rotation invariant, which means it is only capable of detecting objects at a given size and orientation of the corresponding structuring element. Any object at scales or orientations other than expected would be classified as background and

be removed [37].

Deformable Part Models

Deformable part models (DPM) are a more advanced approach in object detection and still state of the art in many applications. Unlike morphological methods, DPM works above the level of individual pixels and make use of more advanced image information such as structure and color. Felzenszwalb et al. from the department of computer science at Brown University first introduced a DPM based person detector in the year 2010 [38]. A deformable part is a geometric object whose shape can change among different instances. Deformable parts are a very helpful object representation, which is able to accommodate for different object views, rotations, scales or even appearances. As shown in figure 7, deformable part models divide objects in characteristic sub-components, whose relative position to each other may vary, depending on the object's appearance.

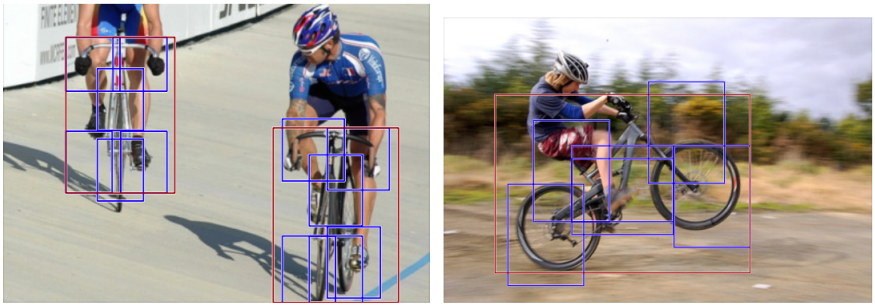


Figure 7. Images of bicycles illustrating the importance of deformations [38]

To implement DPM, three major components are required, a machine-readable image representation, also called a "feature map", a region proposal function and a classifier. The general object detection process with DPM is illustrated in figure 8 and begins with extracting a set of image descriptors or "features" from the image. Next, sub-regions of the image are defined using several possible techniques. Lastly, the corresponding feature set for each sub-region is used to classify the region's

content. This workflow has already been introduced in 2005 before DPM became popular: Dalal and Triggs described a process using HOG-features (Histogram of Oriented Gradients), a sliding window to specify sub-regions and a liner support vector machine classifier (SVM) [39]. Today, a variety of different methods is used along the detection process, which are briefly stated below.

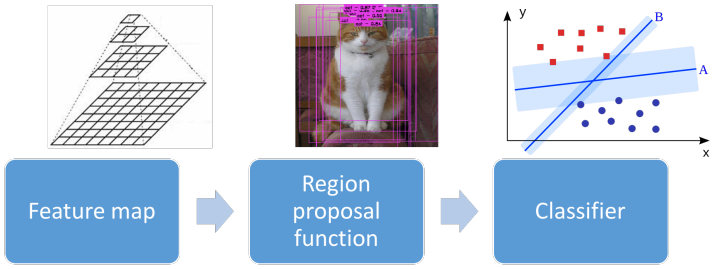


Figure 8. Generalized components for object detection

Feature maps According to Angelis, in data mining, a **feature** is an information increment retrieved from a larger data set, which is relevant for solving a classification task [40]. This definition is also valid for computer vision problems. Digital images in pixel representation as described in section 2.3.1 are optimized for visualization. However, pixel representation does not provide more abstract information about image content, which is necessary for any classification task. To overcome this issue, computer vision research has developed a variety of features, which describe the image with respect to different characteristics, like color, shape or texture. If a feature is computed for an entire 2D-image, the result is called **feature map**. In some implementations, the original image additionally is scaled or rotated and further feature maps are generated. In this case, the outcome is referred to as a **feature pyramid**, due to the pyramid-like shape, which results from stacking differently scaled feature maps on each other (see figure 9). An in-depth description of feature pyramids is given by [41].

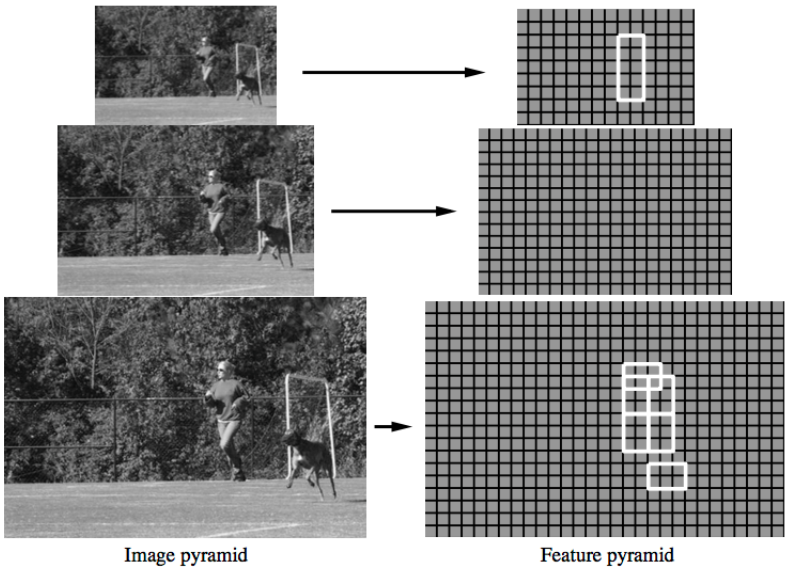


Figure 9. Feature pyramid [38]

Table 2 provides an overview of existing feature descriptors and which information they provide. The descriptors are compared with respect to associated computational costs and their entropy, which is a measure for "information density". Color histograms provide information about the intensity distribution in the different color channels of an image. An image in RGB-colorspace can therefore be characterized by three histograms for red, green and blue color channel [42]. Local binary patterns (LBP), Haar-like and SURF features (speeded up robust features) will not be discussed here, as they are only used in a very limited set of applications, mostly face recognition [43].

Special attention is paid to histogram of oriented gradients (HOG) features, as they are frequently used in traffic related applications (see section 2.3.2) and contain high entropy at low computational expenses. The way HOG features are computed is defined by Dalal and Triggs [39]:

1. Normalize image to reduce differences in illumination

Table 2. Feature descriptors

Feature descriptor	Information	Entropy	Computational cost
Color histograms	color	intermediate	low
Histogram of oriented Gradients (HOG)	shape	high	low
Local binary patterns (LBP)	texture	intermediate	high
Haar-like features	pixel intensity	low	low
Speeded Up Robust Features (SURF)	significant points	intermediate	intermediate

2. Compute intensity gradient at every pixel in x- and y-direction

$$I_x(x,y) = \frac{\partial}{\partial x}; I_y(x,y) = \frac{\partial}{\partial y} \tag{5}$$

$$\nabla I(x,y) = \begin{pmatrix} I_x(x,y) \\ I_y(x,y) \end{pmatrix} \tag{6}$$

3. Categorize gradients into equally spaced bins by their orientation

$$\theta = \arctan(\frac{I_y}{I_x}) \tag{7}$$

4. Divide the image into equally sized blocks, eg. 9x9 pixels and normalize gradient strengths among these blocks. Dalal and Triggs got best results by using the "L2-norm":

$$v \to \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon^2}} \tag{8}$$

Where v is the unnormalized descriptor vector and ε a small constant.

Figure 10 presents the result of HOG feature extraction using different block sizes.

Region proposal functions To locate objects in an image using any kind of classifier, sub-regions of the image have to be determined, which are sized equally

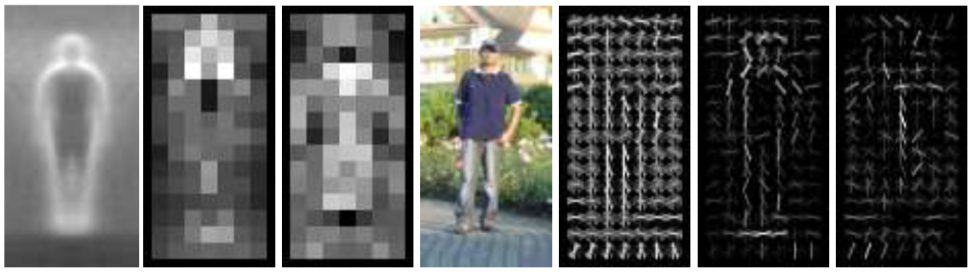


Figure 10. HOG features at different block sizes (original image at the center) [39]

to the expected size of the object. Table 3 lists the most relevant methods for generating region proposals. The simplest way to create such region proposals is to use a **detection window**, which is incrementally slid across the entire image. For each detection window, a feature map is computed. The step in which the detection window is moved forward may be as little as one pixel, but not larger than the detection window itself. A smaller step size decreases the chance of missing an object, although it increases the number of computations and memory required to store all extracted features [44]. If, like in most applications, objects also occur at different scale, the sliding window is applied to re-scaled versions of the image. The resulting pyramid of features can even be enlarged by additionally rotating the image step-wise before moving the detection window along. This way, even objects with orientation outside the capabilities of DPM become detectable. Of course, every additionally created feature map increases required memory and computational cost.

Another approach, which is useful in some cases, are Maximally Stable Extremal Regions (MSER) [45]. However, while less computationally expensive than sliding windows, MSER are only suitable for detecting larger objects in images and therefore not relevant to this study.

Superpixel oversegmentation is a quite popular method for generating region proposals. In 2010, Achanta et al. introduced the SLIC algorithm (Sim-

Table 3. Methods for region proposal generation

Region proposal function	Purpose	Computational cost
Sliding window	locate rigid object	intermediate
Sliding window + scale	locate object at different scales	high
Sliding window + scale + rotation	locate object at different scale and orientation	very high
Maximally Stable Extremal Regions (MSER)	Locate blobs	intermediate
Superpixel oversegmentation	Locate connected contours	low

ple Linear Iterative Clustering), which concatenates similar pixels into clusters. The resulting mosaic-like image can be processed much faster compared to sliding window detectors, since the clustering step heavily reduces the number of features maps, which only are computed once per superpixel [46]. More details about superpixels are given in section 3.1.2.

Classification After feature maps have been computed for all region proposals, it must be determined whether each region contains the object of interest or not. This is a standard pattern recognition task in machine learning with many algorithms capable of solving it. However, since literature has proven **Support Vector Machines** (SVM) to be most suitable for DPM-based object detection, this section is only focusing on this kind of classifier [39, 38, 44]. Other classifiers such as decision trees, naive-Bayes or nearest-neighbor classifiers are extensively covered in the book of Chen from 2015 [47].

Support vector machines are algorithms used to divide a set of objects in such a way that the space outside the class boundaries becomes maximal. Figure 11 provides an example for separating point clouds in 2D-space using lines, however, SVMs can perform classifications in much higher dimensions. In this case, objects are separated into classes by **hyperplanes**. Objects, like region proposals in the case of computer vision, are defined by a vector. For object detection, the feature map as described above is vectorized and used for object representation.

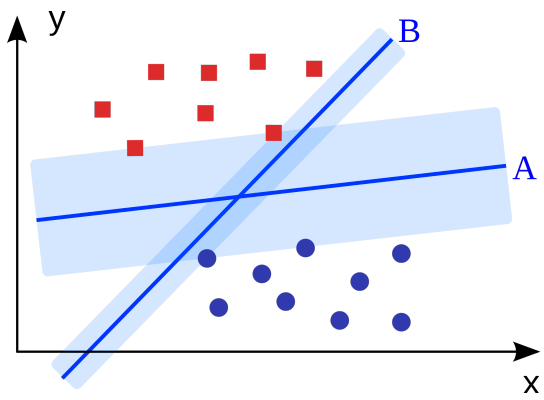


Figure 11. Two possible solutions for hyperplanes separating two object classes) [48]

A hyperplane is determined by providing the SVM with a set of m labeled training samples. For every training sample x_i, the class y_i is known. In object detection, training sets may use between a couple of dozen and up to several thousand training samples.

$$\{(\vec{x}_i, y_i) \mid i = 1, ..., m; y_i \in \{-1, 1\}\} \tag{9}$$

Training generates a decision function, in the case of equation 10 a hyperplane given by normal vector $\vec{w}$ and bias b. Although this section only covers linear SVMS, they may also be used to solve nonlinear problems using different kernel function for decisions [49].

$$y_i = sgn(\langle \vec{w}, \vec{x}_i \rangle + b) \tag{10}$$

To retrieve a classification rule, the problem is formulated as a quadratic program, where ξ_i is a slack variable to consider non linearly separable data and C a positive constant:

$$Min \quad \frac{1}{2}\|\vec{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i \tag{11}$$

$$\text{s.t.} \quad y_i(\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1 - \xi_i \quad \text{for all} \quad 1 \leq i \leq m \tag{12}$$

The program is then solved by using its dual form [50]. Today, support vector machines are in most software packages vastly based on the "SVMLib" implementation of Chang and Lin from 2011 [51].

Convolutional neural networks

To conclude this section, convolutional neural networks (CNN), which are currently state of the art in object detection are briefly introduced. CNNs significantly outperform DPMs on several benchmark datasets. While DPMs are built around "handcrafted" features, which require a lot of domain-specific fine-tuning, CNNs are capable of learning entirely new features from a labeled set of training data. CNNs were already introduced in 1995 by Lecun et al., but could not efficiently be trained before the early 2010s due to lacking computational power [52, 53]. Features created by convolutional neural networks are also proven to be illumination and rotation-invariant [54].

Idea Convolutional neural networks are a special subclass of neural networks, inspired by the human visual cortex. The visual cortex is built from individual neurons responding to stimulation of receptive fields on the retina. This design is also reflected in the architecture of CNNs: CNNs are built from a convolutional layer, followed by a pooling layer. If these two layers are repeated several times, the CNN is referred to as **deep convolutional neural network**. Figure 12 illustrates the general structure of CNNs. When used for image processing, a small matrix (e.g. 9x9), also called "convolution matrix", is slid over the input image. These convolutional matrices are the machine learning equivalent of biological receptive fields. A convolution operation computes the dot product of the input and the entries in the convolutional matrix. The result of this operation is called a "neuron".

A convolutional layer is formed by combining all those neurons. The resulting

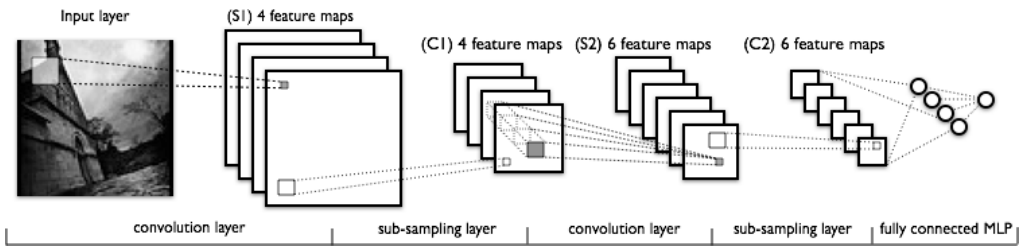


Figure 12. Structure of a CNN [55]

2D-matrix is equivalent to feature maps as used in DPMs [56]. Pooling layers are used to downsample the convolutional layer, which can grow very large in deeper networks. Pooling is done by only keeping the maximum value of directly adjacent neurons. It also prevents the CNN from overfitting [57]. In a last step, one or multiple fully-connected layers are used to perform the actual classification task. In a fully-connected layer, every neuron is connected to every other neuron in the previous layer, equivalent to the architecture of regular neural networks [49]. The number of neurons in fully-connected layers corresponds to the number of object-classes. The neuron, which is activated the most in the last layer points to the object class, which is shown on the input image.

Recent developments Driven by the major innovation of performing matrix computations on graphics processing units (GPUs), the development in the field of convolutional neural networks gained a lot of interest in the research community. Additionally several authors contributed very large datasets with labeled images for training and testing neural networks. The largest and most diversified dataset is the ImageNet database, which currently contains over 14 million labeled images from over 22,000 object classes. It was created by the Stanford Vision Lab [58].

Among many different CNNs, the AlexNet implementation currently performs best on ImageNet data. AlexNet is a deep neural network with a total of 26 subsequent neuronal layers [53]. However, development is evolving quite fast. Forrest et

al. proposed SqueezeNet in early 2017. Although their paper is still under review, it has gained quite a lot of attention in the field, as it increased performance while reducing required GPU memory by more than 500 times.

For object detection in images, CNNs are combined with region proposal techniques as described above (see section 2.3.1). The region-based convolutional neural network (R-CNN) implementation developed by Girshick et al. is currently performing best on various benchmark datasets [56]. With the recent publication of "Fast R-CNN" and "Faster R-CNN", his research group proposed a slightly less accurate object detector, which can be used in real time applications with processing up to 60 images per second [59, 60]. Another real-time object detector based on CNN at equivalent performance levels is the "YOLO"-network (You Only Look Once) by Redmon et al. [61]. This is an important breakthrough in many fields that rely on computer vision, like autonomous driving.

2.3.2 Image processing in traffic-related applications

Object detection became of great importance for a variety of different traffic applications. Methods of computer vision are successfully used for extracting roads from satellite images and traffic surveillance purposes. Even parking area shape extraction has been attempted already. After subsuming the different approaches in image processing, this section provides a brief summary of computer vision application in traffic-related fields.

Road network extraction Previous research has proven satellite imagery and aerial photography to be a valuable resource of static traffic and infrastructure information. In 1995, Gruen et al. described a first viable semi-automatic approach for road extraction from satellite images that used wavelet transform and dynamic programming [62]. Several years later, Chanussot et al. began using mathematical

morphology for road detection [63]. Other researchers used higher-level features and adjusted DPMs for road extraction [64]. Today, automatic road extraction is widely used for creating new and more accurate maps of road networks [65].

Traffic monitoring Computer vision techniques are also widely used for all different kinds of traffic monitoring. For example, stationary camera systems are used for measuring parking occupancy and traffic density. Also, modern vehicles are often already equipped with systems for sensing pedestrians or detecting traffic signs. In this context, only literature using image processing methods will be discussed. Techniques for video analysis are not within the scope of this study.

In 2012, Mogelmose et al. introduced a system for detecting traffic lights and signs in real time with onboard vehicle cameras. The system used HOG and Haar-like features and a linear multiclass SVM classifier for detection [66]. Although the authors were unable to obtain enough training data to achieve a robust model, their research was an important contribution to the emerging of autonomous driving.

Techniques for counting vehicles in traffic and measuring parking occupancy are very similar in general. All published previous approaches used stationary surveillance CCTV cameras (Closed-Circuit TeleVision) in a fixed field of vision. To detect the presence or absence of vehicles, certain image parts are manually pre-selected and then analyzed by computer vision methods [67]. In 2007, True developed an algorithm for occupancy measurements of parking lots. He achieved good results using color histograms as features and a k-nearest-neighbor classifier (KNN) [68]. While invariant of illumination differences across the image and throughout time of the day, the method was susceptible to partial occlusion of neighboring parking spots. Lixia and Dalin used a similar setting, but based their work on LBP features and a SVM instead. Their solution showed a combined error rate (missing rate + false rate) of only one percent [69]. However, it must be

mentioned that the authors only used daylight images of two parking spots at low level of variance in illumination, which simplifies the problem.

Huang and Vu were able to overcome the problem of partial occlusion of parking spots when surveying larger parking lots by using a cuboid 3D-model of a parking spots, rather than 2D-representations. To avoid illumination issues, the authors worked with HOG features, which they extracted of each cube surface per parking area. Their model achieved an averaged accuracy of 99.5 percent, which outperforms all previous techniques by one to five percent [70].

Figure 13 shows an example for surveillance of several on-street parking spots in Braunschweig. The presented implementation was developed as part of the AIPARK project and is based on a combination of color and HOG features. For classification, a linear SVM is used. The system performs similarly to the methods described by [69] and [70]. Additionally, this system is also able to deal with infrared-images during night hours.

Parking area extraction from remote sensing imagery While road extraction from aerial and satellite images is a well researched task, literature barely covers parking area localization in remote sensing images. Only three papers approached this problem to some degree in the past. In 1998, Wang and Hanson developed a method for measuring parking area occupancy based on an elevation model. With a pair of images (stereo image) and given camera parameters, it is possible to compute the approximate height above a reference plane at every pixel. With use of this elevation map, Wang and Hanson were able to identify and count individual vehicles [71]. Although quite sophisticated, their approach was not further developed, because there is no available stereo remote sensing imagery source above the experimental level. Also, their method was only able to measure occupancy, but not identify parking area boundaries with respect to the surrounding

Figure 13. HOG + SVM detector for measuring parking occupancy

scene.

Over ten years later, Seo et al. tried a different approach in identifying parking areas from aerial images: They tried to detect parking areas by their characteristic road markings. As road markings are usually white and in high contrast to the street, Seo et al. filtered their image scenes by the aid of mathematical morphology methods. As parking spots mostly occur in "clusters", they additionally developed an adjusted form of deformable part model to consider the relative position of road marking to each other. Figure 14 depicts their parking area model [72]. Although their detection method of road markings works well due to the use of mature technology, it fails if markings do not have enough contrast with respect to the road or if parking areas are not marked at all. Mathematical morphology requires clean images with high contrast separating the objects of interest from the background. This requirement is typically not given in the case of remote sensing

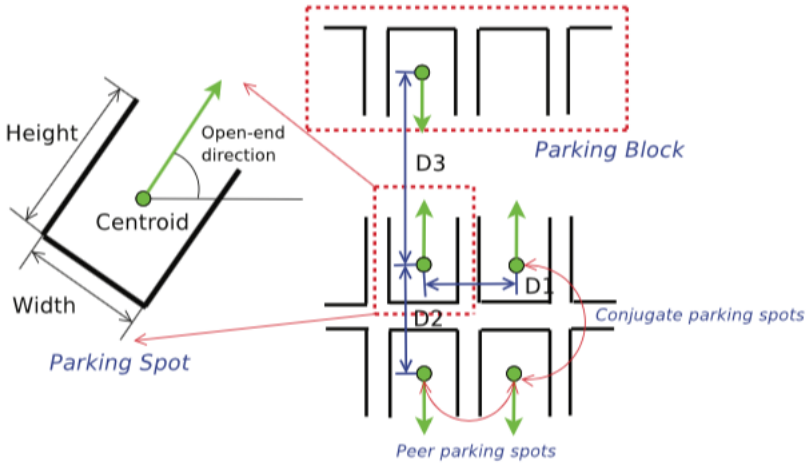


Figure 14. Parking spot model as used by Seo et al. [72]

images, which are often cluttered and captured at low contrast.

The most recent paper from Mexas and Marengoni from 2014 tried to address this issue. The authors tried to combine road extraction and vehicle detection methods to identify parking areas. Their approach is based on the assumption that clusters of vehicles on the road background form parking areas. To localize parking areas, they first detected vehicle clusters using morphological filter operations. Afterwards, parking area outlines were derived from the surrounding asphalt background, which is detected again by morphological operations [73]. Although this is the most comprehensive approach yet, it is still prone to various sources of error. First, the authors were unable to generate satisfying vehicle detection results. Second, the idea of drawing parking area outlines around vehicle clusters by utilizing road extraction methods resulted in quite distorted and inaccurate parking area boundaries.

All previous approaches did not provide any solution which could be used to detect roadside parking areas, which represents the majority of urban parking areas [15, 3].

2.3.3 Remote sensing of vehicles

As vehicle detection from remote sensing images is one of the core techniques used in this study, a summary of the literature in this area is provided, as well as a overview of available sources of remote sensing imagery.

Sources of remote sensing imagery Two main types of imagery may be used for remote sensing parking areas: Aerial images and satellite imagery. Their main difference is given by the maximum ground resolution. Although satellite image sensors have undergone major improvements in the past several years, the resolution of aerial photogrammetry with ground resolutions up to 7 centimeters per pixel is still superior [74]. Satellites, on the other hand, provide generally higher capturing frequency at much larger land coverage. Aerial photogrammetry only conducted once per year or even less in many places [75].

Best ground resolution on satellite images is only achieved on panchromatic (grayscale) images. Colored satellite images using multiple spectral bands are still behind in terms of resolution per pixel. On the other hand, most satellite sensors cover broader spectral bandwidth including infrared bands unlike aerial photogrammetry. Both sources are prone to bad weather and cloud cover [76]. Unless specified otherwise, the term "remote sensing imagery" is used for referring to both aerial photography and satellite imagery within this thesis.

Table 4 compares all data sources for remote sensing images, which are publicly accessible as of 2017, in terms of performance parameters and cost. Satellite data is retrieved from www.satimagingcorp.com, the sales company of Digital Globe, which operates all public remote sensing satellites [77]. Aerial photogrammetry is offered by multiple organizations, which mostly only cover the country in which they are located. In Germany, for instance, aerial imagery may be obtained from state-level bureaus for geodesy [78].

From an economic perspective, aerial images show a better resolution/cost-ratio than satellite imagery. In absolute numbers, especially archive images (older than 90 days) are less expensive to obtain. This price advantage of satellite images shrinks drastically then comparing new images (not older than 90 days) [79]. It is also worth pointing out that pricing depends on several more factors including area size, cloud coverage, resolution, spectral band and image age [80].

Table 4. Comparison of different sources of remote sensing imagery, based on [78, 77, 80, 79]

Sensor	Ground resolution	Revisit	Spectral	Dynamic	$Cost/km^2$ BW/color	
	BW / color (m)	frequency (d)	band	range	archive	new
WorldView-4	0.31 / 1.24	1	Panchromatic 4-band	11 bit/px	n.a.	n.a.
WorldView-3	0.31 / 1.24	1	Panchromatic 8-band	11 bit/px	$14.00 / $19.00	$24.00 / $29.00
WorldView-2	0.46 / 1.84	1.1	Panchromatic 8-band	11 bit/px	$14.00 / $19.00	$24.00 / $29.00
GeoEye-1	0.46 / 1.84	2.8	Panchromatic 4-band	n.a.	$14.00 / $17.50	$24.00 / $27.50
QuickBird	0.65 / 2.62	1	Panchromatic 4-band	11 bit/px	$14.00 / $17.50	n.a.
IKONOS	0.82 / 1.0	3	Panchromatic 4-band	11 bit/px	$10.00 / $10.00	n.a.
Pleiades-1A	0.5 / 2.0	1	Panchromatic 4-band	n.a.	$13.00 / $13.00	$23.00 / $23.00
Aerial Photogrammetry	0.3 - 0.07 color	n.a.	4-band	n.a.	$15.00	$35.00

Main advancements in the field Detecting vehicles in remote sensing images is a relevant task in different areas of research, including infrastructure, geodesy and traffic. From a computer vision perspective, vehicle detection in remote sensing can be considered a sub-problem of the general object detection problem. In remote sensing images, vehicles are always depicted from a top view and also roughly have the same size. However, vehicles may occur in any possible color and orientation. Since they are very small, vehicles are only composed of very few pixels in remote sensing images making them harder to detect.

Following the progress made in computer vision, several papers have been published regarding vehicle detection in remote sensing imagery. Table 5 summarizes

contributions, limitations and achieved detection accuracy for the most influential papers in the field. With advancements made in computer vision, vehicle detection accuracy also increased over time.

Three developments are of special importance to this study. First, providing context to the detection problem by masking the image using GIS based road network data effectively reduces the chance for errors, but also processing time. The general idea was to remove image parts like buildings or planted areas before detection starts, as these parts will not contain any vehicles. Gerhardinger et al. used road network data first in 2005. Three years later, Hinz et al. provided more implementation details on how GIS data can be used during vehicle detection [81, 82]. Second, Teng et al. started using superpixels instead of sliding windows for region proposal generation [83]. Since vehicles are very small compared to full image size, an extremely large number of detection windows was necessary leading to high numbers of false positive classifications in previous papers. Superpixels reduced the amount of detection windows by an order of magnitude, while preserving all information required for detection.

Table 5. Main developments in vehicle remote sensing (* = estimate derived from stated results)

Authors	Year	Feature	Detector	Advancement	Limitation	Accuracy
Gerhardinger et al. [81]	2005	Morphological analysis	Blob detector	GIS map data to provide context	Domain dependant finetuning required	85.45 % *
Leitloff et al. [84]	2005	Morphological analysis	Line detector	Detection of vehicle queues	No single vehicles Only dark vehicles	65.00%
Palubinskas et al. [85]	2008	Background subtraction between images	Blob detector	Detection of traffic congestions	Series of aerial images required	n.a.
Hinz et al. [82]	2008	Morphological analysis	Blob detector	Velocity estimation Implementation details for GIS data inclusion	Stereo images required	n.a.
Choi and Yang [86]	2009	Morphological analysis	Blob detector	Image segmentation	High false positive rate	76.04 %
Leitloff et al. [87]	2010	Edges	Gentle AdaBoost	Machine learning classifier instead Blob detector	Domain dependant finetuning required	82.50 %
Gleason et al. [88]	2011	Color histograms Gabor Kernel (edges)	SVM	Use of state of the art computer vision methods	High false positive rate Not suitable for urban areas	85.00 %
Teng et al. [83]	2014	HOG	SVM	Superpixel segmentation Edge sharpening	Rotation invariant classifier, only road background	n.a.
Liu and Mattyus [89]	2015	HOG feature pyramid	Soft AdaBoost	Vehicle type classification Direction classification	Insufficient implemtation details	74.00 % *
Shu and Du [90]	2016	Superpixel geometry HOG	SVM	Superpixel geometry as feature	Performance only sufficient on highways	88.90 %
Cao et al. [91]	2016	HOG	SVM	Transfer learning to process low resolution images	Extremely slow processing due to sparse representation of features	86.00 %
Li et al. [92]	2016	CNN	SVM	Use of state of the art robust CNN features	High computational effort for training	96.21 %

This brought another significant increase in detection accuracy. The third major improvement was contributed by Li et al. in December 2016. Instead of working with established HOG features, Li et al. used a 9-layer CNN to create rotation and scale invariant feature maps. After training their network with about 100,000 labeled samples, they were finally able to achieve classification accuracy greater than 95 percent [92]. A limitation to their method is given by the great computational effort to train a new neural network architecture from scratch, which requires sufficient GPU hardware.

2.3.4 The problem of detecting small objects

Several papers discussed the applicability and issues of state of the art object detection methods on low resolution images or small objects. While vehicle detection is a quite simple binary classification task where background only needs to be distinguished from vehicles, computer vision algorithms are often observed to perform worse than on their original domains [37]. Throughout this literature review, five reasons could be identified causing computer vision to perform less accurate on the problem of vehicle detection in remote sensing images.

1. Unlike in typical computer vision tasks, like pedestrian or face detection, vehicles on remote sensing imagery may be oriented in any direction. Rotation invariant features are required.

2. Vehicles are a quite diverse class in terms of color. They provide only very few edge and structure information. Hence, the chance of confusion with other rectangular objects like roof elements or containers is naturally high [82]. Errors due to this issue may be diminished by providing additional context using GIS data.

3. Due to the low resolution of remote sensing images compared to “normal”

images, the number of pixel per vehicle object is very small (ca. 5x10px - 10x30px). Region proposals therefore carry much less information than in the regular object detection problem. Chen et al. suggested up-sampling detection windows before using them for classification to reduce this problem [93].

4. The large ratio between image size and object dimensions in remote sensing images leads to a much higher number of region proposals compared to regular objects on normal images. This inevitably leads to a higher absolute number of misclassifications in vehicle detection [37]. Superpixel typically decrease the amount of required classifications by an order of magnitude [83].

5. Typical methods to remove duplicate detections, like Non-Maximum-Suppression (NMS) do not work well on small neighboring objects [94]. Razakarivony suggested applying a simple threshold on the overlapping area of bounding boxes instead of NMS [37].

When using convolutional neural networks for vehicle detection in remote sensing images, two additional issues have to be solved: ImageNet mostly consists of "iconic object images" rather than cluttered scenes or remote sensing images. Since most available CNNs are trained on ImageNet data, learned features may not be perfectly suitable to accurately represent small objects like vehicles [92]. The second problem is induced by the architecture of state of the art convolutional networks. Pooling layers have the effect of downsampling the input image as it is processed by the CNN. Early use of those pooling layers in deep networks tends to discard information before it can be converted into features [93]. Li et al. addressed this issue by their network architecture which is specifically designed for vehicle detection. They delayed the use of pooling in very late neural layers [92].

CHAPTER 3

Procedures

This study utilizes a variety of complex methods and techniques in order to extract parking area information from floating car data and remote sensing imagery. This chapter will first describe the process of extracting parking area positions and capacity from images. Subsequently, an in-depth explanation of the data mining steps used for extracting meta information from floating car data is given.

Data mining reference model CRISP The entire methodology, as described below, is built upon the widespread cross-industry standard process for data mining (CRISP-DM), which was originally proposed by Chapman et al. in 2000 [95]. The model, as shown in figure 15, divides the data mining task into six main tasks from "Business Understanding" to final "Deployment". The first phase serves the purpose of fully understanding the problem, objectives and requirements. During the "Data Understanding" phase, data quality is evaluated and potential issues identified. Interesting data subsets might also be discovered. "Data Preparation" is necessary to compose a final data set, cleaning and transforming it, so it can be fed into subsequent models. The main purpose of this phase is to identify and select data attributes, which are relevant to the problem. This process is also commonly referred to as "Feature Engineering". In the phase of "Modeling", various models are selected, calibrated and tested. Usually, several iterations between data preparation and modeling are necessary until the best combination of features and model is found. In the evaluation phase, model output is compared against the original objectives and requirements. Lastly, the final model or its outputs are deployed, which means applying it to real-world problems, but also defining and

monitoring its extent of validity. This study puts special emphasis on the steps of data preparation, modeling and evaluation.

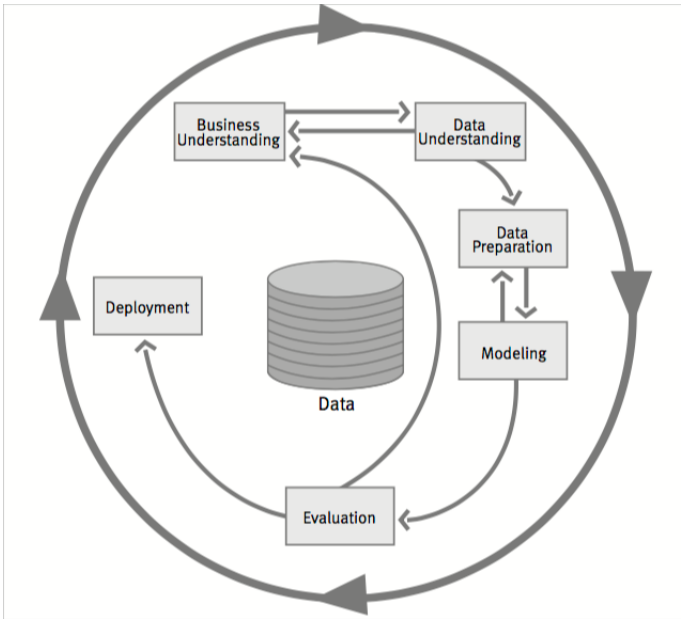


Figure 15. CRISP-DM (CRoss-Industry Standard Process for Data Mining) [95]

3.1 Parking area detection

The method for extraction parking areas from remote sensing imagery is designed as a multistage process. The process is shown in Figure 16 and consists of image segmentation, filtering non-street areas by adding of GIS road network information, removal of tree and road pixels, vehicle detection and finally, queue extraction. The proposed method puts strong emphasis on robustness, since each of these stages is prone to potential errors and technological limitations of computer vision. The idea and implementation of parking area detection is described using an exemplary test setting within the city area of Braunschweig.

3.1.1 Test setting

To demonstrate and investigate the developed method for parking area extraction, an exemplary aerial test image is used. The image depicts a 500 x 500

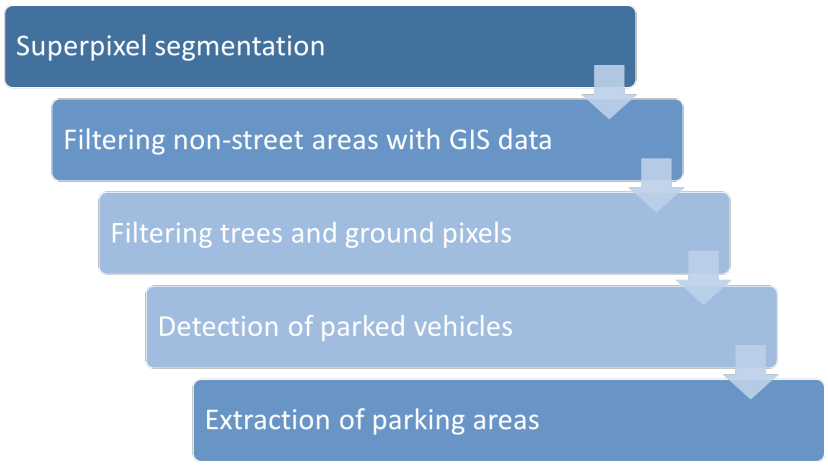


Figure 16. Multistage parking area detection process

meter area in the city of Braunschweig. The image is provided by the municipal government of Braunschweig. It is captured in RGB-colorspace and was taken at a ground resolution of approximately 15 centimeters per pixel. The test image is presented in Figure 17. The scene represents a very characteristic German urban district with mostly residential buildings, only segregated by a major street along the north-south axis.

The scene contains several challenges for parking area detection, similar to what would occur on a larger scale in production: Ground visibility is partly affected by trees and a large share of the image is covered with buildings, increasing the likelihood of missing vehicles or falsely classifying non-vehicle objects during analysis. Narrow streets and tall buildings create a broad range of different illumination levels across the image, which complicates the use of color-based image descriptors.

With having the entire depicted area already covered in the AIPARK database, validation may be conducted very straight-forward by comparing detection results with real-world reference data. The data set was partially collected manually during development of other AIPARK components, but is also composed

Figure 17. Test setting “Oestliches Ringgebiet” in Braunschweig, georeferenced image

of OpenStreetMap data.

3.1.2 Image segmentation using superpixels

Before the image is processed, it is divided into “superpixels” to decrease the number of classification operations in all following steps. Instead of operating on a single-pixel level and moving a sliding window pixel-wise across the image, similar pixels are merged into a superpixel, which is then used for all further processing. Thus, superpixel oversegmentation may be understood as a clustering algorithm. Considering the test image, which has a size of 2,901 x 3,018 pixels, a total of more than 8.7 million sliding windows would be required, if vehicle detection would be performed on the single-pixel level.

After superpixel oversegmentation, the image is divided into 25,000 approximately car-sized superpixels. The computational effort is now decreased to only 0.29 percent of the previously required operations.

For superpixel oversegmentation, the SLIC algorithm as described by [46]

is used. In the test image, a car has an approximate size of 30 x 13 pixels. To obtain car-sized superpixels, the maximum number of desired pixels is set to 25,000. Figure 18 presents a close-up view of the test image after the SLIC algorithm has been applied.

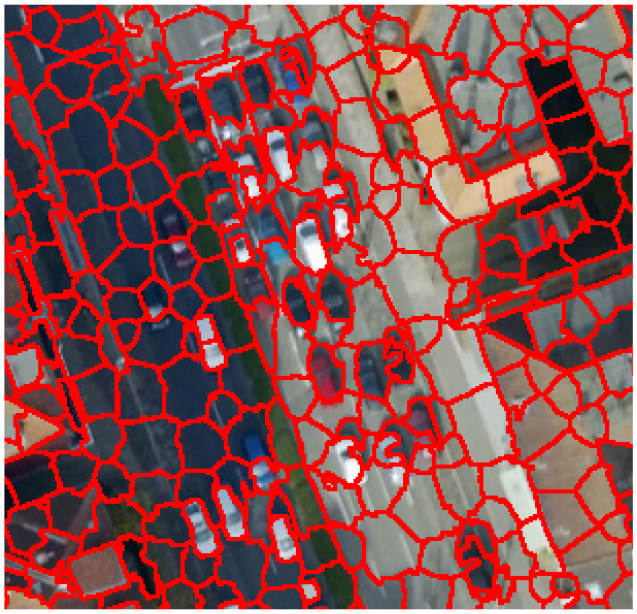


Figure 18. Closeup image after superpixel segmentation

3.1.3 Using GIS data for determining regions of interest

Before the actual object detection algorithm is applied to the image, regions of interest are defined in order to reduce the number of detection operations even further. Specifying ROIs increases robustness of the entire extraction process, as the chance of potential misclassifications is reduced. As the goal is to extract road-side parking areas, only image regions containing streets and their immediate surrounding are of interest. All other image parts may be discarded as they do not contain any information serving the given purpose.

ROIs are determined by mapping the image to the earth's coordinate system

and using road network information provided by GIS data. The used method is strongly related to the road network masking described by [81] and improved by [82].The position of the upper left and lower right corner of the image is given in WGS84 coordinates (World Geodetic System 84). Although the area depicted on the test image of only 500 x 500 meters is fairly small in relation to earth's radius, nonetheless, spherical distortions are corrected by converting Spherical earth coordinates into the Cartesian image coordinate system. Coordinate transform is done by using the following conversion formula as described by [96]:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} N\cos(\varphi)\cos(\lambda) \\ N\cos(\varphi)\sin(\lambda) \\ [(1-e^2)\,N]\sin(\varphi) \end{pmatrix} \tag{13}$$

where

$$N = \frac{a}{\sqrt{1 - e^2 sin^2 \varphi}}$$

and x, y, z - Cartesian coordinates of a point on the ellipsoid,

φ, λ - geodetic coordinates: latitude, longitude

a, b - semi-major and semi-minor axes of the ellipsoid

N - radius of curvature in the prime vertical

e^2 - first eccentricity squared

$$e^2 = 1 - (\frac{b}{a})^2$$

Values a,b, and e^2 are retrieved from WGS84, which specifies necessary parameters to create a reference ellipsoid to earth.

The geometric shape of the road network is retrieved from OpenStreetMap GIS data. The OSM XML-file (eXtended Markup Language) contains a skeleton of waypoints as well as the standard width for various road types, such as motorways, highways or residential streets. To include potential roadside parking areas, the

standard width of every road is extended by ten meters on each side. Subsequently, the entire road network shape is merged into one single polygon, which is then used for masking the image regions containing roads (see Figure 19).

To achieve low error in parking area extraction, traveling vehicles must also be removed from the image. Within this context, vehicles are distinguished into parking and traveling by their position on the road. Roadside vehicles are more likely to be in parking position, while vehicles closer to the road center line are more likely to be in travel mode. Hence, in a next step the road outline itself is also removed from the image by applying a masking operation. The original road width is reduced by two meters on each side. The remaining image now only contains a narrow ribbon of 12 meters width, which is the final region of interest for parking area extraction. Remaining traveling vehicles that are still contained in the image are treated as outliers and removed in later processing steps by forming vehicle queues (see section 3.1.6). Masking unnecessary image parts is achieved by defining a $n \times m$ masking matrix M of zeros, where n is the input image's width and m is the height, respectively. For image pixels outside the defined regions of interest, the mask elements are set to one $M(x, y) = 1$. Figure 19 presents the final result after georeferencing the image and masking regions of interest with GIS data. Only image areas which are not masked in red color will be used in further processing. For this particular example 68.4 percent of the image area has been masked, which means further increase in performance, since the cropped image parts will not be considered during detection phase.

3.1.4 Removal of trees and road background

To decrease computational effort and the likelihood of false positives during vehicle detection even further, image parts containing parks, lawns, bushes, trees and roads are removed. However, as the vast majority of "green areas" in the

Figure 19. Image detail after applying GIS masking

test image is represented by trees, all green areas are referred in the following section as one single "tree" object class. Unlike vehicles, "trees" and "road" are very homogeneous object categories in terms of color and structure, which makes them much easier to detect.

Detection of those background elements is performed by iteratively applying a HOG and color based three-class linear SVM classifier to all superpixels which remain after GIS filtering.

Feature engineering The task of classifying trees and roads on superpixel detection windows is fairly similar. Trees share a limited range of brown or green colors and have an undirected very fine structure. The "street" object class is char-

acterized by a narrow range of gray colors at different levels of intensity. In general "street" objects do not contain any kind of structure. However, street markings may be present on higher resolution images. In this case, road markings can be considered as bright, line-like blobs, which may occur in any orientation.

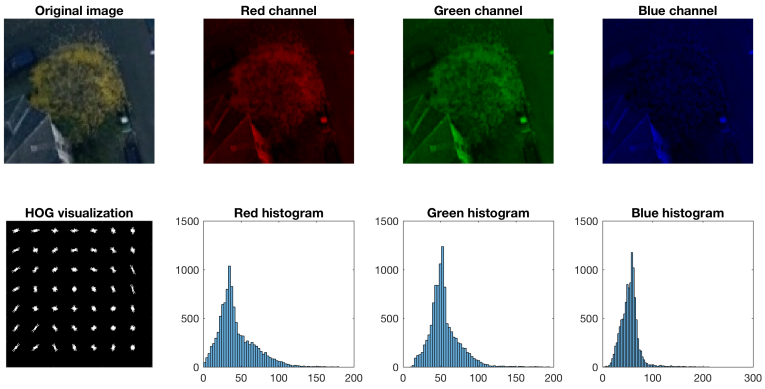


Figure 20. Example color and structure features for background detection

All described characteristics of trees and street may appropriately be reflected by color histograms and HOG features to model texture. Image 20 provides an example of these features for a superpixel detection window containing a tree. The same features applied to classify streets without any changes. The overall feature vector per detection window is formed by concatenating all histograms and the vectorized HOG descriptor into one array $\vec{y}_i$.

$$\vec{y}_i = \left[\vec{hist}_{green}, \vec{hist}_{red}, \vec{hist}_{blue}, \vec{hog}\right]; \tag{14}$$

Model As the literature suggests, two linear support vector machines are used to classify superpixels containing tree or street instances. For greater robustness, two separate SVMs are used to detect both object classes. Allwein et al. proved superior accuracy of multiple binary classifiers to one single multiclass SVM in case of only very few different object classes. [**?**].

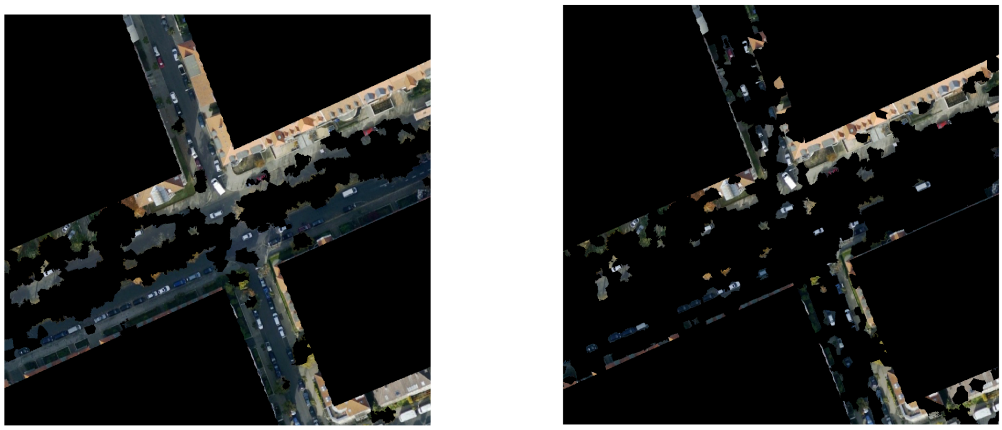
Training and road detection Since both object classes, roads and trees are very homogeneous, sufficient SVM training may be conducted using a relatively small training data set. Figure 21 presents montages of the training data. The street classifier is trained using 100 images per object class at randomized order. To avoid overfitting effects, 10-fold cross-validation is applied.

Figure 21. Data samples used for SVM background detector training

Regarding the tree classifier, only 50 samples per object class are necessary for training. The distinct and narrow band of green and brown colors makes trees and other green areas very easy to distinguish from other image contents. Nonetheless, also this classifier is trained on randomized data and validated using 5-fold cross-validation.

Both classifiers are applied to the masked image, as retrieved from the prior GIS filtering step. Before street instances are identified, superpixels containing trees are detected and removed. Hence, double classification of superpixels is prevented and processing therefore accelerated. Successively, all remaining superpixels are processed yet again to detect and discard pixels containing street instances. Removal of discarded superpixels $S_{discaredd}$ is achieved by adding them to the mask retrieved from the previous step $M([S_{discarded}]) = 1$. Figure 22 illustrates the result,

after tree and road superpixels have been removed.



(a) Image after removing tree superpixels

(b) Image after removing tree and road superpixels

Figure 22. Original image after removal of tree and road instances

3.1.5 Vehicle detection

As already pointed out in sections 2.3.3 and 2.3.4, the detection of small objects like vehicles at relatively low resolutions is still quite challenging for computer vision. Two main reasons account in literature for higher error rate when detecting vehicles in remote sensing images: The first is that compared to regular object detection problems, small objects carry much less information making it harder to extract distinct features from them. The second problem is given by the fact that the task of detecting many small objects instead of a few large will inevitably produce higher absolute numbers of misclassifications. The amount of necessary classification operations increases quadratically with shrinking object size. While the first problem of insufficient information in small objects cannot be resolved with current state of the art computer vision techniques, the absolute number of classification operations may be reduced by pruning the problem before the actual vehicle detection starts.

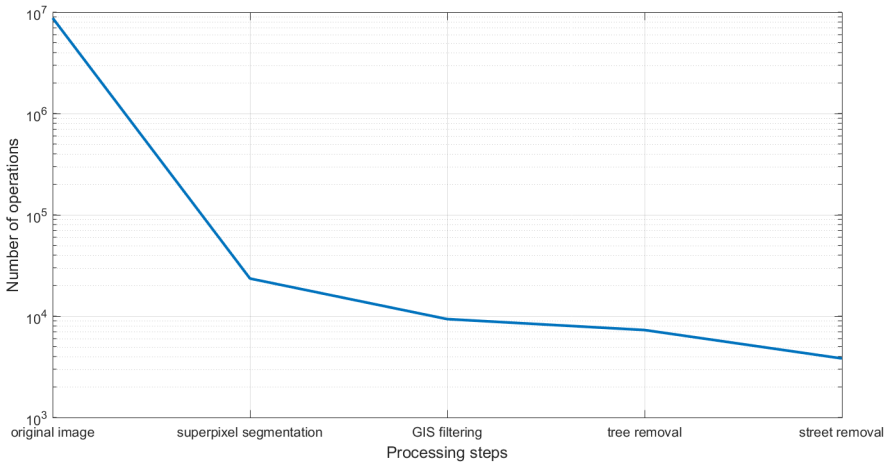


Figure 23. Amount of necessary vehicle classification operations

The described multistage process beginning with superpixel segmentation and ending at street removal effectively reduces the size of the original problem, which also means a great decrease of misclassifications by absolute numbers. The additionally induced error caused by the described upstream steps, however, is very small compared to that. As shown in Figure 23, successively pruning the image shrinks the vehicle detection problem from more than 8.7 million classifications down to only about 3,800 necessary operations after trees and ground have been removed.

Feature extraction using the AlexNet CNN To detect vehicles in the remaining superpixels, rotation-invariant features are required to capture vehicles at all different directions. The features also must be capable of adequately reflecting the difference to image parts similarly structured to cars, like certain parts of roofs for instance. Lastly, in terms of color, the features must be able to reflect the prevalently very fine difference between gray vehicles and street.

After considering and testing various feature types described in 2.3.1, feature maps as generated by convolutional neural networks are chosen. However, training

a specialized network for vehicle detection, like [92] did, is not an option within the scope of this study, due to the insufficient GPU performance capabilities of the used hardware setup. Training of a nine layer architecture including five subsequent convolutional layers requires at least six gigabyte of GPU memory, while the used system only has one gigabyte available. The popular technique of adjusting a pretrained network instead of training an entire CNN from scratch is also infeasible, as even this method would require at least four gigabyte GPU memory.

To overcome this issue, a very common workaround in computer vision is applied: The tasks of feature map generation and classification are split up. Classification is not done by softmax layers at the rear end of the CNN itself but by a separate support vector machine. This separate SVM classifier is fed with the output map of the last fully connected layer in the convolutional neural network.

The AlexNet architecture by Krizhevsky et al. is used for generating feature maps [53]. The network is 25 layers deep and is one of the best performing implementations as of early 2017. AlexNet was trained on approximately 1.2 million images from the ImageNet dataset. For creating the feature vector, which is then fed to the SVM, neuron activations in layer 23 which is the last fully connected layer ("fc8") are written into an array of floating point numbers.

Model Classification of superpixel in vehicle and background categories is performed by a support vector machine with quadratic kernel. The SVM is implemented according to the details provided in sections 2.3.1 and 3.1.4 for tree and road detection. To obtain an estimate on how reliable the SVM result is, the classification score proposed by [97] is used. As shown in equation 15, the classification score is the signed distance between observation $\vec{x}$ and the decision boundary given by support vectors $\vec{x}_j$. Thus, a high classification means greater distance from the

decision boundary and therefore lower chance of false classification.

$$f(\vec{x}) = \sum_{j=1}^{n} \alpha_j y_j G(\vec{x}_j, \vec{x}) + b \tag{15}$$

where $(\alpha_1, ..., \alpha_n, b)$ are the SVM parameters, $G(\vec{x}_j, \vec{x})$ is the dot product between observation $\vec{x}$ and the support vectors $\vec{x}_j$ and y_j equals 1 for the positive and -1 otherwise.

Training and vehicle detection The SVM is trained using a total of 42,000 image samples, which are divided in two classes: "vehicle" and "background". Figure 24 shows 400 examples of each of those classes. The dataset is created by manually labeling 5,250 vehicles on different sample aerial and satellite images and rotating them three times in a 90 degree steps. With this method, the variance in vehicle rotation is artificially increased. Background samples are generated automatically by dividing the remaining parts of the training images in small windows, which do not contain vehicles. Since remote sensing images consist of much more background than vehicles, the larger background dataset size is decreased by randomly removing samples until the number of samples in both classes is equal.

Figure 24. Training dataset for vehicle detection (400 examples per class)

In a next step, features are extracted from all training samples and stored in a table together with their corresponding class label. The labeled feature table is then used for training the support vector machine. After training is completed, the SVMs error rate is determined with the method of 5-fold cross-validation. Since the used dataset is fairly large, using more folds than five would be computationally very expensive and also unnecessary due to the high number of samples per fold. During cross-validation, the training data set is split in five partitions of approximately equal size. Partitions one to four are then used to train a new SVM with the same configurations as the original one. The error rate of this classifier is evaluated by predicting the fifth partitions' class labels. This process repeated four more times to compute the generalized classification error according to equation 16:

$$Generalized\ error = \frac{\sum_{i=1}^{n=5} False\ positives_i + \sum_{i=1}^{n=5} False\ negatives_i}{Total\ number\ of\ classifications} \tag{16}$$

The final SVM is then applied to the remaining superpixels of the test image in order to find those pixels which contain vehicles. Figure 25 illustrates the final result of this last vehicle detection step. Masking the road center lines before detecting vehicles effectively helped distinguish parked from moving vehicles.

The small number of misclassifications is due to inaccuracies in the road network or abnormal local road shapes. A few false positives are caused by roof parts with vehicle-like shape. Lastly the vehicle detection algorithm also failed to detect some dark vehicles which are similar to the road in terms of color. However, since the occurence of the mentioned errors is distributed very randomly across the entire image, these errors will not confuse the subsequent step of forming roadside parking areas as described in section 3.1.6. An in-depth evaluation of the entire vehicle detection process is provided in chapter 4.

Figure 25. Final result of vehicle detection process (before removal of duplicate bounding boxes)

Removal of duplicate bounding boxes After vehicle detection is completed, duplicate bounding boxes must be recognized and removed. Duplicate detections may occur due to the way superpixels are arranged. If vehicles are split up into two pixels or if centroids of neighboring superpixels are close to each other, bounding boxes may overlap each other. Figure 26 provides an example for overlapping detection results. In this case, the red bounding box was generated because the white vehicle was separated into two superpixels and therefore classified twice. Also, the lower green bounding box overlaps the upper green box by a small percentage. However, this overlap can be neglected, since the lower green bounding box contains a true positive detection.

To eliminate duplicate bounding boxes BB like the example in Figure 26, an adjusted form of non-maximum-suppression (NMS) method that like described in [94] is applied. Chen et al. found NMS to be not sensitive enough in small

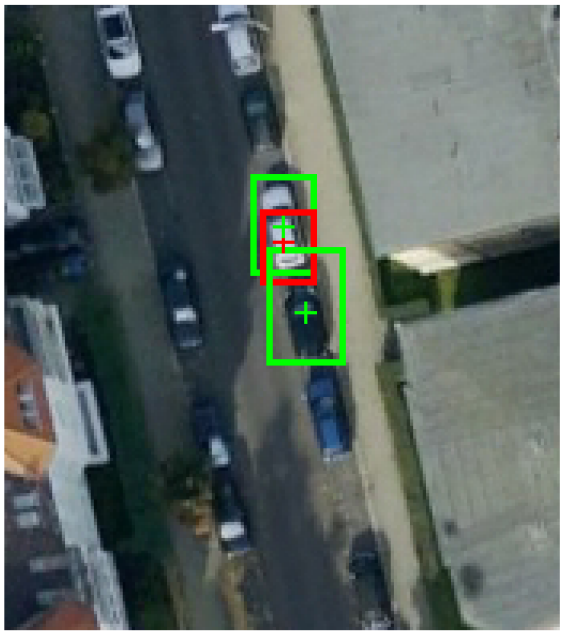


Figure 26. Overlapping bounding boxes

object detection applications [93]. For this study, only bounding boxes which intersect more than 20 percent with each other are removed. If one of the remaining intersecting boxes has less intersections than the other boxes, it is removed. In case of equal numbers of intersections, the box with the largest intersection area is removed. In the unlikely case of equal areas among bounding boxes, one of them is removed at random. To keep computational complexity low, only neighboring bounding boxes which potentially could overlap at all are processed. Algorithm 1 represents a pseudo-code notation of this process.

3.1.6 Finding queues of parked vehicles

In a last step, parking areas are generated by merging the extracted vehicle positions into sets. In contrast to the method proposed by Mexas, parking area outlines are not retrieved at the pixel level, but rather from the relative position of vehicles [73]. This approach is highly advantageous, as it avoids the common problems of shape extraction, namely errors through differences in illumination

Algorithm 1 Elimination of overlapping bounding boxes BB

```
for all i in BB do
    get nearBB with dist(BB, BB_i) < 70px;
    compute intersection areas intArea(nearBB, BB_i);
    only consider intersections > 20%, nearBB(intArea/area(BB_i) ≤ 0.2) = [];
    count intersections intCount of each nearBB;
    if intCount(i) != intCount(j) then
        remove nearBB(min(intCount)) = [];
    else
        if intArea(i) != intArea(j) then
            remove nearBB(max(intArea)) = [];
        else
            remove nearBB(1) = [];
```

and misclassified vehicles.

The method for parking area shape extraction proposed in this section is light invariant and robust with respect to up-stream errors, like misclassified vehicles during image processing. A set of vehicles is merged into one parking area, if the following conditions are fulfilled:

- A parking area is formed by connecting vehicle center points (vertices) through edges
- A parking area must be formed by at least three vehicle vertices
- The edge length between two vertices is < 12 meters (twice the avg. parking spot length [3])
- The absolute angle between two edges who share one common vertice must not be larger than 6°
- The absolute mean angle between the parking area and the main direction angle of the corresponding street must not be larger than 6°
- A parking area may include one tree between two vehicle vertices to take the case of road-side trees into account.

Now, *dist* is an $n \times n$ euclidean distance matrix. The euclidean distance measure is used instead of spherical distances as the distance between adjacent vehicles is very small relative to earth's radius. If V is a set of detected vehicles and T the set of detected trees, then parking areas are determined according to the aforementioned conditions with algorithm 2. The algorithm checks for every element in $[V, T]$, if the conditions of a parking area line are fulfilled by its adjacent vertices. If so, a candidate area line is created from the investigated element and the respective adjacent vertice.

Algorithm 2 Parking area extraction

```
Init: loopSet = [V,T]; pAreas = []
while loopSet != [] do
    Init: tempVertice = loopSet(1); loopSet(1) = [];
    neighborDist = dist(1,:);dist = [];
    nearVertices = loopSet(neighborDist < 12 m)
    if nearNodes != [] then
        for i in neighbors do
            candidateLine = [tempVertice, nearVertices(i)]
            if nodeType(candidateLine) != 'tree,tree' then
                lineAngle = arctan(ΔcandidateLine_y / ΔcandidateLine_x)
                roadAngle = angle of corresponding road section
                if ||lineAngle − roadAngle|| < angleStreet_max and ||lineAngle −
  previousLineAngle|| < angleLine_max then
                    line += nearVertices(i)
                    if length(candidateLine) > 2 then pAreas += candidateLine
                    Recursive vertice insertion(candidateLine)
Remove duplicates from pAreas
```

The process is then repeated for the new vertice by calling the recursive procedure 3. The procedure continues more suitable vertices by calling itself repeatedly until at least one of the conditions is not met any longer.

After all elements in $[V, T]$ have been processed, duplicates and subsets are removed. The remaining areas which overlap each other are merged into connected lines. Figure 27 presents the final parking area extraction result. Road sections are marked in green, road side parking areas are depicted in red. By consistently

Algorithm 3 Recursive vertice insertion

```
1: procedure RECURSIVE VERTICE INSERTION(line)
2:     tempVertice = line(end)
3:     neighbors = dists(tempVertice) < distanceThreshold
4:     if neighbors != [] then
5:         for i in neighbors do
6:             candidateLine = [tempVertice, neighbors(i)]
7:             if nodeType(candidateLine) != 'tree,tree' then
8:                 lineAngle = arctan(Δcandidate_y / Δcandidate_x)
9:                 roadAngle = angle of corresponding road section
10:                if ||lineAngle − roadAngle|| < angleStreet_max and ||lineAngle −
   previousLineAngle|| < angleLine_max then
11:                    line += neighbors(i)
12:                    if length(candidateLine) > 2 then pAreas += line
13:                    RECURSIVE VERTICE INSERTION(line)
```

aligning the generated parking areas parallel to the road and forbidding larger angle between vertices, the process becomes robust with respect to false negative parking vehicle detections. Those two rules would not allow outliers, like moving vehicles on the road, to be included in a line of parked vehicles. However, as seen in the upper left corner of the image, roads which are fully covered by trees can not be processed yet with the proposed method. The capacity of extracted parking areas is estimated by dividing the parking area line length by the average vehicle parking spot length of six meters [3].

3.2 Metadata extraction

Within the context of this thesis, the terms metadata or meta-information are used for referring to any kind of information that further describes a parking area besides its geographic location and shape. The presented process aims to acquire as much metadata as possible by systemically analyzing floating car data as a primary source of information which is enriched with additional spatial demographic data.

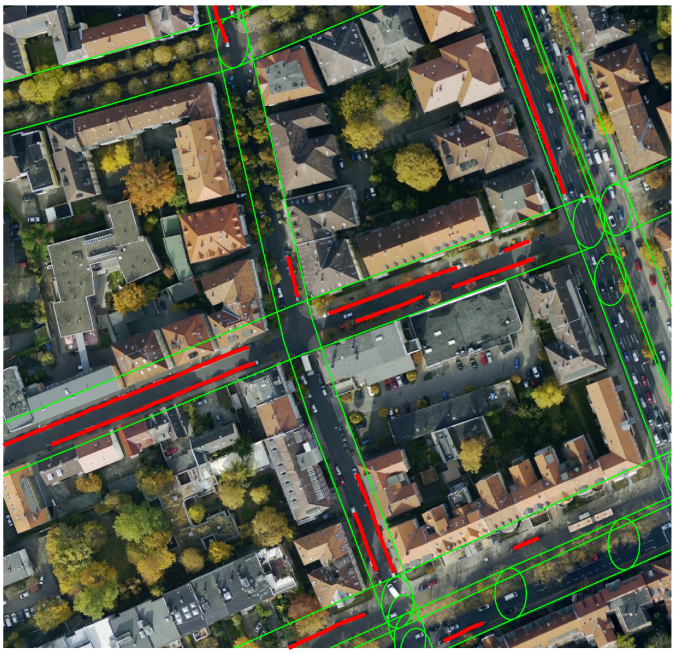


Figure 27. Extracted road side parking areas (red lines)

3.2.1 Definition of desired output

Drivers need manifold information when making a decision of where to park their vehicle. Obviously, the location of parking areas is the most essential data required. Within the AIPARK system, location information is retrieved by parking area extraction as described in section 3.1. Furthermore, drivers might need to know if a parking area is metered or if it is subject to restrictions. Parking areas may be restricted to certain user groups or time frames. An example for time-framed restrictions are operating hours of private car parks. User group restrictions limit parking area access only to certain demographics such as residents, staff or disabled persons.

The process described in the following sections is designed to extract all of this information from floating car data. Since it appears to be infeasible at the current point to identify specific pricing policies or opening hours in the given data, the desired output is simplified and formulated in binary form as follows:

- Metered vs. non-metered parking spot
- User group restricted vs. non-restricted
- Restricted opening hours vs. 24/7 opening hours

Additionally, output should include a measure to assess reliability of classification outputs, like a "confidence score". This is necessary, since the extracted information is going to be integrated with the AIPARK system and all components of the system should be aware of potentially limited data quality.

3.2.2 Data representation

Input data The used data set consists of smartphone floating car data contributed by more than four million active users. The data supplier's name must remain undisclosed due to contractual requirements between the AIPARK project and the supplier. Figure 28 visualizes the spatial data distribution. The scale of the figure is normalized between zero and one, because absolute numbers about the distribution of the data must not be published. FCD is contributed by frequent drivers at ages approximately ranging from 17 to 59. As of March 2017, the data set consists of about 50 million parking events from urban car rides across Germany. New data is added continuously.

Floating car data is originally provided in the form of vehicle trajectories including speed profiles. By analyzing these trajectories with respect to speed and location, two different types of parking events are extracted: "Successful parking events" are used to describe the events of drivers occupying a parking spot. "Leaving parking events" specify the points in time when vehicles are leaving their parking position. Figure 2 a.) provides an exemplary speed profile used for park event extraction. Implementation details of the algorithm for park event generation would exceed the scope of this thesis. For the purpose of this study, it is only

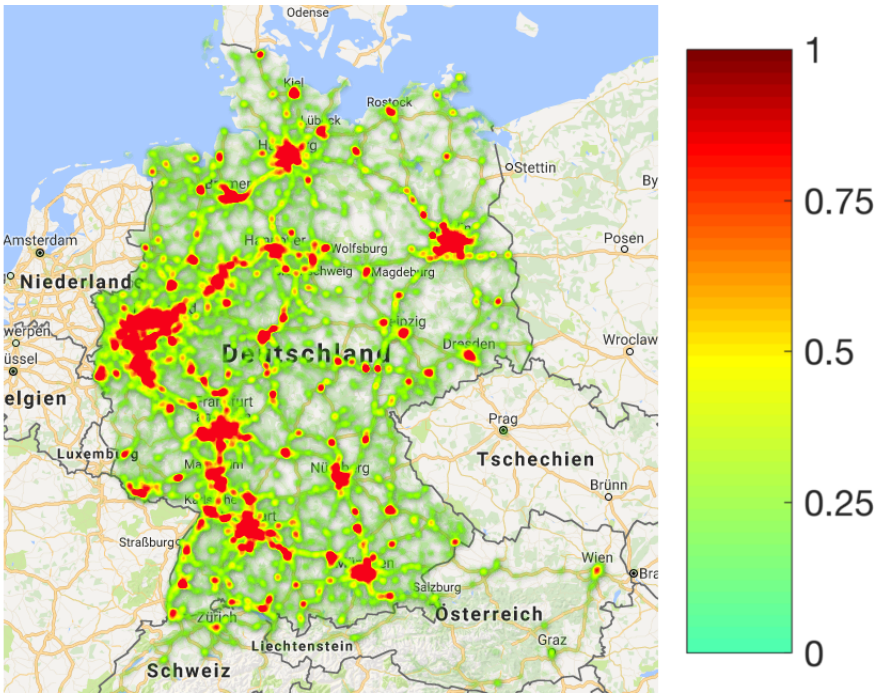


Figure 28. Normalized FCD distribution across Germany (vehicle trajectories)

necessary to understand the type and structure of the raw data that is used during meta information generation.

Table 6 provides the data model associated with every parking event for better understanding of parking events as floating car data representation. Parking events are mainly characterized by their geographic coordinates in latitude and longitude, the event type (successful or leaving) and a unix timestamp. Additionally the origin of the data point is stored, as well as an optionally assigned parking area ID in the AIPARK database in case the parking event can be mapped to a particular adjacent parking area.

Output data model To formally define the desired output of the entire parking area mapping process, the data scheme as shown in table 7 is introduced. It contains information about position, geometry and capacity which is retrieved from remote sensing imagery with the process described in the previous section.

Table 6. Floating car data model using parking events

Attribute	Data type
Data source	String
Parking area ID	Optional Integer 64bit
Geographic coordinates [Lat,Long]	[Float,Float]
Significance	Optional(Float)
Type of event	String
Unix timestamp	Integer 64bit

The name of the parking area is obtained by reverse geocoding the coordinates of the parking area to a readable address. All further metadata which describes the parking area is stored in boolean format with a corresponding confidence score. In future work, this data scheme may be extended, if more information can be retrieved from floating car data or other third party sources. Additional information could include specific opening hours, prices and flags for handicapped parking spots.

Table 7. Parking area metadata model

Attribute	Data type
Name	String
Capacity	Integer
Center point [Lat,Long]	[Float, Float]
Marking line [[Lat],[Long]]	[[Float],[Float]]
Metered parking space	Boolean
Metered parking space confidence	Float
Residential parking space	Boolean
Residential parking space confidence	Float
Opened 24/7	Boolean
Opened 24/7 confidence	Float

3.2.3 Preparation of training datasets

In order to train models that are able to classify parking area metadata, labeled training datasets are required. OpenStreetMap, as the only publicly available database that provides parking area data, is a suitable source for building labeled

training datasets. OSM data is not free from errors or inaccuracies, since it is contributed by voluntary users and not professionals. However, studies focusing on the quality of OSM data have shown that errors in OSM data are mainly due to inaccurate reflection of geographic shapes like coastlines or roads [98]. Other spatial information, such as parking area metadata, was previously proven to be accurate in terms of geographic position and currentness, although data completeness strongly depends on interests of individual map contributors [99]. OSM data quality is therefore considered to be sufficient for the purpose of creating a training dataset. Nonetheless the mentioned issues must be kept in mind, when assessing model results in subsequent steps.

With a parser that reads all parking areas from the OSM map, which contain any kind of meta information, a general baseline dataset is created. This general dataset contains 1.96 million parking areas. The raw data with the form as described in the previous section 3.2.2 can not be fed directly into any kind of machine learning model. Its structure is optimized for readability, high consistency and low storage volume in a MySQL database. Also the raw OSM data includes a large amount of entries that can not be used for different reasons ranging from insufficient data quality to irrelevant information. Hence, the training data sets are prepared in three steps, which are illustrated in figure 29. In a first step only parking area entries which contain at least some kind of metadata that is relevant to the problem are kept. In particular this means existence of information about user group restrictions, opening hours or cost. This way, the size of the raw data set is reduced from 1.96 million data points to about 100,000 entries.

Subsequently, parking events from the surrounding area are assigned to every parking space. For composing the main dataset, the “surrounding area” is set to a 1000 x 1000 m window around the center point of each parking area. This way,

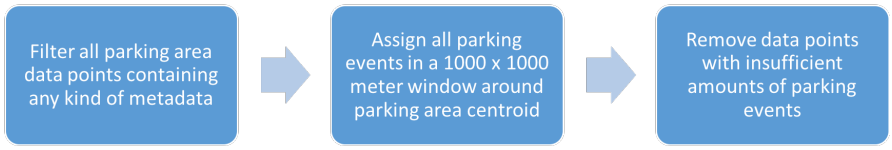


Figure 29. Data preprocessing

the amount of parking events for training may be adjusted at different levels very easily without re-generating the entire data set from scratch. In order to reduce random noise in the data, entries with insufficient amounts of parking events are reduced. If the floating car data source does not provide enough data for certain regions, parking events will not form characteristic patterns, but rather represent only noise. Parking areas with a parking event density lower than 50 events per square kilometer are therefore removed. Next, the training data sets are labeled according to the specification given in section 3.2.2. For this purpose, the main data set is split up in three separate data sets, one for each parking area attribute: cost, user group restrictions and opening hours. OSM already provides a binary attribute called "metered" which is used as a binary class label with values "YES" and "NO". Similarly, the "residential data set" is also labeled using the equivalent OSM attribute. Only the "opening hour data set" needs to be labeled differently. Since opening hours of parking areas are subject to complex rule sets in everyday life, OSM provides only a text field for contributors to enter information about opening hours. The resulting potential for redundant information needs to be considered when assigning class labels to the training data. If a parking area is opened 24/7, OSM contributors are using multiple phrases like "24/7", "00:00-24:00" or "Mo-Su" to express the same kind of information. The variety of all these different phrases is compressed into a binary attribute by merging all phrases for "24/7" in one class and all other types of opening hours in another class.

In a last step, before feature engineering may start, the resulting data sets need to get balanced. Balancing a data set means to ensure that all class labels are represented in equal parts. Failing to balance a dataset before using it for training will result in distorted results, which do not reflect actual classifier performance. For binary classification problems, like in this case, the most simplistic way of balancing is to randomly remove elements from the larger class until both classes are represented equally. As table 8 indicates, the cost data set is reduced to about 23,000 data points due to the fact that the main data set includes many more free than metered spots. Similarly, the opening hour data set is reduced to only 4,344 data points because OSM does not provide enough data for opening hours other than "24/7". Lastly, the residential data set is treated the same way to account for the fact that the data set holds more public than residential parking spots.

Table 8. Class size and distribution before and after balancing

	Unbalanced (# class occurrence)	Balanced # data points
cost data set	52231 (11716 / 40515, "Yes" / "No")	23432
opening hour data set	8490 (2172 / 6318, "Not 24/7" / "24/7")	4344
residential data set	56819 (13123 / 43696, "Yes" / "No")	26246

Finally, it is important to ensure homogeneous spatial data distribution to avoid the model from overfitting with respect to conditions that only occur at a few locations instead of the entire area of interest. Since the training data is crowdsourced, homogeneous spatial distribution must not be taken for granted. A few core contributors, which collect a lot of data in their city could for instance distort the spatial data distribution in undesired ways. However, this is not an issue in this case: As shown on the example for the opening hour data set in figure 30, training data is randomly distributed across urban areas in Germany.

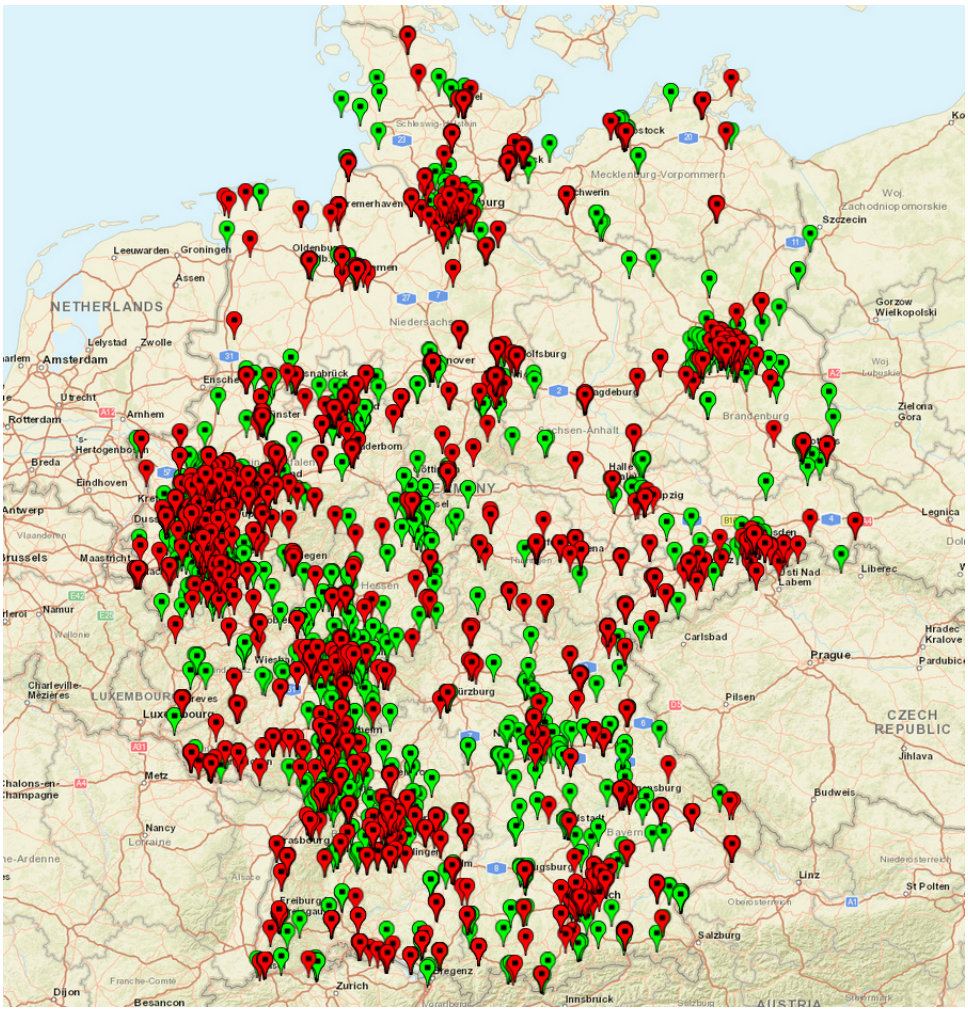


Figure 30. Spatial distribution of opening hour training data across Germany (green: “24/7”, red: “Not 24/7”)

3.2.4 Feature engineering

The step of transforming data points into feature vectors, which are required as classifier input, is called “feature engineering” or “feature preparation” in the CRISP-DM. According to the definition given in section 2.3.1, features are designed and selected such that they describe their corresponding object class optimally. The widely-used checklist for feature engineering and selection by Guyon and Elisseeff is also applied in this case [109]: As suggested in their paper, use of domain knowledge is preferred over the method of brute-forcing the problem by

trying vast amounts of standard features. Hence, features are manually designed based on the following hypotheses:

1. Parking areas with different metadata attributes (cost, restrictions, opening hours) may be distinguished by distinct characteristics in the temporal parking event distribution

2. In combination with temporal parking event distribution may parking areas be distinguished by considering other metadata, like their vehicle capacity for instance

3. Local demographics do potentially have an impact on parking area characteristics (e.g. income, no. of cars per 1000 citizens,...)

4. "Parking attractiveness" of certain locations and parking restrictions therefore depend on the spatial density of points of interests (e.g. shopping center, cinema, university, etc.)

5. Parking attractiveness of certain locations also depends on the individual degree of attractiveness, which is associated to different points of interests (e.g. soccer stadium vs. golf course)

Following the first hypothesis, parking events for every parking area in the training data set are aggregated in various ways depending on their time stamp in order to model temporal occurrence. Figure 31 presents four different parking event aggregations, which may be used as input features for classification. The entire training data was used to create all four graphs. The graph in the upper left corner compares the mean absolute number of parking events per hour per class. The two classes are different from each other in terms of absolute numbers, because mean number of parking events for metered parking areas is generally higher than

for free parking spots. This is a first and important finding, however, floating car data and therefore parking events are spatially not distributed equally. A parking area could be metered, but because the floating car data source is underrepresented at this particular location, a classifier based solely on this feature would classify this parking area as free. The graph in the upper right corner compares the two parking event distribution after normalization to avoid this issue. This feature reveals an interesting trend: Free parking areas frequented earlier than metered parking areas, especially in the morning. At around 6 a.m. the trend begins to reverse and metered parking areas are in demand more than free parking spots. This phenomenon could either be explained by the fact that drivers prefer free over metered parking areas. Another explanation could be given by people frequenting city centers (with mostly metered parking areas) typically throughout the day, and especially before noon, but not at night.

The first derivative of park event occurrence, which expresses the delta of parking events is another possible feature, which can be used for classification. But parking events do not necessarily need to be aggregated in 24 hour "bins". Other aggregations could be at finer resolution in 48 bins or across days instead of hours, like in the case of the graph in the lower right corner. In this particular case the standard deviation of park events per day is used to characterize both classes. While free parking spots in general indicate less variability throughout the entire week, metered parking areas show higher standard deviation. Higher standard deviation in the number of parking events means higher turn-over of metered parking areas, which is the purpose of actively managing parking facilities. Standard deviation of both classes slightly increases on Saturdays and Sundays. The reason for this trend could be caused by car owners, who commute using public transport during the week and use the car for recreational activities during

weekends.

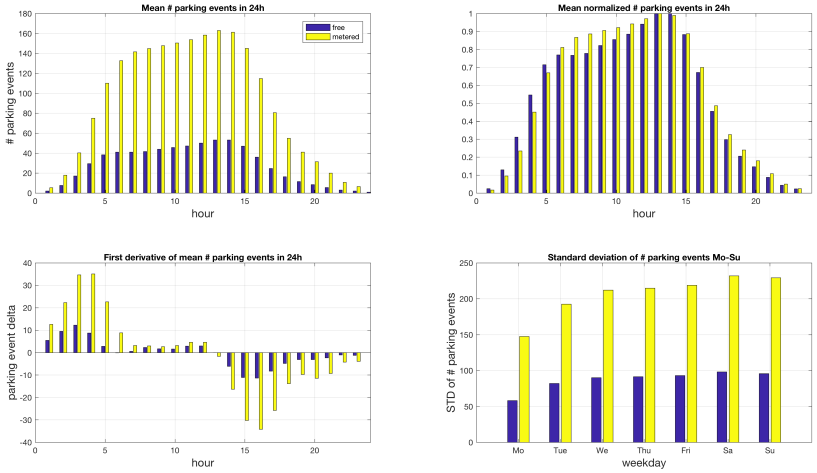


Figure 31. Exemplary feature representations

These four features are just a small share of all possible types of data aggregations. Other features look at successful and leaving parking events separately, consider their derivatives or summary statistics. Table 9 provides an overview about all parking event features, which are used for meta data classification. These features are not only used for the cost classification problem, but accordingly also for the residential and opening hour classification tasks.

Table 9. Parking event features

Feature type	Dimensions	Feature type	Dimensions
24 bins	24	diff 24 success bins	23
48 bins	48	diff 48 success bins	47
weekday bins	7	diff 24 leaving bins	23
24 success bins	24	diff 48 leaving bins	47
48 success bins	48	std 24 bins	1
24 leaving bins	24	std 48 bins	1
48 leaving bins	48	std weekday bins	1
diff 24 bins	23	std 24 success bins	1
diff 48 bins	47	std 48 success bins	1
diff weekday bins	6	std 24 leaving bins	1
		std 48 leaving bins	1

All 21 features listed in table 9 sum up to a total of 446 feature dimensions. All features are concatenated into one single feature vector according to equation 17.

$$\vec{F}_{class_n} = [\vec{f}_1, \vec{f}_2, ..., \vec{f}_n]; \tag{17}$$

Figure 32 illustrates the averaged first 220 elements per class for the cost classification problem, as they could be given to any kind of classification model. Features for residential or opening hour classification are treated accordingly.

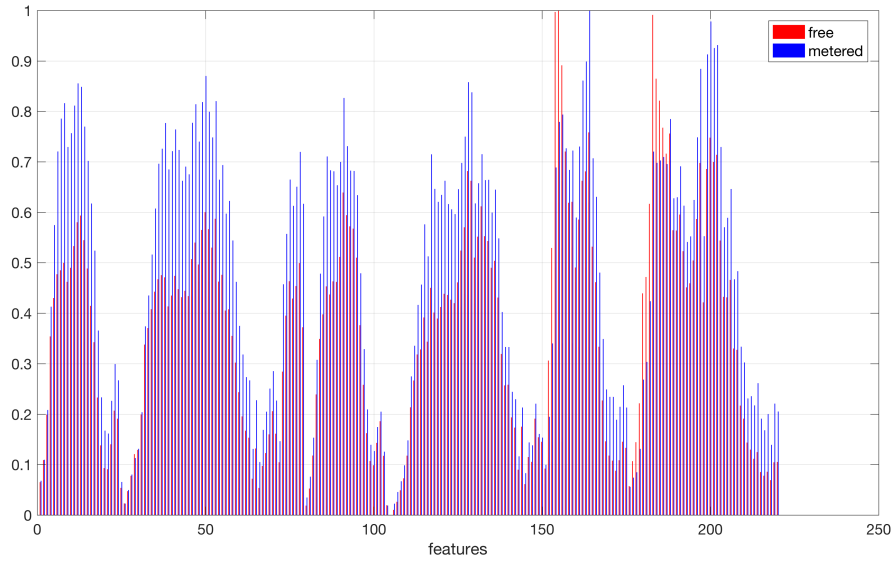


Figure 32. First 220 of 446 feature elements of free vs. metered parking areas

3.2.5 Data enrichment

In addition to the parking event features described in the previous section, the dataset is enriched with several more information to describe the surrounding spatial context with respect to parking behavior. It is assumed that demographics may influence parking occupancy. The "popularity" of a location is assumed to be another important factor likely to be influential on parking behavior. Lastly,

it is assumed that the type of parking area and capacity influence its own parking attractiveness. In total, 14 of those context features are appended to the feature vector. Table 10 provides a summary of these features and their dimensions.

Table 10. Spatial context features

Feature type	Dimensions	Feature type	Dimensions
capacity	1	shareoOfAge45To65	1
parkingAreaType	5	shareOfAge65Onwards	1
perCapitaIncome	1	inhabitantsPerSquareKilometer	1
carsPerThousandCitizens	1	fanCount	1
unemploymentPercentage	1	checkinCount	1
shareOfCityAreaOnTotalArea	1	ratingCount	1
bipPerCitizen	1	talkingAboutCount	1

Capacity and parking area type are features to provide context about the investigated parking area. The estimated parking area capacity is retrieved using the previously described mapping process. More information about the parking area context is provided using a "type" attribute. Depending on the geographic position of the parking area, one of these five values is assigned: On-street, Off-street public, Off-street private, residential neighborhood or urban neighborhood.

Demographic data in multiple dimensions is retrieved from the German Federal Bureau of Statistics, which conducts a nationwide census on a regular basis [100]. Among all available demographic information, vehicles per 1000 citizens, per capita income, population density, age distribution, unemployment rate and level of urbanization were chosen. In terms of spatial resolution, the data was collected per ZIP code area. Demographic data is mapped to particular parking areas by determining the corresponding ZIP code. Data is added to the existing feature vector using min-max-normalization.

To describe popularity, a point-of-interest (POI) dataset provided by Facebook is utilized. Points of interest may be any kind of public place, that attracts people, like cafes, clubs, theaters, monuments and so forth. The dataset includes information about "likes" and the number of check-ins from more than 1.4 million

POIs in Germany. Furthermore, it provides indicators of how often a place has been rated or talked about on Facebook [101].

Popularity is mapped to parking areas based on a cumulative approach: It is assumed that popular places within walking distance of a parking area have impact on its occupancy and therefore also on potential restrictions. To obtain the popularity score per parking area, popularity measures of all POIs in a 500 m x 500 m bounding frame, which is here considered being an acceptable walking distance, are accumulated.

3.2.6 Model

As indicated by the figures 31 and 32 in section 3.2.4, the investigated problems are not linearly separable. In order to account for this circumstance, a non-linear kind of model is required. Classification models potentially capable of solving the given tasks are decision trees, logistic regression, support vector machines with non-linear kernels, k-nearest-neighbor classifiers, ensemble learners like bagged trees or artificial neural networks. Each of these models have their own parameters, properties, requirements, advantages and disadvantages, which are well-researched [102]. In order to select the best models for the given classification problems, all of the mentioned classifiers are trained on the data sets by using model parameters as recommended in the respective papers. The decision of which model should be chosen is made based on the generalized classification error as stated in equation 16. The generalized classification error is determined using the method of 5-fold cross-validation as already described in section 3.1.5.

As section 4.2.2 in the next chapter will reveal, support vector machines with a quadratic kernel perform best on the given data sets. To provide a confidence estimate for every classification made, the SVM classification score by [97] and explained in section 3.1.5 in equation 15 is therefore applied. It is important

to have an estimate about the reliability of a certain classification result when integrating the results of the proposed mapping process into the AIPARK parking area database. The classification score helps to determine exactly which results can be presented to the user or client and which ones need further improvement.

CHAPTER 4

Results and Critical Discussion

This chapter will present and discuss implementation results for all significant process steps described in the previous chapter. First, parking area detection results are presented beginning with image preprocessing, adoption of computer vision methods for tree, road and vehicle detection, and finally, the evaluation of road side parking area line extraction. Subsequently, results and limitations of metainformation mining in floating car data are stated.

4.1 Parking area detection results

4.1.1 Image preprocessing

Image preprocessing includes the steps of superpixel segmentation and road network mapping using GIS data. Image segmentation merges conjugated image parts and treats them as single pixels to reduce the number of required operations per image. Implementing the SLIC clustering algorithm for superpixel segmentation by [46] is very straight-forward, as SLIC only provides three parameters for adjustments, "superpixel compactness", number of superpixels and number of clustering iterations. The number of superpixels is chosen such that each pixel is approximately the size of a vehicle. The "compactness" parameter controls the ability of each pixel to adjust to local shapes. A higher compactness value means more square-like superpixels, while a value close to zero allows adoption of structures in the image like edges or corners. Since SLIC uses gradient ascend methods, results improve with each iteration until convergence is reached. The goal is to have the superpixel centroid approximately match the center point of each vehicle in order to draw a detection window around it. As compactness and iteration parameters were found to have no substantial effect on the detection of rectan-

gular vehicles, both are set to a value of 10 as recommended by the authors [46]. The number of superpixels has a stronger effect on subsequent processes. If the number of superpixels is chosen too low, vehicles will be merged in a few large pixels, while a high number of pixels increases necessary processing operations. Figure 33 illustrates the effect of a low, suitable and high amount of superpixels on the segmentation outcome. To obtain car-sized superpixels in the test image, the maximum pixel number is set to 25,000, which ensures that no vehicle will be lost, while keeping computational effort low.

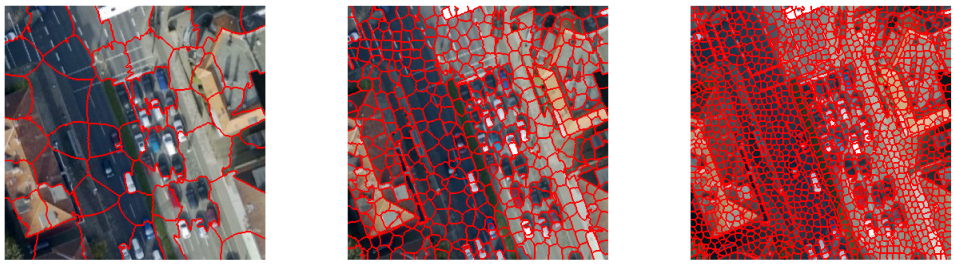


Figure 33. Image segmentation with different numbers of superpixels (from left to right: 50, 500, 2000 superpixels)

While image segmentation is implemented mostly relying on the previous work of Achanta et al., mapping the road network to the image at an appropriate level of accuracy involves several difficulties caused by irregularities in the used GIS data [46]. In the OpenStreetMap XML-File, roads are only defined as a set of connected vertices and optionally also a road type (e.g. motorway or residential street). Although a standard width is associated to those road types, actual road widths may differ from this default value. An additional source of inconsistency in OpenStreetMap data is given by the fact that road types are chosen by individual OSM contributors which may assign unsuitable road categories and therefore the wrong road width. This is problematic since road side vehicles need to be detected.

If the road width given by OSM is smaller than the actual road, road side vehicles will get cut off, but if the width is set too high, irrelevant image parts like buildings are added to the detection region. This increases the chance of false positive vehicle detection, if the detector erroneously classifies objects like roofs or windows as vehicles. Figure 34 a) provides an examples where the residential street in the west-east direction is falsely assigned a highway road type resulting in the unnecessary inclusion of irrelevant image contents.

(a) Wrong road width (b) Misplaced road center line

Figure 34. Inaccuracies in OSM road network

Another problem with OSM road data is grounded in the fact that road center lines are registered manually by contributors on the map. This bears the risk of misplaced center lines, which could lead to inclusion of irrelevant contents on one road side, while image regions that potentially contain parking vehicles on the other side are discarded. The north-south street on Figure 34 a) and b) illustrate the case where the west side of the street almost gets cut off, while the east side exposes too much irrelevant area. Although in the test image only three out of 21 road sections have the wrong category assigned and only one road has a notably

misplaced center line, these issues could turn into a serious source of error in a large scale system. However, with the information provided by OSM it is not possible to correct the described inaccuracies. To avoid these issues, it is recommended to use automatically generated, more accurate road maps from commercial GIS suppliers like Google Earth, Here (formerly known as NavTeq) or Esri [103, 104, 105]. Within the scope of this study, a generous margin of an additional five meters is added at each road side to eliminate the chance of excluding parking vehicles from the detection area. However, this increases the chance of false positives as this margin includes walls and roofs and buildings in some image regions. Hinz et al. also commented on GIS accuracy issues in their article and recommended using NavTeq data [82]. [81] and [84] even only "simulated" GIS data in their work by manually drawing road outlines on their imagery.

4.1.2 Object detection results

Within this study, object detection techniques are used to identify trees, roads and vehicles on remote sensing imagery. Since very similar methods are applied to all three cases, implementation results are summarized into one section. Regardless of object class, detections are made by iterating over all remaining superpixels after completion of GIS filtering. For each superpixel, a detection window of fixed size is drawn around the pixels' centroid, such that it includes the entire superpixel. Next, the detection window content is classified based on the corresponding feature map. Result quality is strongly influenced by the type of chosen feature map and classifier.

Vehicle classification

The chosen vehicle detection method based on AlexNet convolutional neural network features and a quadratic support vector machine is evaluated by com-

paring it against six other approaches suggested in literature. Those methods are color histograms with linear SVM, histograms of oriented gradients with SVM, color and HOG combined with SVM, AlexNet features together with a linear and quadratic SVM classifier and lastly, features from the VGG 16 convolutional neural network by [106] in combination with linear and quadratic SVM. For comparison, the generalized error given in equation 16 and the precision-recall graph is used. Precision is the fraction of detected instances, which are relevant, while recall is the fraction of relevant instances that are detected. Figure 35 provides a visual description of this concept.

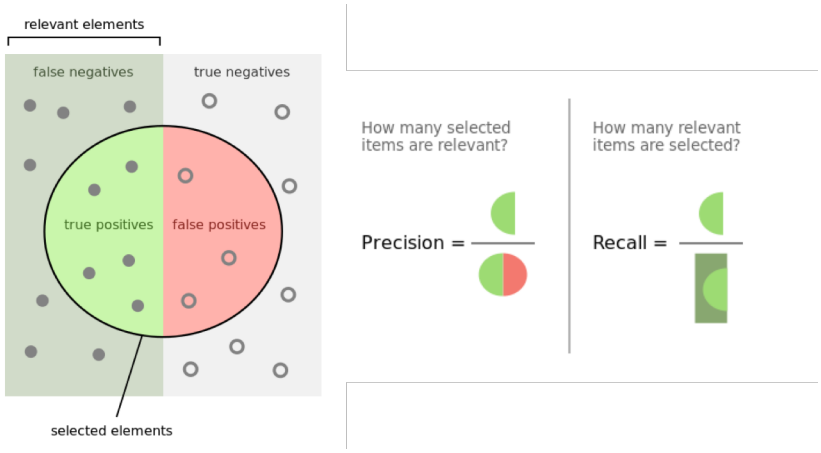


Figure 35. Precision vs. recall [107]

During the process of reviewing the related literature it was found that it is difficult to compare methods from individual papers among each other. Different performance measures and training datasets complicate objective comparisons. To overcome this issue, the aforementioned most promising feature and classifier combinations are adopted and trained on the dataset described in section 4.1. The generalized classification accuracy of all detection methods is determined using 5-fold cross validation. As shown in table 11, the combination of VGG16 features with a quadratic support vector machine outperforms especially color and structure based classifiers significantly in terms of accuracy.

Table 11. Generalized classification accuracy of different detection methods on training data set

Detection method	Generalized accuracy [%]	Average processing time [sec.]
Color + lin. SVM	76.3	0.2558
HOG + lin. SVM	94.4	0.1748
Color/HOG + lin. SVM	96.2	0.2281
AlexNet + lin. SVM	98.4	0.9062
AlexNet + quad. SVM	99.1	0.9682
VGG16 + lin. SVM	99.6	19.9877
VGG16 + quad. SVM	99.8	20.8884

The precision-recall graph presented in figure 36 provides more insights about the individual performance of each method tested. As the graph and table 11 indicate, VGG16 features with a quadratic SVM performed best on the given training data set. It must be noted that the given accuracy values are generally about 3-4 percent above the comparison results from other papers, such as [89], [108] or [92], even for "standard algorithms" such as linear SVM + HOG. This bias is due to the used trained dataset and chosen validation method: First, it is possible that the dataset used in this study provides less variance than other datasets. As illustrated in figure 24, the background class of the dataset contains a fairly high amount of samples with "green colored content", like grass, bushes or trees, while the vehicle class is colored completely differently. It is therefore possible that color sensitive classifiers tend to overfit with respect to this characteristic of the dataset. However, this still does not explain higher accuracy of classifiers like linear SVM + HOG. Another reason could be the method for determining accuracy values. While other authors, like Li et al. estimated the accuracy of their classifier by testing it only once on an entirely separate dataset, 5-fold cross-validation is used in this study to determine the generalized classification accuracy [92]. In the future, the background class of the used training dataset should be enhanced by specifically adding "hard negatives" like street background, roofs or small non-car objects to

the object class.

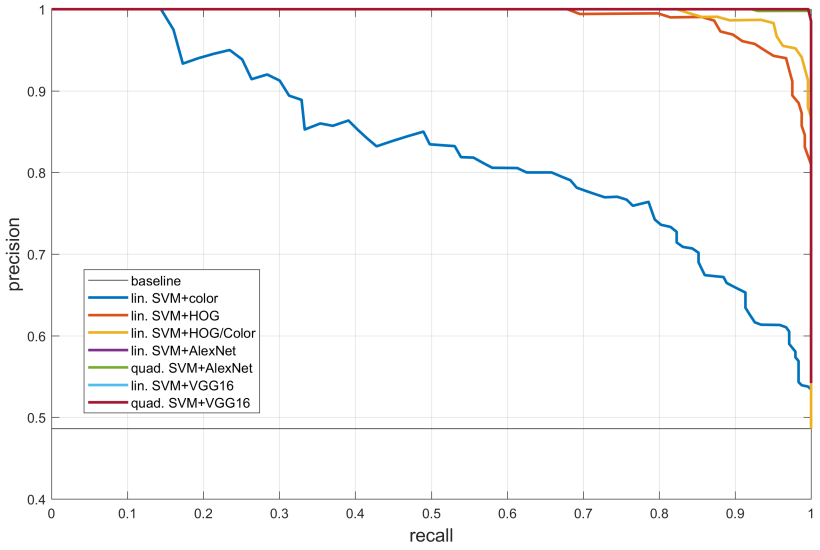


Figure 36. Precision-recall graph for different vehicle detection methods on test data set

Unfortunately, the result of the convolutional neural network proposed by Li et al., which is specifically trained for vehicle detection on remote sensing imagery is not included in the comparison, due to insufficient development hardware used in this study [92]. However, their denoizing-based CNN (DCNN) shows with 96.21 percent accuracy very similar to the quadratic SVM with VGG16 features (even after considering the mentioned dataset bias).

Although classification accuracy with VGG16 features is very satisfying, computational cost is another very important performance indicator. Due to its very deep structure, the VGG16 CNN is relatively slow compared to other methods for feature map generation. To measure the average detection speed, each feature extraction method and classifier was executed and measured on a sample image in 100 iterations. Table 11 provides the average time required for generating the feature map from the sample image and to classify it. While histogram based feature maps are computed really fast, CNN features, especially from VGG16 take much

longer to compute. If classification time is visualized in proportion to classification accuracy, CNN features are found to not justify their longer computing times relative to the gained accuracy. Also, quadratic SVM classifiers are observed as being slightly slower (but also more accurate) than linear SVMs on the same type of feature map.

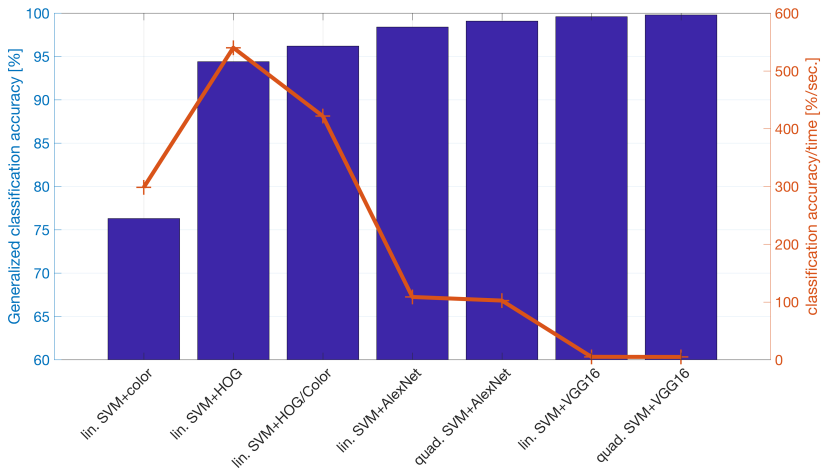


Figure 37. Accuracy vs. classification speed of different detection methods

Figure 37 plots the classification accuracy versus the accuracy per time ratio. As pointed out by this graph, VGG16 features are quite inefficient as they only contribute an accuracy increase of only 0.7 percent while requiring more than 20 times as much computational effort as AlexNet features with quadratic SVM. It has to be mentioned, that feature extraction and classification are normally executed much faster than the numbers in table 11 indicate. To facilitate speed measurements and exclude more complex data handling operations, features were extracted and classified on a "one-by-one"-basis during each iteration. In a real application, feature extraction would be completed first and then be passed over to the classifier, which would process feature vectors much faster in batches. However, the relative computational effort among the different methods remains the same.

The chosen vehicle classification method has to be executed for every super-

pixel in the test image, which means several thousand computations. Since VGG16 feature are too expensive to generate for every detection window, AlexNet features are chosen as they represent an arguable trade-off between speed and accuracy.

4.1.3 Tree and road classification

When deciding for color/HOG features with a linear SVM for tree and road detection, the same process for comparing different solutions was applied as to the vehicle detection problem. Table 12 contains the most relevant accuracy results for both tree and road detection results. Equivalent to vehicle classification, CNNs perform slightly better than color/HOG features. However, since tree and road detection are only intermediate pruning tasks responsible for decreasing computational effort in later stages, there is no need for maximal accuracy. In this case, fast processing is much more valuable, which is why based on the findings in table 12 linear support vector machines with color/HOG features are chosen to detect tree and road elements. To incorporate the chance of false positives during those process stages, only superpixels with a positive SVM score ≥ 0.9 are removed from the test image (see SVM-score defined in equation 15). This prevents the algorithm from accidentally removing superpixels, which actually contain vehicles, during preparatory pruning stages.

Table 12. Generalized accuracy of tree and road classifiers

Detection method	Generalized accuracy "tree" [%]	Generalized accuracy "road" [%]
Color + lin. SVM	97.5	97.0
Color/HOG + quad. SVM	94.4	96.5
AlexNet CNN + lin. SVM	99.1	97.0
AlexNet CNN + quad. SVM	99.1	98.5

Comparing detection results against reference data

Finally, overall performance of vehicle detection is compared against manually generated control data (ground truth) on the test image. For this purpose, all 603

vehicles on the image were labeled with a bounding box. Then, the outlined vehicle detection algorithm including superpixel segmentation, GIS filtering, tree and ground removal is applied to the image. A detection window is counted as true positive, if it's overlap with the ground truth is ≥ 50 percent. Ground truth bounding boxes with overlap smaller than 50 percent are counted as false negative. Detection windows with overlap less than 50 percent are counted as false positive detections. All remaining detection windows are true negatives. Figure 38 provides a closeup view the detection result. Green bounding boxes represent true positive detections, while red boxes indicate either false positive or false negative detection errors. As the image illustrates, detections are already very promising, although the algorithm has some difficulties with dark vehicles in shaded areas and certain parts of buildings.

Figure 38. Vehicle detection accuracy (true positives: green, false positives and negatives: red)

The confusion matrix in table 13 provides more details on the detection result. With 565 true positives and 2,921 true negative classifications, an overall vehicle detection accuracy of 97.48 percent is obtained. The slightly higher number of false negatives compared to false positives indicates that the classifier still struggles to separate dark vehicles from dark backgrounds, like streets which are not exposed to sunlight.

Table 13. Confusion matrix: vehicle detection

		predicted class	
	n = 3576	vehicle	background
true class	vehicle	565 (TP)	51 (FN)
	background	39 (FP)	2921 (TN)

4.1.4 Evaluation of roadside parking area line extraction

In order to extract parking areas from the test image, detected vehicle positions are merged into connected lines of parking vehicles. These lines can then be used as parking area representation. This is achieved by applying the algorithm described in section 3.1.6 to the set of detected vehicles. The results of the line extraction algorithm are evaluated by conducting a sensitivity analysis for its parameters and comparing the estimated capacity against the manually collected validation data in the test area.

Sensitivity analysis of algorithm parameters The recursive algorithm for on-street parking area line extraction is controlled by three parameters: The maximum distance between two nodes determines how close adjacent vehicles are allowed to be in order to be considered as part of a parking area line. If the distance is too small the algorithm will not be able to deal with parking areas that are only partly occupied by vehicles on the analyzed image. However, if the distance is set

too high, parking areas in different road sections that do not belong together might be merged into one single area. The second parameter is the maximum "line angle" formed by two adjacent edges on the parking area line. A very straight parking area line with a very small line angle can not accommodate for slight inaccuracies of the detected vehicle centroid position or natural road curvature. But if the maximum line angle is set too high, the chance for inadvertent inclusion of traveling vehicles, which have not been removed in previous steps, into the parking area line, increases. This is crucial, as the maximum line angle acts during line extraction as a barrier to prevent moving vehicles to lines of parking areas. The third and last parameter is the "road alignment angle", which acts like the line angle. The road alignment angle defines the maximally allowed deviation of a parking area line segment with respect to the general road direction. Thus, it is a measure of how parallel road and parking area line are allowed to be. Again, if the road alignment angle is set too low, it prevents the algorithm from adjusting lines to given local characteristics like road curvature. On the other hand, a too large road alignment angle will cause the algorithm to form parking area lines from nodes that are not adjacent, like vehicles on opposite sides of a street. In this case, the extracted parking area lines would not reflect the given parking area positions.

Figure 39. Errors in vehicle line detection: Missed parking spot between vehicles (left), merging of parking across road section (middle), merging of parking areas on different sides of the road (right)

To obtain an acceptable range of values for each of those parameters, a sen-

sitivity analysis is conducted. Each parameter is tested on seven different levels. Line and road alignment angles are tested in integers from three up to nine degrees. The maximum distance between two nodes is altered in a range from five to eleven meters in integers. To be able to consider potential interactions as well, the analysis was set up as a full factorial design covering any potential parameter combination. Sensitivity is assessed by monitoring changes in the number of parking spots which are included in one parking area (capacity) and visually by checking visualizations of parking areas on the test image for incorrect parking area line geometries. Monitoring changes in estimated parking area capacity allows to discover if a parameter combination makes the algorithm too restrictive. Visual assessment is a simple yet effective way to determine if certain parameter values are not strict enough to generate accurate results.

As sensitive analysis revealed, the algorithm performs robust with respect to changes in the angle between vehicle queue and street, but is sensitive to changes in vehicle angle and distance. The algorithm was found to perform best with an angle of six degrees between vehicle line and main road heading. Furthermore, a maximum distance of 12 meters between individual vehicles on the line lead to good algorithmic results. Lastly, a maximum angle of six degrees between vehicles in the line turned out to be a good trade-off between "line flexibility" and erroneous inclusion of vehicles that are moving or parked on the other side of the street.

Influence of false negatives on capacity estimation To evaluate how accurate the proposed algorithm reflects real parking spot capacity per street, the actual number of parking spots per street is compared to the detection results. The manually collected validation dataset indicates a total of 1,094 parking spots in the entire depicted area of the test image. In order to determine how close the estimated result matches the real capacity, the mean absolute percentage error

(MAPE) presented in equation 18 is computed.

$$M = \frac{100}{n} \sum_{t=1}^{n} \left\| \frac{actual\ value - estimated\ value}{actual\ value} \right\| \tag{18}$$

The MAPE for capacity of all 21 road sections in the test area is about 49 percent. A closer look on the difference between real and estimated value reveals that the estimated capacity of six road sections is zero. These roads are all covered by trees making any information extraction impossible. Recomputing the mean average percentage error by only looking at image regions that are not entirely covered by trees leads to a new value of 21 percent. The issue of false negative detections due to trees covering parking areas may only be overcome in future by analyzing images from autumn and winter months, where the ground visibility is higher.

4.2 Metadata extraction results

The geographic parking area position and capacity estimate are complemented by metadata about cost, user group restrictions and opening hour information. The required information is retrieved from floating car data by applying the process as described in section 3.2. This section will identify relevant features, investigate accuracy of the proposed method and discuss it's sources of error.

4.2.1 Feature importance

In section 3.2.4 were 32 features created with a total array length of 464 floating point values. For a better understanding of the given problem it is necessary to know how every one of these features affects classifier performance [109]. This information is obtained by repeatedly training the model, while leaving one feature out each time. The difference in generalized classification accuracy after 5-fold cross-validation between this run and a baseline run with all features can then be

considered the individual influence of the left out feature. Although this approach is simple and easy to apply, it is unable to detect potential feature interactions. However, applying a full factorial test design, which means systemically leaving out all possible feature combinations is also infeasible due to the long training and cross-validation times. As a trade-off between training time and the risk for missing interaction effects, only two-feature-interactions are included in the test design. Following the sparsity-of-effects principle, using a fractional factorial-like test design like this maximizes the chance for identifying dominating features or feature-combinations [110]. The number of necessary training operations n_o can be expressed the following way:

$$n_o = n + \frac{n!}{(k!(n-k)!)} \tag{19}$$

where n is the number of features and k the desired order of feature interaction. In this case with 32 features and two-feature interaction, a total of 528 training iterations is required. The procedure is repeated for all three classification problems (cost, user group restriction and opening hours). Table 14 presents the five most significant features per problem. Interactions among individual features did not turn out to be important at all. The most relevant features in all three classification problems are quite similar. The first derivative of parking event occurrence appears to be very informative with respect to all three problems. The type of parking area is also quite influencing on the classification outcome of the residential and opening hour dataset. Lastly, points of interest as features have strong impact on classification performance. Furthermore, it can be observed, that accuracy contribution in the opening problem in general is relatively low compared to the other problems. While the type of parking area as feature contributes is relatively unimportant for cost classification, it has strong impact in the residential and

opening hour problems. Low resolution context features about demographics have not been found significantly beneficial to the problem.

Table 14. Accuracy contribution of most important features per classifier

	Cost		Residential		Opening hours	
Rank	Feature	Acc. cont.	Feature	Acc. cont.	Feature	Acc. cont.
1	diffLeaving48Bins	0.1722	parkingAreaType	0.3966	parkingAreaType	0.120
2	diffWeekDayBins	0.1702	POIs	0.1194	POIs	0.058
3	capacity	0.1702	diffLeaving24Bins	0.1035	weekDay	0.054
4	diffBins48	0.1698	success24Bins	0.0994	diffSuccess24Bins	0.053
5	diffSuccess48Bins	0.1691	bins48	0.0901	leaving24Bins	0.053

4.2.2 Accuracy evaluation

Determing the best model In order to determine the models for each data set, a number of different models is trained and tested on each of the data sets. The test considers a total of 24 classification models including decision trees at different configurations, logistic regression for classification, support vector machines with different kernels, k-nearest neighbor classifiers, ensemble learners and artificial neural networks with different numbers of hidden layers.

All models are validated using 5-fold cross-validation in order to avoid overfitting effects. The generalized classification accuracy as performance indicator is determined as $1 -$ *generalized classification error*, which is given in equation 16. Table 15 presents the results of this model screening for each of the three classification problems.

While the general performance of all models differs among the datasets, the support vector machine using a quadratic kernel function slightly outperformed the other models on all datasets. While the quadratic SVM only performs 0.1 percent better than the second best model on the cost data set, the percentage difference is larger on the residential and opening hour datasets. On the residential dataset, the quadratic SVM performance is 0.7 percent better than the second best model, a boosted trees model. The highest percentage difference was found in the opening

Table 15. Performance of quadratic SVM vs. other models

	Generalized classication accuracy		
Model	Cost data set	Residential data set	Opening hour data set
Complex Tree	76.6%	94.2%	56.5%
Medium Tree	76.1%	94.2%	58.4%
Simple Tree	74.6%	94.0%	58.0%
Logistic Regression	77.6%	93.6%	57.4%
Linear SVM	76.5%	93.4%	58.3%
Quadratic SVM	**79.8%**	**95.3%**	**62.9%**
Cubic SVM	78.9%	93.0%	55.6%
Fine Gaussian SVM	75.2%	84.2%	57.5%
Medium Gaussian SVM	79.7%	93.9%	58.4%
Coarse Gaussian SVM	74.4%	93.3%	55.7%
Fine KNN	76.0%	87.1%	54.2%
Medium KNN	73.3%	89.0%	57.2%
Coarse KNN	70.4%	90.1%	55.7%
Cosine KNN	73.6%	86.4%	56.8%
Cubic KNN	70.8%	88.0%	56.4%
Weighted KNN	76.5%	89.9%	57.7%
Boosted Trees	78.5%	94.6%	59.5%
Bagged Trees	78.9%	93.5%	56.2%
Subspace Discriminant	75.0%	93.5%	57.6%
Subspace KNN	78.4%	65.6%	53.7%
RUSBoosted Trees	76.5%	94.2%	58.2%
10 hidden layer ANN	74.9%	93.1%	56.9%
5 hidden layer ANN	76.1%	94.2%	54.1%
3 hidden layer ANN	76.3%	92.8%	61.3%

hour dataset. Although the general accuracy level is still quite low compared to the other datasets, the quadratic SVM performed 2.6 percent better than the second best model, which is an ANN with three hidden layers. Looking at the different model types, it can be observed that SVMs and ensemble learners perform similarly well, followed by medium sized artificial neural networks.

If the quadratic SVMs of each classification problem are compared directly to each other, different levels of performance are observed between the individual data sets. It appears that the combination of the used training data and classification model works very well for user group restriction (residential-only vs. public). Free and metered parking areas may be distinguished at an accuracy level of about 80 percent. Only the opening hour classifier performs at an unsatisfactory level of about 63 percent, which is only marginally better than a random guesser in the case of a balanced dataset like the used one. Figure 40 visualizes each classifier's performance using the Receiver-Operator-Characteristic chart (ROC). As

illustrated in the chart by the area under each chart are performance differences between the datasets not randomly caused, but rather from systematic error. The reasons for the remaining error, especially in the opening hour, but also the cost dataset, are discussed in the next chapter.

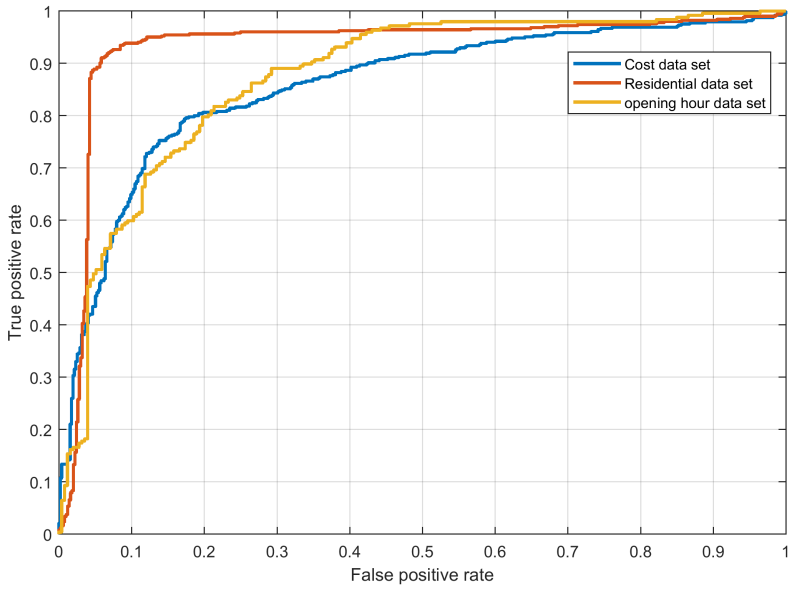


Figure 40. ROC curves of all three SVM classifiers (larger area under curve is better)

Model accuracy vs. parking event inclusion window size As explained in section 3.2.4, the idea of predicting parking area attributes is based on the assumption that parking behavior around a particular parking area is influenced by the respective attribute itself. However, parking events are spatially distributed a little inaccurately due to GPS inaccuracies or through parking event detection imprecisions. The question is now, in which window around a parking spot parking events should be considered when making this prediction. Setting the edge length of the window too high bears the risk of falsely considering parking events that are actually influenced by restrictions from other (but nearby) parking areas.

Choosing a very small parking event inclusion window limits the number of parking events, which makes it hard to learn patterns from their temporal distribution. Figure 41 illustrates this issue schematically. The blue square represents the parking event inclusion window. If the square's edge length is set too high, "green" parking events which actually belong to the free parking area in the upper left corner will be considered too, when investigating the metered parking area in the square's centroid. Doing so increases the chance of misclassifications. However, if the inclusion window edge length is set too low, parking events which actually belong the correct parking area and carry important information will eventually get missed.

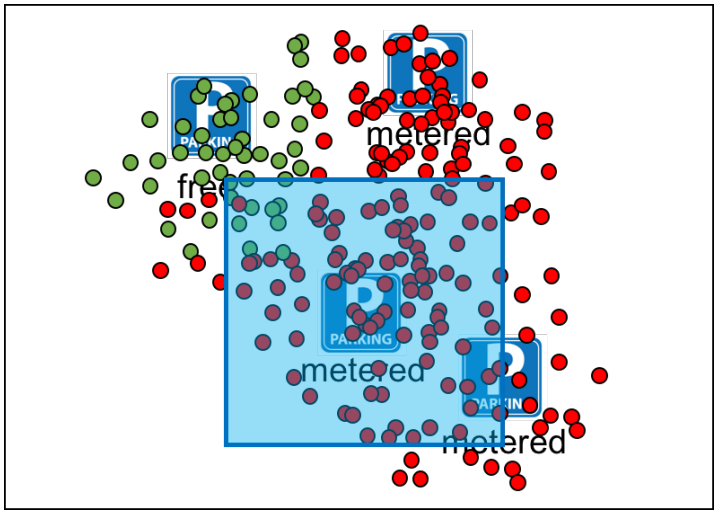


Figure 41. Schematic parking event distribution (green correspond to free, red to metered areas)

To determine a good inclusion window size, generalized classification accuracy is formulated as a function of the rectangle's edge length. The edge length is varied systematically and a reference SVM is trained with all parking events inside the current detection window. Figure 42 presents the results between 10 and 1000 meters edge length. The large gradient between 10 and 50 meters indicates that classification accuracy drops drastically if the window does not include enough

parking events to learn from. The curve reaches at maximum at 500 meters edge length. A higher edge length causes lower accuracy, due to the fact of other nearby parking areas disturbing the parking event distribution of interest. Since the processing of feature preparation, training and cross-validation, especially with increasing numbers of included parking events, is very time consuming, this test was only conducted regarding the cost data set. It is therefore assumed that the parking event-accuracy relation is similar in the residential and opening hour problem, because the effect causing this phenomenon (see figure 41) is the same.

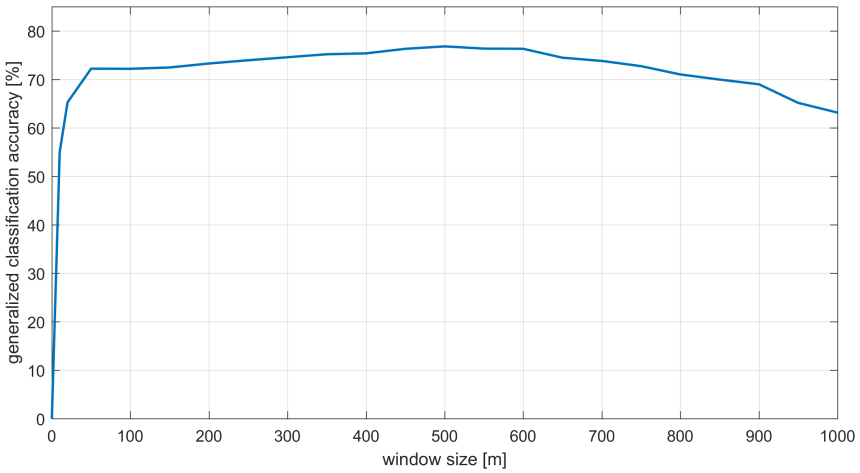


Figure 42. Generalized cost classifier accuracy vs. parking event inclusion window size

4.2.3 Sources of error

As pointed out in the previous section, classification accuracy has not yet achieved satisfactory levels, especially for the cost and opening hour problems. The causes for the remaining error are diverse:

- Inaccurate training data
- Insufficient floating car data resolution or density
- Too large radius for parking event inclusion

- Faulty class labels
- Unfit features
- Unsuitable choice of model
- Insufficient size of training data set

"Inaccurate training data" refers to data quality issues in the OSM data, which was used to create the original training data set. OSM training data may include false, outdated or misleadingly named entries. Faulty class labels are hence also caused by OSM data quality issues. Unfortunately there is no scientific way to assess the quality of the used OSM information, as the data is sparsely distributed across the entire German territory. Data quality may only be judged based on the findings in literature as pointed out in section 3.2.3. At the current point, it can only be concluded that OSM, as an user contributed source of training data, is likely to contain some error, although it can not be quantified. However, at the moment there is no alternative source of metadata publicly available to overcome this issue.

Insufficient amounts of parking events lead to another problem: If the parking event density in an area is low, the radius of parking event inclusion around a parking area needs to be set higher in order to retrieve sufficient amounts of parking events. However, doing so bears the risk of mistakenly adding parking events to one parking area data set of one class, while those parking events actually reflect parking behavior of another parking area from a different class. Setting the radius for parking event inclusion higher therefore increases the chance of duplicate parking event allocation, which leads to blurring class boundaries since class-specific data patterns may get mixed up. The effect of this error source was exploited and minimized by testing classification performance depending on

different radii of parking event inclusion as explained in section 4.2.2. As previously shown, classification accuracy decreases drastically if the radius of inclusion is set too low and the number of included parking events is therefore not enough. However, effects of this phenomenon will be diminished in the future, as the spatial parking event density inevitably increases over time, and as new data is added to the database continuously.

Other potential reasons for error could be grounded in bad choice of features or classification model. However, those factors are relatively controllable compared to all aforementioned effects. Design and choice of used features have been extensively and carefully carried out and the used methodology is supported by a broad base of literature. Since the chosen quadratic SVM classifier outperformed multiple state of the art classifiers like decision trees, KNNs, ANNs or random forests (see section 4.2.2), the existence of any significantly more accurate classification technique is unlikely.

Lastly, classifiers like support vector machines require a certain amount of training data in order to learn class-characteristic patterns. The size of a sufficient training data set depends on many factors including the number of classes, kernel type, quality of features and general problem complexity. A common way to assess training set sufficiency is to consider the classification error to be a function of training set size [111]. If the resulting learning curve flattens with increasing data set size, additional training data will not lead to significant improvements in classification accuracy. However, if the learning curve indicates a downtrend, obtaining more training data might help improve the classifier's performance. Figure 43 provides the cost classifiers' learning curve. The graph was created by using random sub samples, which represent a certain share of the entire dataset. The training error curve is based on the classification error, if the classifier is applied

to the data it was trained with. The test error curve is obtained by applying the classifier to a set of 1,000 samples, which were not included in the training data.

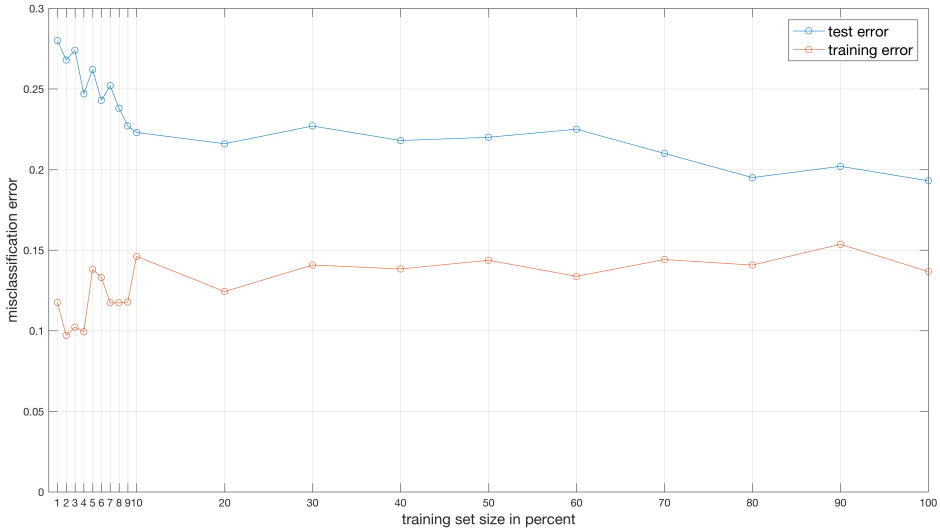


Figure 43. Learning curve of cost classifier, total size of training set: 22,172 elements

The test error curve shows a slight downtrend, which indicates that additional training data could further improve classification accuracy. However, most significant improvements are achieved within the first 10 percent of the training data, which means about 2,200 data points in the cost data set.

Since all three classification problems depend on very similar relevant features as discussed in previous chapters, required training set size is expected to behave similarly. This explains the relatively low accuracy of the opening hour classifier, as it was trained only using about 4,300 training samples. Figure 44 supports this assumption, because the opening hour classifier test error curve decreases much faster compared to the learning curve in figure 43, which was trained with five times more data.

As the list suggests, real world data mining problems, like the given one, are prone to many sources of error. As experienced by other researchers in all different

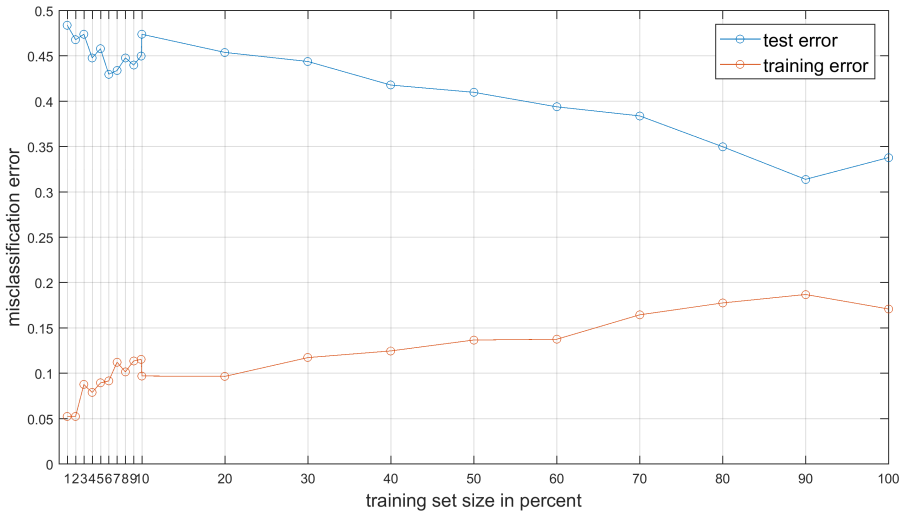


Figure 44. Learning curve of opening hour classifier, total size of training set: 4,318 elements

kinds of fields, the problem domain does not provide enough information to determine exactly what error percentage is contributed by which particular source. Based on the domain knowledge gained throughout this study, low spatial parking event density and inaccurate training data are suspected to be the main sources of error. This assumption is supported by the fact that parking area restrictions typically occur in clusters. This means that parking area restrictions are often found to be similar for adjacent parking areas [3]. But although restrictions tend to occur in clusters, there are also "outliers". An example for an "outlier" could be one single metered on-street parking area in a mostly residential neighborhood with free parking. A classifier, which uses a parking event inclusion radius of 400 meters is, of course, not accurate enough to detect outliers of this kind. Since the current classifier works at a resolution above the spatial resolution of parking area restrictions, "outliers" turn into classification error. In the future, the classifiers parking event inclusion radius may be decreased, because of the increasing spatial parking event density. This also means improving classification accuracy, as the

spatial resolution of the classifier is raised.

4.3 General discussion of results

After having evaluated and discussed each component of the developed process in depth, the overall result needs to be compared to the original objective of this thesis. The requirement of high scalability is fulfilled because the developed method only relies on remote sensing imagery, road network information and floating car data. All of this data is homogeneously available for almost any urbanized area in the world. Therefore, the proposed process could be used to map parking areas across entire cities. It must be noted however, that the individual components may still need to be optimized with regards to execution time.

In its current implementation, the process is only able to map roadside parking areas, which are aligned in parallel to the road. This is due to the fact that vehicles not within the road network are not further considered for analysis. Also the line extraction algorithm would currently not be able to deal with off-street parking areas, where vehicles park in clusters instead of lines. This problem could be solved by extending the line extraction algorithm's capabilities and including densely populated off-street vehicle clusters during the mapping process. However, since off-street parking is already covered to some degree by other sources as pointed out in section 2.1, so this limitation does not represent a severe impact. Moreover, bringing off-street vehicles into the problem would, in the current implementation, increase the risk of misclassifications of on-street parking areas and extend processing time (see 3.1.5).

Floating car data turned out to be a very informative source of parking meta information. Although the investigated binary classification problems are formulated quite simple, they impressively revealed the value of floating car data from smartphones. The accuracy levels that can be achieved using state of the art ma-

chine learning tools are remarkable, although not yet perfect. As the analysis of error sources in section 4.2.3 proves, processing more data will increase spatial resolution of the method and improve accuracy of the used classifiers. Furthermore, even more detailed questions, like for specific opening hours or different pricing models could be answered with larger amounts of parking events. The used dataset includes data from only about six months. Using more data from longer time spans is an effective measure for further improvement of the results of this thesis.

The results given by the chosen validation method of investigating a 500 x 500 m test area in Braunschweig are probably not entirely representative for all German cities or even other countries. Additionally, the metadata training set is based on user-contributed OpenStreetMap data with unknown exact data quality as already discussed in section 4.2.3. However, as in many spatial analysis problems, obtaining high quality validation data is very time-consuming. Acquiring reference data for a larger spatial area would imply quadratic increase in data collection effort. In order to obtain more representative validation results, one could try to obtain reference data, which is homogeneously distributed within the entire extent of process validity. However, doing so and collecting data in many sparsely distributed locations would lead to disproportionately high travel efforts and is therefore also considered infeasible within the scope of this study. Following the Pareto principle, investigating a small yet characteristic example, like the test setting in Braunschweig, is considered a sufficient starting point for developing and testing a new mapping process as in this case [112] Nonetheless, the validity of the parking area mapping process could be increased in the future by collecting additional metadata for training and general validation data.

CHAPTER 5

Conclusion

5.1 Summary

Throughout this thesis, a process for generating urban parking area maps has been developed, which is composed of two core components: Extracting parking area positions from remote sensing imagery and mining spatiotemporal traffic data (floating car data) to obtain corresponding parking area metainformation. The method for parking area detection is based on image processing and computer vision techniques. In order to minimize classification and processing time, a robust multistage process of image segmentation, image masking with GIS data and parking vehicle detection is developed and applied. Parking areas are determined by recursively analyzing the relative position and distance among parking vehicles and between vehicles and the road orientation. Doing so increases the method's robustness against misclassifications. The method detects on-street parking areas as lines on the road sides.

Only knowing about the geographic position of parking areas is not yet beneficial. Drivers need to have additional information, when they are deciding about a particular parking area where they will navigate to. This "parking area metadata" includes information like user group restriction (e.g. residents only), prices or opening hours. Manually collecting this data is expensive, time-consuming and not scalable and therefore not a preferable option. Floating car data in the form of "parking events" provided by smartphones are a promising alternative, potentially capable of providing all of this information. This hypothesis is based on the assumption that parking restrictions affect parking behavior, which can be captured by floating car data. This concept is proven with three exemplary binary classification models to predict: free vs. metered, residential vs. public and 24/7

opened vs. not-24/7 opened parking spots. The models are trained using the approach of supervised machine-learning on labeled datasets. The class labels are retrieved from OpenStreetMap. Screening tests with multiple models revealed that support vector machines using non-linear kernels perform best on these datasets. Residential vs. public parking areas can already be classified at an accuracy above 95 percent. The distinction between free and metered parking spots can be made at an accuracy of about 80 percent. Predicting opening hours remains difficult: The best classifier performed only at about 63 percent accuracy. An analysis of the error sources revealed that more training data (labeled examples as well as parking events) will lead to further improvements with this method.

The fusion of these floating car data analysis results with parking areas extracted from remote sensing imagery is a highly scalable source of parking information, which could potentially be applied in any urbanized area in the world.

5.2 Future research

Several parts of the proposed process still need improvements, since it was either not possible to implement a satisfying solution with the used hardware setup or these process steps are prone to technical limitations in their respective field of research.

In particular, the hardware setup with a one gigabyte GPU memory presents a strong limitation to the achievable accuracy in vehicle detection. A better hardware setup with at least four or up to six gigabytes of GPU memory would allow the design and training of a CNN architecture which is more suitable to the vehicle detection problem than standard solutions. Domain-specific adjusting of a pretrained CNN would also increase accuracy compared to the current approach of using CNN features together with a SVM classifier.

Regarding the extraction process for metadata, classification error is mainly

due to insufficient spatial density of the used dataset. The dataset only includes data starting from August 2016. Approximately 500,000 new parking events in Germany are added to the database every day, which results in even higher spatial data density. Therefore, classification accuracy will improve in the future just by this effect. Additionally, also more feature and model variations may be tested for further improvements of the internal model performance.

Lastly, in order to transfer the proposed method into a productive system, improvements regarding processing speed should be made. The current process was prototyped in Matlab, since the package is quite powerful for operating large matrices like images or feature maps. Although the use of a high level script language like Matlab allows adjustments to be made very easily, iterative tasks that can not be vectorized are executed slowly, compared to languages like C or C++. For future work, it is recommended to implement the computationally expensive tasks like feature extraction and iterative operations on large datasets using frameworks like OpenCV for computer vision and TensorFlow for general machine learning tasks [113, 114]. Both frameworks provide Python APIs (Application Programming Interfaces), which could be used for controlling the process and handling data on a higher level. Implemented this way, parking area mapping can be executed on a remote server and even be triggered on a regular basis to improve results and to account for infrastructure changes over time.

LIST OF REFERENCES

[1] N. Moini, D. Hill, and M. Gruteser, "Impact analyses of curb-street parking guidance system on mobility and environment," Center for advanced Infrastructure & Transportation Rutgers University," resreport, 02 2012. [Online]. Available: https://cait.rutgers.edu/files/ATT-RU3528_0.pdf

[2] J. Pucher, "Urban travel behavior as the outcome of public policy: The example of modal-split in western europe and north america," *Journal of the American Planning Association*, vol. 54, no. 4, pp. 509–520, dec 1988. [Online]. Available: https://doi.org/10.1080%2F01944368808976677

[3] G. Pasaoglu, D. Fiorello, A. Martino, G. Scarcella, A. Alemanno, A. Zubaryeva, and C. Thiel, "Driving and parking patterns of european car drivers-a mobility survey," *Luxembourg: European Commission Joint Research Centre*, 2012.

[4] R. Salpietro, L. Bedogni, M. D. Felice, and L. Bononi, "Park here! a smart parking system based on smartphones' embedded sensors and short range communication technologies," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, dec 2015. [Online]. Available: https://doi.org/10.1109%2Fwf-iot.2015.7389020

[5] G. Ma, M. Dwivedi, R. Li, and C. Sun, "Parking guidance system," in *SAE Technical Paper Series*. SAE International, jan 2011. [Online]. Available: https://doi.org/10.4271%2F2011-26-0095

[6] S. Nawaz, C. Efstratiou, and C. Mascolo, "ParkSense," in *Proceedings of the 19th annual international conference on Mobile computing & networking - MobiCom '13*. ACM Press, 2013. [Online]. Available: https://doi.org/10.1145%2F2500423.2500438

[7] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "ParkNet," in *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*. ACM Press, 2010. [Online]. Available: https://doi.org/10.1145%2F1814433.1814448

[8] M. Chen and S. Chien, "Dynamic freeway travel-time prediction with probe vehicle data: Link based versus path based," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1768, pp. 157–161, jan 2001. [Online]. Available: https://doi.org/10.3141%2F1768-19

[9] M. Rahmani, H. N. Koutsopoulos, and A. Ranganathan, "Requirements and potential of GPS-based floating car data for traffic management: Stockholm case study," in *13th International IEEE Conference on Intelligent Transportation Systems*. Institute of Electrical and Electronics Engineers (IEEE), sep 2010. [Online]. Available: https://doi.org/10.1109%2Fitsc.2010.5625177

[10] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, feb 2010. [Online]. Available: https://doi.org/10.1126%2Fscience.1177170

[11] M. C. González, C. A. Hidalgo, and A.-L. Barabási, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, jun 2008. [Online]. Available: https://doi.org/10.1038%2Fnature06958

[12] DIN - German Standardization Institute, "Intelligent transport systems - traffic and travel information via transport protocol experts group, generation 1 (tpeg1), binary data format - part 7: Parking inforamtion," DIN Deutsches Institut für Normung e.V. (German Standardization Institute), Standard DIN CEN ISO/TS 18234-7, 2014.

[13] J. Ansorge, H. Kirschfink, and S. von der Ruhren, "Einbindung städtischer verkehrsinformationen in ein regionales verkehrsmanagement," 2010.

[14] M. Haklay and P. Weber, "OpenStreetMap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, oct 2008. [Online]. Available: https://doi.org/10.1109%2Fmprv.2008.80

[15] S. Rikus, D. S. Hoffmann, T. Ungureanu, D. S. Rommerskirchen, and M. Plesker, "Auskunft über verfügbare parkplätze in städten," 2015.

[16] P. Neis and A. Zipf, "Analyzing the contributor activity of a volunteered geographic information project — the case of OpenStreetMap," *ISPRS International Journal of Geo-Information*, vol. 1, no. 3, pp. 146–165, jul 2012. [Online]. Available: https://doi.org/10.3390%2Fijgi1020146

[17] OpenStreetMap contributors, "OSM Tag distribution retrieved from https://taginfo.openstreetmap.org/keys/parkingoverview ," https://www.openstreetmap.org, 2017.

[18] Inrix Inc. . "Inrix parking & traffic." 2017. [Online]. Available: http://inrix.com/products/

[19] ilogs Gmbh. "Parkinghq suite." [Online]. Available: http://www.ilogs.com/en/parkinghq-suite-en

[20] Streetline Inc. "The streetline solution." 2017. [Online]. Available: https://www.streetline.com/how-it-works/

[21] Parkopedia Ltd. "about parkopedia." [Online]. Available: https://secure.parkopedia.com/about-us/

[22] D. S. Evans and R. Schmalensee, "Failure to launch: Critical mass in platform businesses," *Review of Network Economics*, vol. 9, no. 4, 2010.

[23] S. Lorkowski, P. Mieth, and R.-P. Schaefer, "New its applications for metropolitan areas based on floating car data," in *ECTRI Young Researcher Conference*, DLR. DLR, 2005.

[24] E. Brockfeld and S. Lorkowski, "Benefits and limits of recent floating car data technology - an evaluation study," in *WCTR Conference*, ser. WCTR Conference, DLR (German Aerospace Center). DLR, Dec. 2007. [Online]. Available: http://elib.dlr.de/49618/1/Brockfeld_WCTR2007.pdf

[25] R.-P. Schäfer and K.-U. Thiessenhusen, "About traffic information system by means of real-time floating-car data," in *Institute of Transport Research*. DLR (German Aerospace Center), Institute of Transport Research, 2002.

[26] S. Messelodi, C. M. Modena, M. Zanin, F. G. D. Natale, F. Granelli, E. Betterle, and A. Guarise, "Intelligent extended floating car data collection," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4213–4227, apr 2009. [Online]. Available: https://doi.org/10.1016%2Fj.eswa.2008.04.008

[27] R. Bishop. "Floating car data projects worldwide." 2004. [Online]. Available: http://www.authorstream.com/Presentation/Gourangi-56968-040426-Bishop-FloatingCarProjects-Floating-Car-Data/Projects-Worldwide-Selective-Review-Outline-f-Education-ppt-powerpoint/

[28] J.-G. Krieg, G. Jakllari, H. Toma, and A.-L. Beylot, "Unlocking the smartphone's senses for smart city parking," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–7.

[29] T. Jeske, "Floating car data from smartphones: What google and waze about you and how hacker can control traffic," 2013. [Online]. Available: https://media.blackhat.com/eu-13/briefings/Jeske/bh-eu-13-floating-car-data-jeske-wp.pdf

[30] J. Li, Q. Qin, L. You, and C. Xie, "Parking lot extraction method based on floating car data," *Geomatics and Information Science of Wuhan University*, vol. 38, no. 5, pp. 599–603, 2013.

[31] G. C. S. Linda G. Shapiro, *Computer Vision*. ADDISON WESLEY PUB CO INC, 2001.

[32] J. Serra, *Image analysis and mathematical morphology, v. 1*. Academic press, 1982.

[33] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 4, pp. 532–550, jul 1987. [Online]. Available: https://doi.org/10.1109%2Ftpami.1987.4767941

[34] O. Trier and T. Taxt, "Evaluation of binarization methods for document images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 312–315, mar 1995. [Online]. Available: https://doi.org/10.1109%2F34.368197

[35] B. Manjunath, J.-R. Ohm, V. Vasudevan, and A. Yamada, "Color and texture descriptors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 703–715, jun 2001. [Online]. Available: https://doi.org/10.1109%2F76.927424

[36] J. Serra, "Introduction to mathematical morphology," *Computer Vision, Graphics, and Image Processing*, vol. 35, no. 3, pp. 283–305, sep 1986. [Online]. Available: https://doi.org/10.1016%2F0734-189x%2886%2990002-2

[37] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery : A small target detection benchmark," *Journal of Visual Communication and Image Representation*, vol. 34, pp. 187–203, jan 2016. [Online]. Available: https://doi.org/10.1016%2Fj.jvcir.2015.11.002

[38] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[39] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05*. IEEE, 2005. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2005.177

[40] V. Angelis, G. Felici, and G. Mancinelli, "Feature selection for data mining," in *Massive Computing*. Springer US, 2006, pp. 227–252. [Online]. Available: https://doi.org/10.1007%2F0-387-34296-6_6

[41] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, aug 2014. [Online]. Available: https://doi.org/10.1109%2Ftpami.2014.2300479

[42] M. Pietikäinen, T. Mäenpää, and J. Viertola, "Color texture classification with color histograms and local binary patterns," in *Workshop on Texture Analysis in Machine Vision*. Citeseer, 2002, pp. 109–112.

[43] L. Juan and O. Gwun, "A comparison of sift, pca-sift and surf," *International Journal of Image Processing (IJIP)*, vol. 3, no. 4, pp. 143–152, 2009.

[44] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[45] D. Nistér and H. Stewénius, "Linear time maximally stable extremal regions," *Computer Vision–ECCV 2008*, pp. 183–196, 2008.

[46] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels," ., Tech. Rep., 2010.

[47] C.-h. Chen, *Handbook of pattern recognition and computer vision*. World Scientific, 2015.

[48] W. Commons. "Two classes given as vectors with two possible separation lines and their corresponding margin areas between the class areas. line a has a larger empty margin area than line b." 2010. [Online]. Available: https://de.wikipedia.org/wiki/Datei:Svm_intro.svg

[49] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[50] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, sep 1995. [Online]. Available: https://doi.org/10.1007%2Fbf00994018

[51] C.-C. Chang and C.-J. Lin, "LIBSVM," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, apr 2011. [Online]. Available: https://doi.org/10.1145%2F1961189.1961199

[52] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[54] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–104.

[55] Theano Development Team. "Convolutional neural networks (lenet)." 2010. [Online]. Available: http://deeplearning.net/tutorial/lenet.html

[56] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, jan 2016. [Online]. Available: https://doi.org/10.1109%2Ftpami.2015.2437384

[57] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2. IEEE, 2006, pp. 1735–1742.

[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 248–255.

[59] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region/proposal-networks.pdf

[60] R. Girshick, "Fast r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[61] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[62] A. Gruen and H. Li, "Road extraction from aerial and satellite images by dynamic programming," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 50, no. 4, pp. 11–20, aug 1995. [Online]. Available: https://doi.org/10.1016%2F0924-2716%2895%2998233-p

[63] J. Chanussot and P. Lambert, "An application of mathematical morphology to road network extraction on sar images," *COMPUTATIONAL IMAGING AND VISION*, vol. 12, pp. 399–406, 1998.

[64] F. Tupin, H. Maitre, J.-F. Mangin, J.-M. Nicolas, and E. Pechersky, "Detection of linear features in sar images: Application to road network extraction," *IEEE transactions on geoscience and remote sensing*, vol. 36, no. 2, pp. 434–453, 1998.

[65] S. Hinz and A. Baumgartner, "Automatic extraction of urban road networks from multi-view aerial imagery," *ISPRS Journal of Photogrammetry and*

Remote Sensing, vol. 58, no. 1-2, pp. 83–98, jun 2003. [Online]. Available: https://doi.org/10.1016%2Fs0924-2716%2803%2900019-4

[66] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, dec 2012. [Online]. Available: https://doi.org/10.1109%2Ftits.2012.2209421

[67] N. Buch, S. A. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920–939, sep 2011. [Online]. Available: https://doi.org/10.1109%2Ftits.2011.2119372

[68] N. True, "Vacant parking space detection in static images," *University of California, San Diego*, vol. 17, 2007.

[69] W. Lixia and J. Dalin, "A method of parking space detection based on image segmentation and LBP," in *2012 Fourth International Conference on Multimedia Information Networking and Security*. IEEE, nov 2012. [Online]. Available: https://doi.org/10.1109%2Fmines.2012.27

[70] C.-C. Huang and H. T. Vu, "A multi-layer discriminative framework for parking space detection," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[71] X. Wang and A. R. Hanson, "Parking lot analysis and visualization from aerial images," in *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*. IEEE, 1998, pp. 36–41.

[72] Y.-W. Seo, N. D. Ratliff, and C. Urmson, "Self-supervised aerial image analysis for extracting parking lot structure." in *IJCAI*, 2009, pp. 1837–1842.

[73] A. H. Mexas and M. Marengoni, "Unsupervised recognition of parking lot areas," in *Proceedings of the International Conference on Machine Vision and Machine Learning*, 2014.

[74] T. Blaschke, G. J. Hay, M. Kelly, S. Lang, P. Hofmann, E. Addink, R. Q. Feitosa, F. van der Meer, H. van der Werff, F. van Coillie, and D. Tiede, "Geographic object-based image analysis – towards a new paradigm," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 180–191, jan 2014. [Online]. Available: https://doi.org/10.1016%2Fj.isprsjprs.2013.09.014

[75] T. Blaschke, "Object based image analysis for remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 1, pp. 2–16, jan 2010. [Online]. Available: https://doi.org/10.1016%2Fj.isprsjprs.2009.06.004

[76] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 92, pp. 79–97, jun 2014. [Online]. Available: https://doi.org/10.1016%2Fj.isprsjprs.2014.02.013

[77] Satellite Imaging Corporation. "Satellite sensors." 2017. [Online]. Available: http://www.satimagingcorp.com/satellite-sensors/

[78] Landesamt für Geoinformation und Landesvermessung Niedersachsen. "Digitale luftbilder niedersachsen." 2017. [Online]. Available: luftbildniedersachsen

[79] LAND INFO Worldwide Mapping, LLC. "Buying satellite imagery: Pricing information for high resolution satellite imagery." 2017. [Online]. Available: http://www.landinfo.com/satellite-imagery-pricing.html

[80] A. Matese, P. Toscano, S. D. Gennaro, L. Genesio, F. Vaccari, J. Primicerio, C. Belli, A. Zaldei, R. Bianconi, and B. Gioli, "Intercomparison of UAV, aircraft and satellite remote sensing platforms for precision viticulture," *Remote Sensing*, vol. 7, no. 3, pp. 2971–2990, mar 2015. [Online]. Available: https://doi.org/10.3390%2Frs70302971

[81] A. Gerhardinger, D. Ehrlich, and M. Pesaresi, "Vehicles detection from very high resolution satellite imagery," *International Archives of Photogrammetry and Remote Sensing*, vol. 36, no. Part 3, p. W24, 2005.

[82] S. Hinz, D. Lenhart, and J. Leitloff, "Traffic extraction and characterisation from optical remote sensing data," *The Photogrammetric Record*, vol. 23, no. 124, pp. 424–440, dec 2008. [Online]. Available: https://doi.org/10.1111%2Fj.1477-9730.2008.00497.x

[83] X. Teng, L. Cao, C. Wang, and Z. Chen, "Vehicle detection from remote sensing image based on superpixel segmentation and image enhancement," in *Proceedings of International Conference on Internet Multimedia Computing and Service*. ACM, 2014, p. 140.

[84] J. Leitloff, S. Hinz, and U. Stilla, "Vehicle queue detection in satellite images of urban areas," in *Proceedings of the ISPRS joint conference 3rd International Symposium Remote Sensing and Data Fusion Over Urban Areas (URBAN 2005) and 5th International Symposium Remote Sensing of Urban Areas (URS 2005). Tempe, AZ, USA*, 2005.

[85] G. Palubinskas, F. Kurz, and P. Reinartz, "Detection of traffic congestion in optical remote sensing imagery," in *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, vol. 2. IEEE, 2008, pp. II–426.

[86] J.-Y. Choi and Y.-K. Yang, "Vehicle detection from aerial images using local shape information," in *Advances in Image and Video Technology*. Springer Berlin Heidelberg, 2009, pp. 227–236. [Online]. Available: https://doi.org/10.1007%2F978-3-540-92957-4_20

[87] J. Leitloff, S. Hinz, and U. Stilla, "Vehicle detection in very high resolution satellite images of city areas," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 7, pp. 2795–2806, jul 2010. [Online]. Available: https://doi.org/10.1109%2Ftgrs.2010.2043109

[88] J. Gleason, A. V. Nefian, X. Bouyssounousse, T. Fong, and G. Bebis, "Vehicle detection from aerial imagery," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, may 2011. [Online]. Available: https://doi.org/10.1109%2Ficra.2011.5979853

[89] K. Liu and G. Mattyus, "Fast multiclass vehicle detection on aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1938–1942, sep 2015. [Online]. Available: https://doi.org/10.1109%2Flgrs.2015.2439517

[90] M. Shu and S. Du, "Geoscene-based vehicle detection from very-high-resolution images," in *2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*. IEEE, jul 2016. [Online]. Available: https://doi.org/10.1109%2Feorsa.2016.7552816

[91] L. Cao, C. Wang, and J. Li, "Vehicle detection from highway satellite images via transfer learning," *Information Sciences*, vol. 366, pp. 177–187, oct 2016. [Online]. Available: https://doi.org/10.1016%2Fj.ins.2016.01.004

[92] H. Li, K. Fu, M. Yan, X. Sun, H. Sun, and W. Diao, "Vehicle detection in remote sensing images using denoizing-based convolutional neural networks," *Remote Sensing Letters*, vol. 8, no. 3, pp. 262–270, dec 2016. [Online]. Available: https://doi.org/10.1080%2F2150704x.2016.1258127

[93] C. Chen, M.-Y. Liu, O. Tuzel, and J. Xiao, "R-cnn for small object detection," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 214–230.

[94] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3. IEEE, 2006, pp. 850–855.

[95] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "Crisp-dm 1.0 step-by-step data mining guide," 2000.

[96] M. Ligas and P. Banasik, "Conversion between cartesian and geodetic coordinates on a rotational ellipsoid by solving a system of nonlinear equations," *Geodesy and Cartography*, vol. 60, no. 2, pp. 145–159, 2011.

[97] A. M. Andrew, "An introduction to support vector machines and other kernel-based learning methods by nello christianini and john shawe-taylor, cambridge university press, cambridge, 2000, xiii+ 189 pp., isbn 0-521-78019-5 (hbk,£ 27.50)." 2000.

[98] O. Kounadi, "Assessing the quality of openstreetmap data," *Msc geographical information science, University College of London Department of Civil, Environmental And Geomatic Engineering*, 2009.

[99] P. Mooney, P. Corcoran, and A. C. Winstanley, "Towards quality metrics for openstreetmap," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2010, pp. 514–517.

[100] Census database of the census 2011 from Federal Statistical Offices. "Zensus2011." 2011. [Online]. Available: https://ergebnisse.zensus2011.de/?locale=en#dynTable:

[101] Facebook Inc. "Event graph api v2.9." 2017. [Online]. Available: https://developers.facebook.com/docs/graph-api/reference/event

[102] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[103] Google Inc. "Google maps apis." 2017. [Online]. Available: https://developers.google.com/maps/?hl=en

[104] HERE Global B.V. "Here maps for developers." 2017. [Online]. Available: https://developer.here.com

[105] ESRI Inc. "Arcgis for developers." 2017. [Online]. Available: https://developers.arcgis.com

[106] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2015.

[107] W. Commons. "Precision and recall." 2014. [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg

[108] S. Tuermer, F. Kurz, P. Reinartz, and U. Stilla, "Airborne vehicle detection in dense urban areas using hog features and disparity maps," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 6, pp. 2327–2337, 2013.

[109] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[110] G. E. Box, J. S. Hunter, and W. G. Hunter, *Statistics for experimenters: design, innovation, and discovery.* Wiley-Interscience New York, 2005, vol. 2.

[111] M. Opper and R. Urbanczik, "Universal learning curves of support vector machines," *Physical Review Letters*, vol. 86, no. 19, p. 4410, 2001.

[112] V. Pareto, *The mind and society.* Ripol Klassik, 1935, vol. 1.

[113] G. Bradski, *Dr. Dobb's Journal of Software Tools*, 2000.

[114] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S., "Slic superpixels," ., Tech. Rep., 2010.

Andrew, A. M., "An introduction to support vector machines and other kernel-based learning methods by nello christianini and john shawe-taylor, cambridge university press, cambridge, 2000, xiii+ 189 pp., isbn 0-521-78019-5 (hbk,£ 27.50)." 2000.

Angelis, V., Felici, G., and Mancinelli, G., "Feature selection for data mining," in *Massive Computing*. Springer US, 2006, pp. 227–252. [Online]. Available: https://doi.org/10.1007%2F0-387-34296-6_6

Ansorge, J., Kirschfink, H., and von der Ruhren, S., "Einbindung städtischer verkehrsinformationen in ein regionales verkehrsmanagement," 2010.

Bishop, R. "Floating car data projects worldwide." 2004. [Online]. Available: http://www.authorstream.com/Presentation/Gourangi-56968-040426-Bishop-FloatingCarProjects-Floating-Car-Data/Projects-Worldwide-Selective-Review-Outline-f-Education-ppt-powerpoint/

Blaschke, T., "Object based image analysis for remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 1, pp. 2–16, jan 2010. [Online]. Available: https://doi.org/10.1016%2Fj.isprsjprs.2009.06.004

Blaschke, T., Hay, G. J., Kelly, M., Lang, S., Hofmann, P., Addink, E., Feitosa, R. Q., van der Meer, F., van der Werff, H., van Coillie, F., and Tiede, D., "Geographic object-based image analysis – towards a new paradigm," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 180–191, jan 2014. [Online]. Available: https://doi.org/10.1016%2Fj.isprsjprs.2013.09.014

Box, G. E., Hunter, J. S., and Hunter, W. G., *Statistics for experimenters: design, innovation, and discovery*. Wiley-Interscience New York, 2005, vol. 2.

Bradski, G., *Dr. Dobb's Journal of Software Tools*, 2000.

Brockfeld, E. and Lorkowski, S., "Benefits and limits of recent floating car data technology - an evaluation study," in *WCTR Conference*, ser. WCTR Conference, DLR (German Aerospace Center). DLR, Dec. 2007. [Online]. Available: http://elib.dlr.de/49618/1/Brockfeld_WCTR2007.pdf

Buch, N., Velastin, S. A., and Orwell, J., "A review of computer vision techniques for the analysis of urban traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920–939, sep 2011. [Online]. Available: https://doi.org/10.1109%2Ftits.2011.2119372

Cao, L., Wang, C., and Li, J., "Vehicle detection from highway satellite images via transfer learning," *Information Sciences*, vol. 366, pp. 177–187, oct 2016. [Online]. Available: https://doi.org/10.1016%2Fj.ins.2016.01.004

Census database of the census 2011 from Federal Statistical Offices. "Zensus2011." 2011. [Online]. Available: https://ergebnisse.zensus2011.de/?locale=en#dynTable:

Chang, C.-C. and Lin, C.-J., "LIBSVM," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, apr 2011. [Online]. Available: https://doi.org/10.1145%2F1961189.1961199

Chanussot, J. and Lambert, P., "An application of mathematical morphology to road network extraction on sar images," *COMPUTATIONAL IMAGING AND VISION*, vol. 12, pp. 399–406, 1998.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R., "Crisp-dm 1.0 step-by-step data mining guide," 2000.

Chen, C., Liu, M.-Y., Tuzel, O., and Xiao, J., "R-cnn for small object detection," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 214–230.

Chen, C.-h., *Handbook of pattern recognition and computer vision*. World Scientific, 2015.

Chen, M. and Chien, S., "Dynamic freeway travel-time prediction with probe vehicle data: Link based versus path based," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1768, pp. 157–161, jan 2001. [Online]. Available: https://doi.org/10.3141%2F1768-19

Choi, J.-Y. and Yang, Y.-K., "Vehicle detection from aerial images using local shape information," in *Advances in Image and Video Technology*. Springer Berlin Heidelberg, 2009, pp. 227–236. [Online]. Available: https://doi.org/10.1007%2F978-3-540-92957-4_20

Colomina, I. and Molina, P., "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 92, pp. 79–97, jun 2014. [Online]. Available: https://doi.org/10.1016%2Fj.isprsjprs.2014.02.013

Commons, W. "Two classes given as vectors with two possible separation lines and their corresponding margin areas between the class areas. line a has a larger empty margin area than line b." 2010. [Online]. Available: https://de.wikipedia.org/wiki/Datei:Svm_intro.svg

Commons, W. "Precision and recall." 2014. [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg

Cortes, C. and Vapnik, V., "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, sep 1995. [Online]. Available: https://doi.org/10.1007%2Fbf00994018

Dalal, N. and Triggs, B., "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05*. IEEE, 2005. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2005.177

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

DIN - German Standardization Institute, "Intelligent transport systems - traffic and travel information via transport protocol experts group, generation 1 (tpeg1), binary data format - part 7: Parking inforamtion," DIN Deutsches Institut für Normung e.V. (German Standardization Institute), Standard DIN CEN ISO/TS 18234-7, 2014.

Dollar, P., Appel, R., Belongie, S., and Perona, P., "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, aug 2014. [Online]. Available: https://doi.org/10.1109%2Ftpami.2014.2300479

ESRI Inc. "Arcgis for developers." 2017. [Online]. Available: https://developers.arcgis.com

Evans, D. S. and Schmalensee, R., "Failure to launch: Critical mass in platform businesses," *Review of Network Economics*, vol. 9, no. 4, 2010.

Facebook Inc. "Event graph api v2.9." 2017. [Online]. Available: https://developers.facebook.com/docs/graph-api/reference/event

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D., "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

Gerhardinger, A., Ehrlich, D., and Pesaresi, M., "Vehicles detection from very high resolution satellite imagery," *International Archives of Photogrammetry and Remote Sensing*, vol. 36, no. Part 3, p. W24, 2005.

Girshick, R., "Fast r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, jan 2016. [Online]. Available: https://doi.org/10.1109%2Ftpami.2015.2437384

Gleason, J., Nefian, A. V., Bouyssounousse, X., Fong, T., and Bebis, G., "Vehicle detection from aerial imagery," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, may 2011. [Online]. Available: https://doi.org/10.1109%2Ficra.2011.5979853

González, M. C., Hidalgo, C. A., and Barabási, A.-L., "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, jun 2008. [Online]. Available: https://doi.org/10.1038%2Fnature06958

Google Inc. "Google maps apis." 2017. [Online]. Available: https://developers.google.com/maps/?hl=en

Gruen, A. and Li, H., "Road extraction from aerial and satellite images by dynamic programming," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 50, no. 4, pp. 11–20, aug 1995. [Online]. Available: https://doi.org/10.1016%2F0924-2716%2895%2998233-p

Guyon, I. and Elisseeff, A., "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

Hadsell, R., Chopra, S., and LeCun, Y., "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2. IEEE, 2006, pp. 1735–1742.

Haklay, M. and Weber, P., "OpenStreetMap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, oct 2008. [Online]. Available: https://doi.org/10.1109%2Fmprv.2008.80

Haralick, R. M., Sternberg, S. R., and Zhuang, X., "Image analysis using mathematical morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 4, pp. 532–550, jul 1987. [Online]. Available: https://doi.org/10.1109%2Ftpami.1987.4767941

HERE Global B.V. "Here maps for developers." 2017. [Online]. Available: https://developer.here.com

Hinz, S. and Baumgartner, A., "Automatic extraction of urban road networks from multi-view aerial imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 58, no. 1-2, pp. 83–98, jun 2003. [Online]. Available: https://doi.org/10.1016%2Fs0924-2716%2803%2900019-4

Hinz, S., Lenhart, D., and Leitloff, J., "Traffic extraction and characterisation from optical remote sensing data," *The Photogrammetric Record*, vol. 23, no. 124, pp. 424–440, dec 2008. [Online]. Available: https://doi.org/10.1111%2Fj.1477-9730.2008.00497.x

Huang, C.-C. and Vu, H. T., "A multi-layer discriminative framework for parking space detection," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

ilogs Gmbh. "Parkinghq suite." [Online]. Available: http://www.ilogs.com/en/parkinghq-suite-en

Inrix Inc. . "Inrix parking & traffic." 2017. [Online]. Available: http://inrix.com/products/

Jeske, T., "Floating car data from smartphones: What google and waze about you and how hacker can control traffic," 2013. [Online]. Available: https://media.blackhat.com/eu-13/briefings/Jeske/bh-eu-13-floating-car-data-jeske-wp.pdf

Juan, L. and Gwun, O., "A comparison of sift, pca-sift and surf," *International Journal of Image Processing (IJIP)*, vol. 3, no. 4, pp. 143–152, 2009.

Kounadi, O., "Assessing the quality of openstreetmap data," *Msc geographical information science, University College of London Department of Civil, Environmental And Geomatic Engineering*, 2009.

Krieg, J.-G., Jakllari, G., Toma, H., and Beylot, A.-L., "Unlocking the smartphone's senses for smart city parking," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–7.

Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

LAND INFO Worldwide Mapping, LLC. "Buying satellite imagery: Pricing information for high resolution satellite imagery." 2017. [Online]. Available: http://www.landinfo.com/satellite-imagery-pricing.html

Landesamt für Geoinformation und Landesvermessung Niedersachsen. "Digitale luftbilder niedersachsen." 2017. [Online]. Available: luftbildniedersachsen

LeCun, Y., Bengio, Y., *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

LeCun, Y., Huang, F. J., and Bottou, L., "Learning methods for generic object recognition with invariance to pose and lighting," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–104.

Leitloff, J., Hinz, S., and Stilla, U., "Vehicle queue detection in satellite images of urban areas," in *Proceedings of the ISPRS joint conference 3rd International Symposium Remote Sensing and Data Fusion Over Urban Areas (URBAN 2005) and 5th International Symposium Remote Sensing of Urban Areas (URS 2005). Tempe, AZ, USA*, 2005.

Leitloff, J., Hinz, S., and Stilla, U., "Vehicle detection in very high resolution satellite images of city areas," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 7, pp. 2795–2806, jul 2010. [Online]. Available: https://doi.org/10.1109%2Ftgrs.2010.2043109

Li, H., Fu, K., Yan, M., Sun, X., Sun, H., and Diao, W., "Vehicle detection in remote sensing images using denoizing-based convolutional neural networks," *Remote Sensing Letters*, vol. 8, no. 3, pp. 262–270, dec 2016. [Online]. Available: https://doi.org/10.1080%2F2150704x.2016.1258127

Li, J., Qin, Q., You, L., and Xie, C., "Parking lot extraction method based on floating car data," *Geomatics and Information Science of Wuhan University*, vol. 38, no. 5, pp. 599–603, 2013.

Ligas, M. and Banasik, P., "Conversion between cartesian and geodetic coordinates on a rotational ellipsoid by solving a system of nonlinear equations," *Geodesy and Cartography*, vol. 60, no. 2, pp. 145–159, 2011.

Linda G. Shapiro, G. C. S., *Computer Vision*. ADDISON WESLEY PUB CO INC, 2001.

Liu, K. and Mattyus, G., "Fast multiclass vehicle detection on aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 9, pp. 1938–1942, sep 2015. [Online]. Available: https://doi.org/10.1109%2Flgrs.2015.2439517

Lixia, W. and Dalin, J., "A method of parking space detection based on image segmentation and LBP," in *2012 Fourth International Conference on Multimedia Information Networking and Security*. IEEE, nov 2012. [Online]. Available: https://doi.org/10.1109%2Fmines.2012.27

Lorkowski, S., Mieth, P., and Schaefer, R.-P., "New its applications for metropolitan areas based on floating car data," in *ECTRI Young Researcher Conference*, DLR. DLR, 2005.

Ma, G., Dwivedi, M., Li, R., and Sun, C., "Parking guidance system," in *SAE Technical Paper Series*. SAE International, jan 2011. [Online]. Available: https://doi.org/10.4271%2F2011-26-0095

Manjunath, B., Ohm, J.-R., Vasudevan, V., and Yamada, A., "Color and texture descriptors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 703–715, jun 2001. [Online]. Available: https://doi.org/10.1109%2F76.927424

Matese, A., Toscano, P., Gennaro, S. D., Genesio, L., Vaccari, F., Primicerio, J., Belli, C., Zaldei, A., Bianconi, R., and Gioli, B., "Intercomparison of UAV, aircraft and satellite remote sensing platforms for precision viticulture," *Remote Sensing*, vol. 7, no. 3, pp. 2971–2990, mar 2015. [Online]. Available: https://doi.org/10.3390%2Frs70302971

Mathur, S., Jin, T., Kasturirangan, N., Chandrasekaran, J., Xue, W., Gruteser, M., and Trappe, W., "ParkNet," in *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*. ACM Press, 2010. [Online]. Available: https://doi.org/10.1145%2F1814433.1814448

Messelodi, S., Modena, C. M., Zanin, M., Natale, F. G. D., Granelli, F., Betterle, E., and Guarise, A., "Intelligent extended floating car data collection," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4213–4227, apr 2009. [Online]. Available: https://doi.org/10.1016%2Fj.eswa.2008.04.008

Mexas, A. H. and Marengoni, M., "Unsupervised recognition of parking lot areas," in *Proceedings of the International Conference on Machine Vision and Machine Learning*, 2014.

Mogelmose, A., Trivedi, M. M., and Moeslund, T. B., "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, dec 2012. [Online]. Available: https://doi.org/10.1109%2Ftits.2012.2209421

Moini, N., Hill, D., and Gruteser, M., "Impact analyses of curb-street parking guidance system on mobility and environment," Center for advanced Infrastructure & Transportation Rutgers University," resreport, 02 2012. [Online]. Available: https://cait.rutgers.edu/files/ATT-RU3528_0.pdf

Mooney, P., Corcoran, P., and Winstanley, A. C., "Towards quality metrics for openstreetmap," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2010, pp. 514–517.

Nawaz, S., Efstratiou, C., and Mascolo, C., "ParkSense," in *Proceedings of the 19th annual international conference on Mobile computing & networking - MobiCom '13*. ACM Press, 2013. [Online]. Available: https://doi.org/10.1145%2F2500423.2500438

Neis, P. and Zipf, A., "Analyzing the contributor activity of a volunteered geographic information project — the case of OpenStreetMap," *ISPRS International Journal of Geo-Information*, vol. 1, no. 3, pp. 146–165, jul 2012. [Online]. Available: https://doi.org/10.3390%2Fijgi1020146

Neubeck, A. and Van Gool, L., "Efficient non-maximum suppression," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3. IEEE, 2006, pp. 850–855.

Nistér, D. and Stewénius, H., "Linear time maximally stable extremal regions," *Computer Vision–ECCV 2008*, pp. 183–196, 2008.

OpenStreetMap contributors, "OSM Tag distribution retrieved from https://taginfo.openstreetmap.org/keys/parkingoverview ," https://www.openstreetmap.org, 2017.

Opper, M. and Urbanczik, R., "Universal learning curves of support vector machines," *Physical Review Letters*, vol. 86, no. 19, p. 4410, 2001.

Palubinskas, G., Kurz, F., and Reinartz, P., "Detection of traffic congestion in optical remote sensing imagery," in *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, vol. 2. IEEE, 2008, pp. II–426.

Pareto, V., *The mind and society*. Ripol Klassik, 1935, vol. 1.

Parkopedia Ltd. "about parkopedia." [Online]. Available: https://secure.parkopedia.com/about-us/

Pasaoglu, G., Fiorello, D., Martino, A., Scarcella, G., Alemanno, A., Zubaryeva, A., and Thiel, C., "Driving and parking patterns of european car drivers-a mobility survey," *Luxembourg: European Commission Joint Research Centre*, 2012.

Pietikäinen, M., Mäenpää, T., and Viertola, J., "Color texture classification with color histograms and local binary patterns," in *Workshop on Texture Analysis in Machine Vision*. Citeseer, 2002, pp. 109–112.

Pucher, J., "Urban travel behavior as the outcome of public policy: The example of modal-split in western europe and north america," *Journal of the American Planning Association*, vol. 54, no. 4, pp. 509–520, dec 1988. [Online]. Available: https://doi.org/10.1080%2F01944368808976677

Rahmani, M., Koutsopoulos, H. N., and Ranganathan, A., "Requirements and potential of GPS-based floating car data for traffic management: Stockholm case study," in *13th International IEEE Conference on Intelligent Transportation Systems*. Institute of Electrical and Electronics Engineers (IEEE), sep 2010. [Online]. Available: https://doi.org/10.1109%2Fitsc.2010.5625177

Razakarivony, S. and Jurie, F., "Vehicle detection in aerial imagery : A small target detection benchmark," *Journal of Visual Communication and Image Representation*, vol. 34, pp. 187–203, jan 2016. [Online]. Available: https://doi.org/10.1016%2Fj.jvcir.2015.11.002

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

Ren, S., He, K., Girshick, R., and Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., Eds. Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region/proposal-networks.pdf

Rikus, S., Hoffmann, D. S., Ungureanu, T., Rommerskirchen, D. S., and Plesker, M., "Auskunft über verfügbare parkplätze in städten," 2015.

Salpietro, R., Bedogni, L., Felice, M. D., and Bononi, L., "Park here! a smart parking system based on smartphones' embedded sensors and short range communication technologies," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, dec 2015. [Online]. Available: https://doi.org/10.1109%2Fwf-iot.2015.7389020

Satellite Imaging Corporation. "Satellite sensors." 2017. [Online]. Available: http://www.satimagingcorp.com/satellite-sensors/

Schölkopf, B. and Smola, A. J., *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Schäfer, R.-P. and Thiessenhusen, K.-U., "About traffic information system by means of real-time floating-car data," in *Institute of Transport Research.* DLR (German Aerospace Center), Institute of Transport Research, 2002.

Seo, Y.-W., Ratliff, N. D., and Urmson, C., "Self-supervised aerial image analysis for extracting parking lot structure." in *IJCAI*, 2009, pp. 1837–1842.

Serra, J., *Image analysis and mathematical morphology, v. 1.* Academic press, 1982.

Serra, J., "Introduction to mathematical morphology," *Computer Vision, Graphics, and Image Processing*, vol. 35, no. 3, pp. 283–305, sep 1986. [Online]. Available: https://doi.org/10.1016%2F0734-189x%2886%2990002-2

Shu, M. and Du, S., "Geoscene-based vehicle detection from very-high-resolution images," in *2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA).* IEEE, jul 2016. [Online]. Available: https://doi.org/10.1109%2Feorsa.2016.7552816

Simonyan, K. and Zisserman, A., "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2015.

Song, C., Qu, Z., Blumm, N., and Barabasi, A.-L., "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, feb 2010. [Online]. Available: https://doi.org/10.1126%2Fscience.1177170

Streetline Inc. "The streetline solution." 2017. [Online]. Available: https://www.streetline.com/how-it-works/

Teng, X., Cao, L., Wang, C., and Chen, Z., "Vehicle detection from remote sensing image based on superpixel segmentation and image enhancement," in *Proceedings of International Conference on Internet Multimedia Computing and Service.* ACM, 2014, p. 140.

Theano Development Team. "Convolutional neural networks (lenet)." 2010. [Online]. Available: http://deeplearning.net/tutorial/lenet.html

Trier, O. and Taxt, T., "Evaluation of binarization methods for document images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 312–315, mar 1995. [Online]. Available: https://doi.org/10.1109%2F34.368197

True, N., "Vacant parking space detection in static images," *University of California, San Diego*, vol. 17, 2007.

Tuermer, S., Kurz, F., Reinartz, P., and Stilla, U., "Airborne vehicle detection in dense urban areas using hog features and disparity maps," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 6, pp. 2327–2337, 2013.

Tupin, F., Maitre, H., Mangin, J.-F., Nicolas, J.-M., and Pechersky, E., "Detection of linear features in sar images: Application to road network extraction," *IEEE transactions on geoscience and remote sensing*, vol. 36, no. 2, pp. 434–453, 1998.

Wang, X. and Hanson, A. R., "Parking lot analysis and visualization from aerial images," in *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on.* IEEE, 1998, pp. 36–41.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.