Institute for Visualization and Interactive Systems

University of Stuttgart
Pfaffenwaldring 5a
D–70569 Stuttgart

Master's Thesis Nr. 3522

Extending Parking Assistance for Automotive User Interfaces

Radu Georoceanu

Course of Study:	Infotech
Examiner:	Prof. Dr. Albrecht Schmidt
Supervisors:	Dipl.-Inf. Bastian Pfleging
	M.Sc. B.Eng. Philipp Mayer
Commenced:	2013-07-09
Completed:	2014-01-08
CR-Classification:	H.5.2, I.5.5

Abstract

Nowadays the trend in the automotive industry is to integrate systems that go beyond the scope of just maneuvering the car. Navigation, communication, and entertainment functions have become usual in most cars. The multitude of sensors present in vehicles today can be used to collect information that can be shared with other drivers in order to make the roads safer and cleaner. A more troubling issue that affects drivers is the search for free parking spots, because of the time waste, fuel consumtion and effort. There are already solutions available that try to help drivers diminish these problems, like crowdsourcing smartphone apps, but they are still far away from being a reliable solution. The overall goal of this thesis is to find new ways of providing parking information to drivers. This information is collected from vehicles which are equipped with latest sensoric hardware capable of detecting parking spaces while driving and distribute these information to the cloud, sharing it with other drivers using smartphones or vehicle's integrated displays. Though the idea is simple, there are many challanges that need to be addressed. The thesis will also look into ways of improving parking surveillance for vehicles to make them less susceptible to vandalism and thefts, by using latest vehicle-integrated video camera systems. A study will be made to see what information drivers want to have related to parking and how this information can be displayed to them. Further, a cloud based-implementation of such a system will be presented in detail and an evaluation will be made to see how the system behaves in the real world.

Kurzfassung

Der aktuelle Trend der Automobilindustrie ist es Systeme zu integrieren, die über das Ziel hinausgehen ein Fahrzeug lediglich zu fahren. Navigations-, Kommunikations- und Entertainmentfunktionen sind inzwischen üblich in vielen Fahrzeugen. Die Vielzahl an verfügbaren Sensoren, die heutzutage in Fahrzeugen verfügbar sind, ermöglichen es Informationen zu sammeln welche mit anderen Fahrern geteilt werden können, um Straßen sicherer und sauberer zu machen. Ein nervenauftreibendes Problem, welches viele Fahrer aufgrund des Zeitverlustes, des Benzinverbrauchs und des Aufwands beeinflusst, ist die Suche nach freien Parkplätzen. Es existieren bereits Lösungen, diese Probleme für den Fahrer verringern, wie z.B. Crowdsourcing Smartphone Apps, aber diese sind immer noch weit davon entfernt zuverlässige Lösungen darzustellen. Das übergreifende Ziel dieser Thesis ist es neue Möglichkeiten zu finden dem Fahrer Parkinformationen zur Verfügung zu stellen. Diese Informationen werden von Fahrzeugen gesammelt, welche mit der neuesten Sensorik ausgestattet sind, die es ermöglicht während der Fahrt Parkplätze zu detektieren und diese Informationen über die Cloud zu verteilen und somit mit anderen Fahrern über Smartphones oder integrierte Displays zu teilen. Obwohl die Idee recht einfach ist gibt es viele Herausforderungen, die bewältigt werden müssen. Dazu wurde auch eine Studie ausgetragen, um zu untersuchen welche Informationen Fahrer im Bezug auf Parken gerne zur Verfügung hätten und wie diese Informationen ihnen angezeigt werden können. Weiterhin werden in dieser Thesis Möglichkeiten evaluiert die Überwachung von Fahrzeugen durch die Verwendung von in Fahrzeug integrierten Videosystemen zu verbessern. Eine Cloud-basierte Implementierung des beschriebenen Systems wird im Detail präsentiert und eine darauf basierende Evaluierung vorgestellt, um zu sehen wie sich derartige Systeme in der realen Welt verhalten.

I want to gratefully and sincerely thank for their contribution in realizing this thesis to...
Prof. Dr. Albrecht Schmidt, Dipl.-Inf. Bastian Pfleging and M.Sc. B.Eng. Philipp Mayer.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Interaction in automotive environments has broadened beyond the typical tasks that are required to maneuver a car. Current cars offer a multitude of features and functions from navigation and assistance to entertainment and remote communication. The rise of new technologies supporting vehicle intelligence (sensors, networks, computers) brought all this features closer to the public, allowing for safer and more comfortable driver experiences.

The technological evolution of driver assistance systems has pushed the automotive industry closer to autonomous driving. Many driver assistance systems are available in production vehicles nowadays. Adaptive cruise control uses radar sensors to permanently scan the carriageway, adapting the speed and maintaining the distance to the lead vehicle. Video camera systems are being used in lane departure warning systems, that can alert the driver if this occurs and than automatically steer the car back into the right lane. The traffic jam assist system allows when travelling in heavy traffic at speeds of up to around 40 km/h, to move with the other traffic and stay relaxed while passing through the congestion. It automatically keeps the desired distance from the vehicle in front by controlling speed independently, braking down to a standstill and even actively controlling the steering. Research is being made also for systems that can detect driver drowsiness and can send a proper alert to the driver.[1] With the help of ultrasonic sensors, the automatic parking assist systems assist the driver with the parallel parking maneuvers by automatically controlling the steering wheel. This systems have also been extended to park the car without the driver even being inside it, and only sending the necessary commands through a smartphone.[2]

Internet and cloud applications are also making their way into the vehicles. By using an UMTS module and a SIM card, many Internet services can be provided, like Google Earth, weather, news and music stream that are displayed using the car's head unit (Fig. 1.1). On top of

[1] http://www.bmw.com/com/en/insights/technology/technology_guide/index_category.html?code=4&view=Safety

[2] http://www.bosch.com/en/com/boschglobal/automated_driving/technology_for_greater_safety/pagination_1.html

Figure 1.1: Car head unit integrated in vehicle's dashboard provides cloud based services (Source: B.Pfleging)

these, the car can be used as an Internet hotspot so that all passengers can have access to Internet inside the vehicle. The BMW Connected Drive[3] offers a wide range of intelligent services and apps that provide the driver and passengers with information and entertainment during the journey. Among these are real-time traffic information, intelligent emergency call, remote services, Internet and online entertainment. Daimler is also focusing on developing new communication technologies. The cloud-based telematic strategy @yourCOMAND from the F 125 research vehicle provides a preview of the Mercedes-Benz infotainment systems of the future. Authorised persons can access information, such as fuel level, next service date or fuel consumption via the vehicle's own homepage at any time. Similarly, the navigation or entertainment system or the desired temperature for each seat can be preset via a PC or smartphone[4].

Smartphones have evolved very rapidly in the last few years, which made it possible to provide powerful functionalities like networking, web surfing, video-streaming and games. The processor power and memory size are now similar to that of laptop computers which allow for the introduction of these devices into new contexts. Integrating smartphones into cars by means of car docks (Fig. 1.2) offers the possibility of introducing new infotainment and driver assistance services.

[3]http://www.bmw.com/com/en/insights/technology/connecteddrive/2013/index.html
[4]http://www.intellimec.com/myconnectedcar/mobile-media/mercedes-benz-infotainment-system/l

Figure 1.2: Car dock supporting the smartphone inside the vehicle

1.1 Problem statement

One of the main challanges that car drivers have to face is the search for a parking space. They most likely want to park their cars as close as possible to their destination. However, in big cities, at certain times during the day this is not always possible so they have to drive around and search for a free parking space. This is not a pleasent situation, especially when they do not have sufficient time at their disposal. One study has shown [Shoup, 1997] that the average search time for parking is nearly 10 minutes during evening hours and around 9 minutes on average. It is further shown that around 30% of the traffic is parking traffic, that means people that just drive around and search for a parking sopt. Adding all vehicles on the street results in a enormous amount of excess driving leading to air pollution and traffic congestion. Another study [Caliskan et al., 2006] has shown that in the district of Schwabing of Munich, the annual damage caused by parking search consists of around 20 million Euros of economical damage, 3.5 million Euros for gasoline and diesel and 150 000 hours of wasted time. Clearly this has a significant impact in urban areas with regard to economical and ecological aspects and finding solutions that help eliminate or diminish these problems is desired.

Another troubling issue related to parking is that especially in unsupervised parking locations cars are possible targets for acts of vandalism or even stealing. A report done by the US Department of Justice says that: *"Car-related thefts are among the most common offenses calling for a police response."* There are many reasons for why thiefs do such acts: joyriding, car use, disassembiling cars for spare parts or even stealing items from inside the vehicle, such as radios, batteries or wheels. Most thefts happen when cars are parked on the streets or on the owner's property, the main reason for this being that this is where the cars are usually parked. Even so, the same report mentions that the risk is higher in dedicated parking facilities because of the poor security. Many cars are parked in poor lighting or blind spots where cars

the cars can not be supervised. All these problems call for finding better solutions for car surveillance during parking hours.

1.2 Proposed solution

This thesis looks into existing ways that support driver assistance focusing on enhancing parking assistance and how this can be extended to other parking functions. It will show how parking information can be provided to drivers in order to overcome the main challanges of parking space search and also on how to improve parking surveillance. One goal of this thesis is to find new ways of providing this parking information to drivers to help them find a suitable parking space before they arrive at their destination so that time and fuel consumtion are reduced. The focus will be on vehicles which are equipped with latest hardware and software technologies capable of acquiring parking space information. This information can be collected, processed and distributed to the cloud so that is shared with other drivers using smartphones or vehicle's integrated display. Though the idea is simple, there are many challanges that need to be addressed. We will look how these information can be collected and what kind of processing needs to be done in order to provide only reliable information to drivers. Parking space information can be combined with open source map-services data in order to bring even higher value to this concept. The second goal is to find new ways of improving parking survaillance of the vehicles using the latest available technologies. We will look into how multicamera systems integrated in the vehicles can help in this matter. The thesis also focuses on the end-users and will try to find the best way for making the information available to them. A study will be made to see what information drivers want to have related to parking and how this information can be displayed to them. Further, a cloud-based implementation of such a system will be presented in detail and an evaluation will be made to see what are the advantages of using such a system compared to other solutions.

1.3 Overview of the following chapters

Chapter 2 looks into similar work that has been done so far, related to vehicle cloud-based application and parking support systems. Chapter 3 presents the basic technologies on which our proposed system relies. Chapter 4 describes the basic concept of our proposed solution for a parking space search system. Chapter 5 includes a study in which we try to understand what are the user needs related to parking space search. For this, several user interface prototypes as well as a focus group report will be presented. Chapter 6 contains a detailed description of the prototypical implementation of our system, starting from the acquisition of data, distribution to the cloud server, necessary processing steps and the end-user interface. Chapter 7 shows several evaluations of the built system. We will analyze the costs of the system, show what are the main challenges and how we overcome them and test several algorithms for filtering the acquired data. Chapter 8 draws a final conclusion for this work.

2 Related Work

This chapter will take a short glimse into state of the art work related to cloud applications in the automotive environment. First, general cloud applications will be presented and afterwards the focus will be on applications that provide parking support. Afterwards, will present a short discussion about how our work differes from the presented ones.

2.1 Smartphones and The Cloud in the Automotive Environment

The rise of the Internet and wireless communication technologies, combined with the processing power of the smartphones, made it possible to connect vehicles to the WWW world. More and more applications are developed that provide information or entertainment to drivers and passengers, either by using direct Internet connection in the vehicle, or by using smartphones.

Bose et al. [Bose et al., 2011] presents the terminal mode concept. In terminal mode, the smartphone is the application platform for the automotive environment, and the car head unit is responsible for user input and output. The smartphone hosts and runs all applications and services and also is responsible for transferring data to the cloud while the head unit provides the physical I/O mechanisms through which the driver or passenger accesses the smartphone. The user interface hardware can include output devices such as one or more displays, audio playback, and haptic feedback systems as well as input devices such as touch screens, buttons, multifunctional knobs, and microphones. By treating the smartphone like an application platform, Terminal Mode makes it much easier to add new capabilities to the car head unit by simply upgrading the smartphone. One of the advantages of the Terminal Mode concept is that it provides the capability of running the latest applications and services that connect the car to the environment. The user gets the same experience as when using an application running natively on the head unit but without any of the associated disadvantages.

Another use of the smartphone in the automotive environment is to collect data from the car sensors and electronic components and to display the readings through the user interface.

Torque Pro[1] is an Android application that can show what the car is doing in real-time, get On-Board Diagnostic (OBD) fault codes, car performance data, sensor data and more. It is a vehicle performance diagnostics tool and scanner that uses an OBD-II (improved version of OBD) Bluetooth adapter to connect to the engine management ECU.

Same idea of collecting vehicle data is presented as SMaRTCaR [Campolo et al., 2012]. This is a software/hardware integrated platform which can retrieve different types of vehicle and environment related data and remotely transfer them. This is realised by an interconnection with the vehicle communication bus and augmented through some dedicated hardware to retrieve data from external sensors, not directly connected to the CAN bus. Its software core is based on a mobile application specifically developed for Android smartphones and the main functions are triggering the retrieval of data, data visualization, and remote data transfer through internet connectivity. For the aquisition and preprocessing of data, an Arduino USB-enabled microcontroller board is used. Kinematics-related data are supplied by the internal vehicle CAN bus. The smartphone is responsible for the data visiaulzation and transmission. An important aspect presented in this paper is the tagging of data. The information gathered by the microcontroller is complemented with the time and GPS position information gathered from the smartphone. After that, data are tagged according to their latency and delivery requirements, with different priorities.

The OBD-II protocol is also used in [Zaldivar et al., 2011] for presenting a fully automated accident detection concept. Accident detection is done by analyzing parameters provided through the OBD-II interface (e.g. airbag triggering, acceleration). This data is combined with the smartphone GPS readings in order to locate where the accident has happend and then sent through the smartphone's Internet connection to an emergency services database or to other third parties defined by the user. The whole process, including reading critical data from the vehicle's bus, delivering information about the accident through both e-mail and SMS, and finally starting an emergency call, takes less than three seconds. Such a system is already integrated into production vehicles, like BMW's eCall[2], that can report parameters like number of car occupants, number of airbags deployed, the precise location and severity towards a BMW call center in case an accident occurs.

2.2 Providing Parking Space Information

There are different ways in which parking space information can be collected and provided to drivers in order to assist them in their search for a free parking space.

One way is to collect third party information and make it available to drivers. In US, Audi has launched an update for their Audi Connect interface that helps drivers find a suitable parking space near a location [3]. The parking locations information is taken from parking garages that want to be part of this project and contribute by providing their availability information

[1] https://play.google.com/store/apps/details?id=org.prowl.torque&hl=ro
[2] http://www.euroncap.com/rewards/bmw_assist_advanced_ecall.aspx
[3] http://www.automotto.com/audi-connect-best-place-park-vehicle.html

to Audi. Another example is the collaboration between Cisco and Streetline, where drivers from several cities in US can search for a free parking space directly on their smarthpone, by using the Parker app[4]. Magnetometer and light sensors are installed in the asphalt to detect whether a parking space is free or occupied. By using a Cisco Wi-Fi mesh network, these sensors can send their state to the cloud. Users can interact with the data provided by these sensors through the app and are able to find free parking spaces in real-time. Navigation to these parking spaces is also possible and also drivers can mark where they parked the car. BestParking[5] is another example app that can steer drivers towards the prefered parking garages.

Many applications use crowdsourcing as their source of information, by using the collective intelligence of their users. This idea has been introduced also for parking search applications, like Google's OpenSpot[6]. This is an Android application that helps drivers find and share parking spots in their vicinity. Chen et al. [Chen et al., 2012] presents a similar system, together with its challenges and drawbacks. Another interesting idea that is using crowdsourcing data is bidding for parking. Applications like Parkonaut[7] are encouraging users to list their personal parking spaces inside the app so that other people can use them. For a small amount of money, users can make bids on a listed parking space and the winner can use that space for a certain amount of time, until the owner needs it back again.

Global Positioning System (GPS) information can also be used for the purpose of determining parking locations, by GPS trajectories. For a study [Montini et al., 2012], a large GPS data set has been collected in two cities, Zurich and Geneva over a period of two years consisting of GPS positions with timestamps. Applying different algorithms, parking search traces are separated from other driving and so, parking locations have been determined. A similar concept is also presented in [Yang et al., 2013]. Large GPS data sets obtained from vehicles are transformed into parking space locations. The concept is based on User Generated Geo-Content, so that users are able to insert their trajectories into the system and this is able to determine if a parking space is an on-streetparking space or parking zone.

Chen et al. [Chen et al., 2010] proposes a vision-based system for counting the number of vehicles and determining the parking space availability in outdoor parking lots. This system uses multiple cameras for merging multiple scenes by applying an affine transformation. Then using a set of features like color, position and motion vehicles can be tracked and thus the availability of parking spaces is determined.

The idea of collecting parking space information from the vehicles has already been proposed. ParkNet [Mathur et al., 2010] is a mobile system comprising vehicles that collect parking space occupancy information while driving by. Vehicles are equiped with GPS receivers and ultrasonic sensors to spot free parking spaces. A central server collects the data and builds

[4] http://reviews.cnet.com/8301-13746_7-57556954-48/parker-smartphone-app-enables-realtime-parking-search/
[5] https://play.google.com/store/apps/details?id=com.bestparking&hl=en
[6] http://www.geek.com/android/google-releases-open-space-a-crowdsourced-parking-spot-app-for-android-1269214/
[7] https://play.google.com/store/apps/details?id=de.parkonaut&hl=en

a real-time map of parking availability. Users can then query the server to receive available parking spaces information.The location coordinates provided by a GPS receiver are usually accurate to 3 m. This paper introduces an environmental positioning concept to achieve a greater accuracy for matching the parking slots. For this, as a first step GPS error correlation is calculated, by means of location-tagging of certain objects on the road. Over multiple runs, video traces are collected and analyzed. Fingerprinting the environment by relying on features in the sensor trace that are produced by fixed objects in the environment, provides a possible means to improve location accuracy beyond that provided by GPS alone. However, this requires multiple traces from a certain street, from which the locations of objects that are very likely fixed can be determined. After this, the estimation of the GPS error is done by comparing the reported location of the pattern produced by a series of fixed objects to the known location of the same pattern. The offset between these two gives the error in the reported location. It was proved that parking spot counts are 95% accurate and occupancy maps can achieve over 90% accuracy. Other researches address the same problem of mapping street parking spaces using ultrasonic sensors. It is possible to use data from these sensors to classify parking spaces into legal and illegal. [Coric and Gruteser, 2013] Multiple passes of a vehicle on the same street provide the parking occupancy data of that street. To decide if a parking space is legal or illegal, the algorithm uses the following idea: spaces that almost never have parked cars are probably not legal parking spots (fire hydrants, garages, bus zones etc.); spaces which are frequently occupied almost every time are likely to be valid parking spaces. Such information can be inferred by aggregating available time series.

2.3 Supporting Parking Surveillance

The standard security mechanisms present in cars, like locks and alarm systems may not be sufficient for protecting the car against experienced thiefs or against vandalism. Also, old camera-based surveillance systems that use human operators for monitoring the images by looking at several monitors, is not an ideal solution, because it has to rely on human operation and finding an automated system that does this is desired.

Foresti et al. [Foresti et al., 2004] presents an automatic visual-based surveillance of parking lots. Images are observed for in order to get and analyze specific events that occur. These events are classified into standard or dangerous according to predifined motion models. Similar event recognition system is explained by Ng et al. [Ng and Chua, 2012]. Here, four main processes are presented: object tracking, event representation, training and event recognition. Object tracking involves the detection of the moving object and the collection of spatial-temporal information. Event representation process is in charge of determining the feature vector from motion trajectories and the training process computes the representation of the known-events. The event recognition identifies each type of event.

Another approach is to also use acoustic sensors, on top of visual sensors. Such a system is presented by Na et al. [Na et al., 2009] which uses microphone sensors to localize acoustic events such as car alarms. When such an event is localized, the cameras are directed towards that location.

2.4 Discussion

Even though there are quite a few applications that support parking search and parking surveillance, each of these have their own drawbacks and with the constant evolution of the Internet and vehicle technologies, there is certainly room for improvement.

Using parking space information from parking lots and garages, like Audi's parking recommendation provides too little information for drivers. This kind of systems can only show if a parking garage has available spots or not and the location. And what about on-street parking? Parker app might be a better solution, but mounting sensors into every parking spot across a big city involves enormous costs. Same problem occurs when using video-based systems for determing parking availability. These are only feasible to be mounted in parking lots and garages, doing this for every street again involves high costs. Crowdsourcing can be the way to go, but like Chen et al. [Chen et al., 2012] concludes in their paper, users that contribute with providing parking information not always give the most accurate data. Misled or even bad intended people can decisevely lower the quality of this data. The participation rate is a very important factor when designing such a crowdsourcing system, much more important than the amount of information that is provided by each individual. If the rate is too low, some other mechanisms need to be introduced in order to compensate for the lack of data sources. Acquiring data directly from vehicles can be the most convinient and precise way to provide parking information. Everything can be done automatically, without affectiong drivers in any way and with high accuracy for the provided information.

Our proposed system uses vehicle sensor data and GPS information for providing parking space information. Like the ParkNet system this information is collected by using ultrasonic sensors whicle driving by. Even though very high accuracy can be achieved, ParkNet uses object recognition mechanisms for fingerprinting the environment, which requires quite extensive processing and also a lot of passes over the same area. For determining where the legal parking spaces are located, we propose collecting park-in and park-out messages from vehicles. Using latest vehicle automatic parking technologies this can be done and consequently, a precise map of legal parking spaces can be built. Also, we show that by using map services data, the quality of the information can be even further increased.

Regarding vehicle parking surveillance, most of the existing systems are based on video cameras that need to be mounted in parking lots or parking garages. Our proposed system uses own vehicle cameras for monitoring the vehicles while they are parked. Doing this, increased security is achieved even for on-street parking or any other location where vehicles are parked. There is no dependency on separate systems, so every vehicle owner can secure its vehicle without any other help.

3 Technical Background

For better understanding the proposed system, this section presents the main technologies on which our system relies.

3.1 Car Sensors

The modern day car means much more than an engine with four wheels that can be used to move from A to B. It has to provide comfort, entertainment, communication, efficiency and most important, safety. Integrating all these functions would not be possible without the use of electronics and sensors. The most common sensors present in vehicles are: vehicle dynamics sensors, ultrasonic sensors, radar sensors, lidar sensors and vision based sensors.

3.1.1 Vehicle Dynamics Sensors

Main functions of the dynamics sensors are to identify the position of certain vehicle components relative to the three orientation axes. The main sensors are [Winner, 2012a]:

- *Wheel sensors*: can determine wheel movement parameters like speed, accelaration and direction. This are one of the most important parameters since they determine vehicle's deceleration and stability.
- *Steering wheel angle sensors*: determine the position of the steering wheel in order to control vehicle's stability.
- *Rotation sensors*: determine the vehicle's body rotational movements in all three spacial axes. Electronic stability control (ESP) system uses this information.

- *Acceleration sensors*: these measures the gradient driving forces on the main orientation axis. Also the radial acceleration can be determined for the measurement of the road inclination.
- *Brake pressure sensors*: are used to determine the driver's braking force, by measuring the pressure in the master brake cylinder. In more complex vehicle dynamics control systems even the pressure in each wheel brake cylinder is measured.

3.1.2 Ultrasonic Sensors

Piezoceramic ultrasonic sensors have been available in production vehicles since 1992 [Winner, 2012a]. Today's sensors are very compact and can be integrated even in the painted vehicle bumpers. Improvements are still being made regarding measuring distance reduction, signal to noise ration and better self diagnosis ability. Ultrasonic sensors are being used in parking assistance systems for measuring the parking spaces to determine if there is enough space to park. They are mounted on the left and right sides of the vehicle as well as incorporated in the front and back vehicle's bumpers. They can detect parked vehicles, their lateral boundaries and corner positions, can measure the parking space in depth and also possible obstructions on the path of the vehicle. Automatic parking systems are already on the market which use these sensors in order to automatically steer the vehicle into the desired parking space. A newer application where ultrasonic sensors are used is the Side View Assist (SVA). When traveling with speeds up to 140 km/h and initiating an overtake manouver, the driver is informed if another vehicle is in his "blind spot" so that he knows if he can do the manouver.

3.1.3 Radar Sensors

Radar (Radio Detection and Ranging) is a technology that can measure distance to objects by using radiowaves. This technology first appeared in vehicles in 1998 for ACC [Winner, 2012a]. Another application where it is currently used is collision detection. Current world-wide regulated frequency for automotive radar is 76.5 GHz, but also other frequency ranges are available (24.0-24.25 GHz, 77-81 GHz etc.). Though a lot of effort is put into lowering the costs of radar sensors, these are still quite high because of the strict requirements that automotive applications need: smaller frequency spacing, smaller Doppler frequencties, multiple target capability and lower sizes. The main competitor of radar, Lidar, has also made a lot of progress in the last few years. Nevertheless, radar has its benefits: ability to measure the Doppler effect and a high weather robustness.

3.1.4 Lidar Sensors

LIDAR (Light Detection And Ranging) is an optical measurement method for the detection and measurement of the distance to objects [Winner, 2012a]. The principle is similar to radar, only instad of radiowaves, Lidar uses light. Current development directions are targeting

higher power with reduced instalation space and costs. The trend is to find more and more cost effective sensors and consequently Lidar sensors come more and more to the front.

3.1.5 Vision Based Sensors

An image sensor builds light patterns from multidimensional measurement signals. From these primary measurements, secondary measured values are then extracted by means of image analysis, such as the positions, the velocities, or the type of objects of interest [Winner, 2012a]. Since the prices of cameras are getting lower and lower, they become more attractive for automotive applications. The advantage compared to other types of sensors is that they provide the most comprehensive representation of information. They are superior in potentiol to other sensors, but the light patterns information acquired needs to be translated to 3D information using complex image processing algorithms. In contrast, lidar and radar provide these information directly. Table 3.1 shows a short comparision between vision, lidar and radar sensors.

		Vision	**Lidar**	**Radar**
Wavelength [m]		10^{-7}-10^{-6}	10^{-6}	10^{-3}-10^{-2}
Weather conditions dependance		yes	yes	low
Resolution [no. of readings]	horizontal	10^{2}-10^{3}	10^{2}-10^{3}	10^{1}-10^{2}
	vertical	10^{2}-10^{3}	10^{1}-10^{2}	10^{1}
	chronological	10^{1}-10^{5}	10^{1}	10^{1}
Primary measurements	Position	-	+	+
	speed	-	-	+
	Brightness pattern	+	+	-
Application behaviour	Object detection	+	+	+
	Object recognition	+	+	+/-
	Road recognition	+	+/-	-
	Traffic sign recognition	+	-	-

Table 3.1: Comparison of different sensor types and their behaviour in different automotive applications (Source: [Winner, 2012a])

3.2 Vehicle Electronic Networks

Electronics represent more than 25 percent in the production value of a car. The complexity of such automotive systems is growing higher with the integration of new innovative functions. More and more mechanical and hydraulic systems are being replaced by electronic systems (e.g. Drive by Wire). We have reached the point where we find more than 100 electronic control units (ECU) and more than 4 kilometers of wire in a single vehicle [Kim et al., 2011].

3.2.1 Architecture

The architecture of an automotive electronic system is depicted in Fig. 3.1. It is actually an in-car network that consists of ECU's that need to communicate with each other using different bus systems, in order to complete their functions. Automobile electronics can be split in three domains: power train management, body electronics and information processing. These categories require different levels of functional safety and reliability. The growth in the number of ECU's becomes an increasing problem for the bus systems, that have to provide determinism and massive bandwidth capabilities.

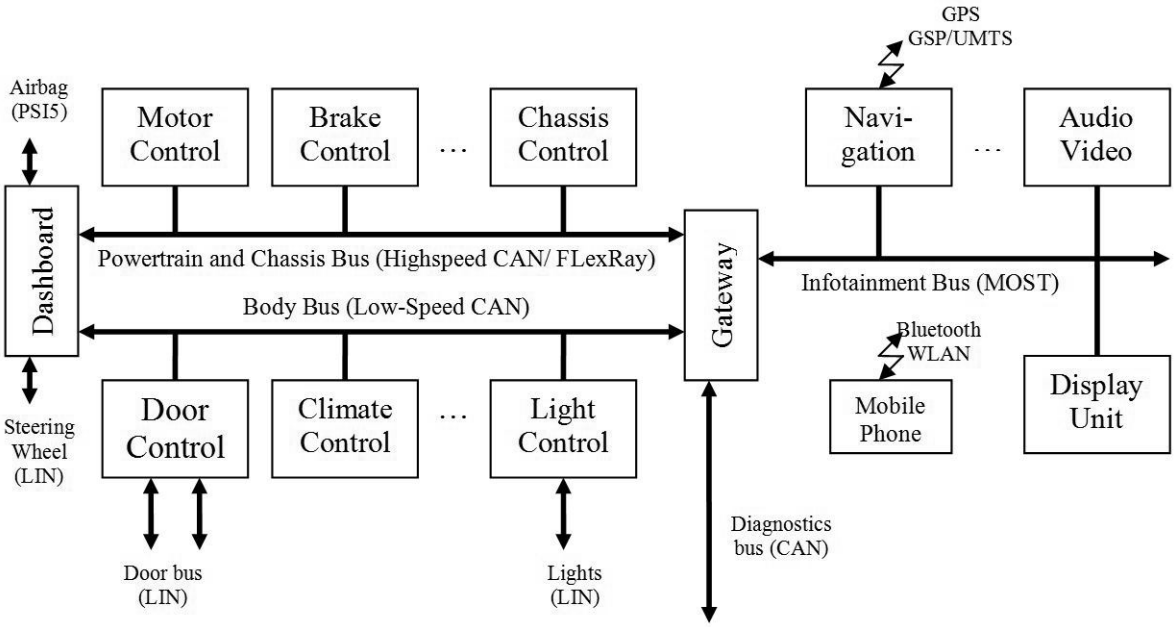

Figure 3.1: Electronic System in Modern Vehicles - ECU's communicate with each other through different bus systems (Source: [Zimmermann and Schmidgall, 2010])

A car is supposed to work in severe conditions like the temperature ranges within the engine compartment, high electromagnetic interferences (EMI), vibrations etc. For the electronics in a car all these factors can have negative effects regarding reliability. Safety critical systems are those systems that should never fail, even in the presence of this negative factors, or life threatening consequences may follow. That is why the car manufacturers have to comply with standards in order to achieve the needed safety requirements. The ISO 26262 [Leitner-Fischer and Leue, 2011] standard defines the functional safety aspects throughout the life-cycle of electric and electronic systems found in the automotive industry. The standard defines four safety levels named Automotive Safety Integrity Levels (ASIL),which range from A (lowest safety-critical) to D (highest safety-critical). The levels are based on the probability of exposure, controllability and severity. A certain product has to be developed according to the ASIL that has been specified beforehand. The main goal is to ensure functional safety by recording all the steps in the development process.

3.2.2 Bus Systems

As depicted in Fig. 3.1, there are different bus systems that handle different vehicle functions. FlexRay[1] is the latest and most powerful one, created for handling all the current needs of the automotive industry. Even though this standard is already available, old standards like Local Interconnect Network (LIN) and Controlled Area Network (CAN) are still being used mainly because of the lower costs. A short comparison with the previous standards is shown in Table 3.2, according to [Talbot S.C., 2009].

	Bandwidth	**Messaging**	**Error Management**	**Physical**	**Topologies**	**Applications**	**Cost**
LIN	max 10 kBit/s	Deterministic, static scheduling	CRC Frame Field, ID Field Parity Bits, Diagnostic Frames	1-Wire	Master + Slaves	Displays, Lighting, Alarm Systems, A/C, Seat & Mirror Adjustments, Power Windows, Windshield Wipers, Headlamps	Low
CAN	max 1 MBit/s	Non-deterministic, Event-triggered	CRC Frame Field, ID Field Parity Bits, Diagnostic Frames	1-Channel, 2-Wire	Bus		Medium
TTCAN	max 10 MBit/s	Deterministic, Event-triggered & Time-triggered			Bus	Engine, Transmission, Braking, Steering, Suspension, Assistance, Safety, Diagnostics	High
FlexRay	max 2x10 MBit/s	Deterministic, Event-triggered & Time-triggered	2 CRC Frame Fields, Bus Guardian	2-Channel, 2-Wire	Bus + Star		

Table 3.2: Comparison of different bus systems. The new FlexRay protocol is certainly the best, but the old ones have the lower costs advantage
(Source: [Talbot S.C., 2009])

3.3 Parking Assistance Systems

Parking assistance systems can help drivers to find a suitable parking space, or to receive additional warnings while parking and even do the necessary parking maneuvers. In this chapter we will try to understand parking assistance systems that are available in vehicles today.

[1] www.flexray.com

3.3.1 Degrees of Parking Assistance

Many possible forms of parking assistance systems are now avalable in production vehicles. Among the challanges drivers have to face when parking is the vehicle's front and rear geometry. Aerodynamic design related requirements and windows can limit the driver's field of view. To compensate for this, assistance systems based on environmental sensor data have been developed. These systems can be divided into [Winner, 2012b]:

- *Informative assistance systems*: include the systems in which distance to objects notify the longitudinal direction, as well as those for pure parking space measurement.
- *Guided parking assistance*: concrete maneuver indications are given based on environment information. These include cameras with superimposed guides or parking assistant that suggests steering maneuvers.
- *Semi-automatic parking*: system automatically controls steering and the driver only has to control the brake and acceleration.
- *Fully-automated parking*: the system is in charge of the entire vehicle guidance.

3.3.2 Requirements for Parking Assistance Systems

Depending on the specific system and the degree of assistance some requirements must be specified when designing such a system. First, the system must be for everyday use. An easy to use interface must be available and it must be usable in real situation. Regarding the environment sensors, the following requirements can be formulated [Winner, 2012b]:

- high robustness against environmental influences like pollution and precipitations;
- high accuracy of detected parking spaces;
- low signal distortion;
- low cost;
- minimum space requirement.

3.3.3 Informative Assistance Systems

The most common parking assistance is the ultrasonic-based parking aid. [Pruckner et al., 2003] The sensors are installed in the front and in the back of the vehicle in order to determine the distance to the nearest object. The distance is reported by an intermitent sound to the driver. The frequencies for front and rear warnings can be chosed differently so that the driver can distinguish them accordingly. Recently, acoustic warnings have been extended to visual output by using colored areas around the vehicle. For example, the area changes colour when an object is detected (see figure 3.2). Ultrasonic sensors are mostly used because of low system costs. Alternatively, short-range radar sensors can be used. These have the advantage that they can have a neutral design and can be hidden in the rear bumper and lights. However,

these are more expensive and are only used in combination with more advanced assitance systems like automatic cruise control.

Figure 3.2: Visual output of ultrasonic sensors shows different colours around the car according to how close the surrounding objects are located from the vehicle (Source: Bosch GmbH)

This category of Informative assistance systems also includes those that are able to detect suitable parking spaces when the vehicle passes around them. The driver is informed when a parking spot is detected, usually with using visual display. However, this only works at low speeds usually between 15 and 30 km/h [Winner, 2012b].

3.3.4 Guided Parking Assistance

These systems give explicit driving instruction to the driver in order to assist maneuvering the car into the parking space. Usually, guiding information is added on top of the reversing camera image. Wide-angle lenses are used for these cameras to cover a wide as possible range behind the vehicle [Vestri et al., 2005]. This system can be used to park in both perpendicular and longitudinal parking spaces. In the case of perpendicular spaces, contour of the vehicle as well as the predicted driving path can be displayed. For parallel parking spaces, labeled fields and auxiliary lines will be displayed. The following steps are necessary: survey of parking spaces, trajectory planning, continuous positioning, display driving actions.

The trajectory planning is composed of straight lines and circular arcs. It is important to take into account the driver reaction times. For the displaying of driving actions it is important to position the vehicle relative to the parking space so that the planned path can be determined precisely. This can be done by analyzing natural reference points in the environment, but due to their dynamic behavior are less suitable. An alternative is the so called odometry that is based on observation of the wheels. The distance traveled is calculated by counting the revolutions of the wheel. For a better quality of the localization, other internal vehicle information is used like steering wheel angle and yaw rate. Using these values, the vehicle behaviour can be mapped to a single track. The following information should be displayed to the driver: target steering angle or steering angle difference, direction, stop points, end of parking.

3.3.5 Semi Automatic Parking

In semi-automatic parking, the driver is relieved of the vehicle steering maneuvers, usually for lateral parking (figure 3.3). Instead of only showing guiding directions, the car is able to steer on its own. This is achieved by using Electric Power Steering (EPS). For such a system to be implemented, some strict regulations must be followed. The driver must be able at any given time to take control of the steering and also the system must be switched off if a certain speed limit is exceeded. Another aborting situation is when for some reason some of the signals are no longer received or are faulty [Jung et al., 2010].

Most semi-automatic parking systems are based on data from camera systems and ultrasonic sensors. Compared to ultrasonic based systems, optical systems show variable performance since they are dependent on the lighting and weather conditions. Semi-automatic parking systems require intensive communication for the components involved. This is usually implemented via the CAN bus. The relevant compontents are (see figure 3.4):

- ECU of the parking system;
- button to activate system;
- speed sensors on all wheels;
- steering angle sensor;
- acceleration sensors;
- ultrasonic sensors (side / front / rear);
- parking aid warning buzzer;
- electronic steering;
- brake control unit.

Figure 3.3: Vehicle doing reverse parallel parking maneuver with the support of semi-automatic parallel parking using ultrasonic sensors (Source: Bosch GmbH)

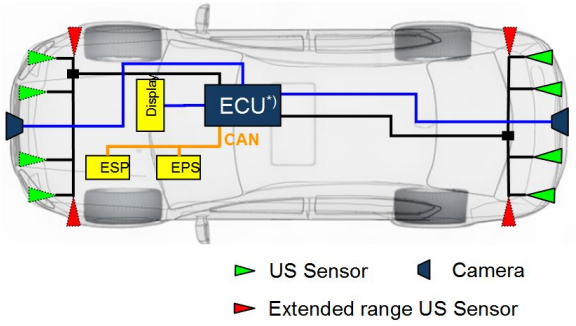

Figure 3.4: Vehicle "xray" presenting different components of the semi-automatic parking system (Source: Bosch GmbH)

3.3.6 Fully Automated Parking

The first automatic parking systems have already been developed[2]. The Fully Automated Parking Assist system developed by Bosch is used in combination with a parking space search system, being able to first detect a suitable parking space and then do the necessary steering manouvers to drive into the space. When parking, the driver only has to start the system by interracting with the user-interface and wait until the parking maneuvers are completed by the system.

3.4 Android

Android[3] is one of the largest mobile platforms powering millions of mobile devices all around the world. It is a powerful platform that gives the chance to everyone to create apps and games and distribute them freely to the marketplace. In the last years Android has had a really fast growing rate, every day another million users choosing to buy a device running on this platfrom. One of the main reasons for this is that it is open-source, its purpose being to encourage developers to build innovative apps.

Android provides a powerful development framework that makes it possible for every developer to develop a single app that can be installed across a wide variaty of devices. Android Developer Tools form a Java IDE that contain everything needed to develop a new app.

Google Play is the place where the Android apps are distributed and sold. The developer is in much control of how to realize this, being able to set the payment type, market segments, target devices, visibility.

[2]http://www.bosch.com/en/com/boschglobal/automated_driving/technology_for_greater_safety/pagination_1.html
[3]http://developer.android.com

3.4.1 Hardware Capabilities

Android devices have build-in sensors that provide different measurement capabilities. There are three main categories in which these sensors can be split:

- *Motion sensors*: measure acceleration and rotational forces along three axes. These include rotational vector sensors, gyroscopes, gravity and accelerometers
- *Environmental sensors*: measure environment parameters, like air temperature, air pressure, illumination and humidity. These include barometers, photometers and thermometers.
- *Position sensors*: measure the physical position of a device. These include orientation sensors, magnetometers and a built in GPS. This can easily provide to apps localization information like latitude, longitude and height.
- *Other sensors*: offer specialized functions. E.g. microphone, camera.

3.4.2 Android Architecture

Android is basically composed of the following layers:

- *Applications*: a developers work; written in Java and executing in Dalvik virtual machine.
- *Application Framework*: provides the support for the development of apps like managing the life cycle of an application, holding information about installed application packages, display management, managing telephony services and many more;
- *Libraries*: are implemented in C and accessed through the Applicatiion Framework. These include: standard C library (libc tuned for embedded devices), media libraries (audio/video playback, recording), Surface Manager (access to display subsystem), SGL (2D graphics engine), Open GL (3D graphics), FreeType (bitmap and vector font rendering), WebKit (web browser engine), SQLite (relational database engine);
- *Android Runtime*: each application runs in its own Dalvik virtual machine. This is optimized for low memory and processing requirements. Java byte code is converted to Dalvik Executable format (.dex);
- *Operating System*: it is Linux Kernel 2.6; manages system resources and handles hardware access.

3.4.3 Application Components

Android applications consist of components. These components facilitate re-use, meaning that one application can use the components of other applications. There are four component types:

- *Activity*: this is the most important type because it provides a graphical user iterface (GUI). It supports one particular task, e.g. reading an email;
- *Service:* supports long-running background tasks. It it has no GUI, being controlled only through the activities that communicate with the service;
- *Broadcast Receiver*: receive and react to broadcast messages, also with no GUI;
- *Content Provider*: provides data to applications, by storing it in local database or files.

3.4.4 Activity lifecycle

The activity life cycle is depicted in figure 3.5. State changes are signaled to activity via callbacks (onCreate(), onStart() etc.). Consequently, Android activities have four states:

- *Active/Running state*: this is the state when the activity is completely visible and the user can interract to it. It is in the foreground (on top of other activities) and has the focus;
- *Paused state*: the activity only partially visible, but not active and lost focus. The activity is completely alive, but it can still be killed by the system to free up memory;
- *Stopped state*: the activity is no longer visible and another one is in foreground. Though it is still alive and preserves its state, it is more likely to be killed by the system in order to free up resources;
- *Destroyed state*: this is the case when the activity no longer exists in memory.

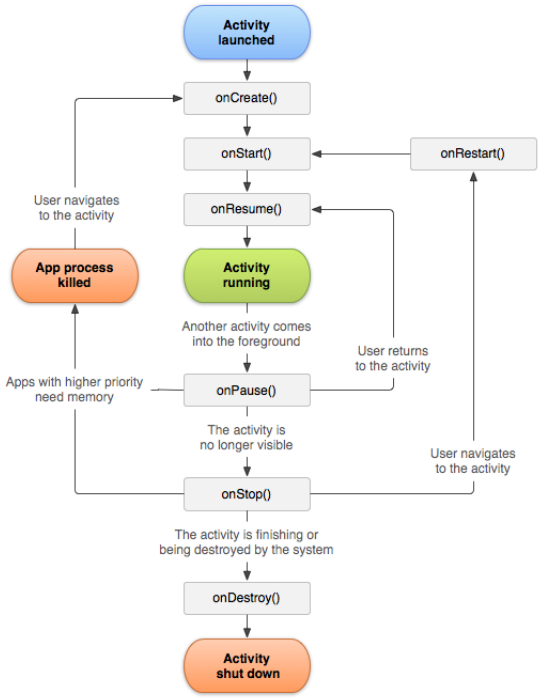

Figure 3.5: Android activity lifecycle consists of the four main states: running, paused, stopped and destroyed (Source: Android Open Source Project)

3.5 Cloud Computing

Cloud computing is a term wrongfully conceived as a metaphor for the Internet. However, it means much more than that. Cloud computing can be defined as a construct that allows users to access applications that reside at a different location than their computer or device [Velte et al., 2010]. "Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services" [Armbrust et al., 2010]. There is no unique way to implement a cloud solution. The infrastructure will depend on the application and on how the provider has chosen to build the solution. There are many vendors that provide cloud services. Among these are: Google (App Engine and Web Toolkit), EMC, Amazon (Elastic Compute Cloud, Simple DB, Simple Storage Service, CloudFront), Salesforce, IBM etc.

3.5.1 Cloud Computing Components

The simplified topology of a cloud solution is made up of three elements as illustrated in Fig. 3.6: clients, service providers and cloud providers [Armbrust et al., 2010].

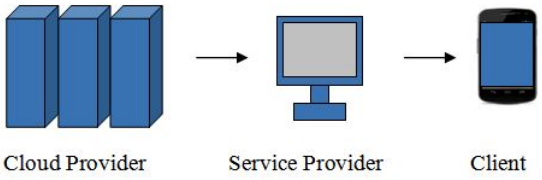

Figure 3.6: Users and providers of cloud computing and their relationship - service providers buy platform from cloud providers in order to be able to offer their services to clients

3.5.1.1 Clients

Clients are the end systems that interact with the cloud. They can be desktop computers, laptops, smartphones etc. Three categories of clients can be distinguished:

- *Mobile*: PDA's or smartphones;
- *Thin*: clients that have no internal hard-drives and all the work is done by the cloud servers. They are used only to display the information;
- *Thick*: regular PC's

Thin clients become more and more popular, because of several advantages like lower hardware costs, lower IT costs, security, less power consumption, less noise.

3.5.1.2 Service Providers

The service providers are the datacenters where applications are hosted. They are the clients of cloud providers and further provide services to cloud clients. An important aspect is virtualization that consists in running multiple instances of virtual servers on the same physical server.

3.5.1.3 Cloud Providers

In cloud computing, not all the servers have to be located in the same place. Even if these are placed far away from each user, to the cloud users this would have absolutely no impact. This offers the service provider a lot more flexibility in designing and securing the system.

3.5.2 Ups and downs

The idea behind cloud computing is that other companies host your application. This offers many advantages for the developers of applications. First of all, scalability is a main concern when launching an application on the market. Instead of buying, installing and configuring new equipment, you can just buy more storage or CPU capacity from the hosting company. Another advantage is that your software will have less capital expenditures since there is no need to buy the necessary equipment. Also the cost for maintainance is reduced since everything is handled by the hosting company. Simplicity of this concept is also an advantage, because it offers the possibility to get the application running in no-time. On the down side, there is the issue regarding senzitive information. In some cases, vendors can not agree to introduce a third party into their business plan. Privacy can be achieved by encrypting the data before sending it to the third party.

3.6 Machine learning

Data classification is of large importance in many applications and data mining techniques are used to overcome it. The purpose of a classification algorithm is to build a classification model given some examples of the classes we are trying to model. This model can be used to classify new examples or just understand better the given data. It is being used in a wide variaty of areas like science, industry and finance. Using a set of data (called training data) machine learning algorithms can predict an outcome for new data, quantitive or categorical, based on a set of features [Hastie et al., 2009].

The outcome of a prediction can be quantitive or categorical. Regression is the statistical approach for getting the quantitive predictions. Examples: predicting the prices of houses, speed of cars, number of errors. Categorical classification gives discrete results e.g. sick/not sick, will rain/will not rain, predict if a vegetable is a carrot/tomato/onion/none.

There are two main types of machine learning techniques: supervised and unsupervised. In supervized learning, the training data is labeled, meaning that we have outcomes for this set of data. Let's use the example of predicting height of the people based on their weight. In this example, the height is the outcome variable and the weight is the feature. We have a training set in which each entry reprezents a person with a known weight. A supervized learning algorithm also needs the labels i.e. the heights for each entry of the training set in order to "learn". Unsupervised learning does not need labels in the training data set for creating preadictions.

3.6.1 Linear Models for Regression

The linear model has been one of the most important models in statistics in the past 30 years [Hastie et al., 2009]. This model assumes that the function F(Y/X) is linear in the

inputs $X_1, ...X_p$. Having this input vector of features X we can predict the output Y using the hypothisis $h(\theta)$:

$$Y = h_\theta(X) = \theta_0 + \sum_{j=1}^{n} X_j \theta_j \tag{3.1}$$

where θ_0 is known as the bias in machine learning.

The most popular method for fitting the training set to a linear model is the least squares method. The idea behind is to find the vector θ so that $h(\theta)$ is close to Y for our training set example (X, Y). This means minimizing the squared error over our training set:

$$RSS(\theta) = \sum_{i=1}^{N} (h_\theta x_i^T - y_\hat{i})^2 \tag{3.2}$$

$RSS(\theta)$ is a quadratic function of the parameters. Its minimum alwas exists, but may not be unique.

Though linear models are quite old, there are still a few good reasons for why they are still used. They are simple and and it is easy to see how the inputs effect the output. They sometimes perform better then newer non-linear models, especially where only small training data sets are available.

3.6.2 Logistic Regression

Logistic regression is one of the most used classification algorithms. It is used to put data into classes. For example, it can be used to decide weather a mail is spam (y = 1) or not (y = 0). The output of such a classifier is a probability value ($0 \leq h_\theta(x) \leq 1$) and by using a threshold new data can be inserted into one of the classes.

Logistic regression can use the sigmoid function for the hypothesis:

$$h_\theta(x) = g(\theta^T x) \tag{3.3}$$

where

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.4}$$

$h_\theta(x)$ is the estimated probability that $y = 1$ on input x. For example, $h_\theta(x) = 80\%$ means that there is 80% chance that the mail is a spam.

3.6.3 Tree Based Methods

Tree-based methods split the feature space into sets and then fit a simple model to each set. A popular method is Classification And Regression Tree (CART). Let's consider a regression example with continuous response Y and inputs X_1 and X_2 that take values in the unit interval. Fig. 3.7 shows an example representation of a binary tree. The full data set is located at the top of the tree. Observations satisfying the condition at each juntion are assigned to the left branch and the others to the right branch.The leaves of the tree represent the regions in which the dataset is split. The regression model predicts Y with a constant c_m in region R_m:

$$f(x) = \sum_{m=1}^{5} c_m I\{(X_1, X_2 \in R_m)\} \tag{3.5}$$

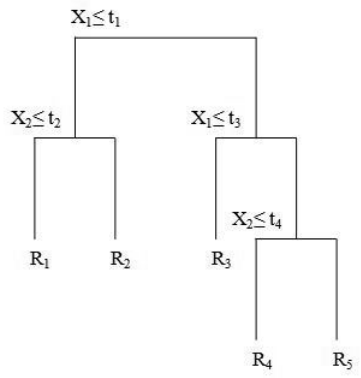

Figure 3.7: Example binary tree with five regions in which the dataset is split

A key advantage of the recursive binary tree is its interoperability [Hastie et al., 2009]. The feature space can be fully described by a single tree and it can handle both numerical and categorical data. However, with more than two inputs partitions can be difficult to draw and because the decision is made at each node, such algorithms do not guarantee to return an optimal decision globally. Also, complex trees do not generalise well from the training data, problem known as overfitting.

4 Concept

The multitude of sensors present in vehicles today can be used to collect information in such a way that can be shared between drivers. This thesis proposes a concept for parking spaces data acquisition from ultrasonic and camera sensors mounted on vehicles and distribution of this data to the cloud, where it can be accessed by other drivers as well as a concept for remote monitoring of the vehicle while it is parked. The system is able to construct a parking spaces map by using only these sensor information in combination with GPS position information and map services data. Remote monitoring is done using video images taken from cameras directly integrated into vehicles and sent to the vehicle owner's smartphone.

4.1 Basic Architecture

In order to collect sensor data from vehicles we envision an architecture like the one in Fig. 4.1. It consists of vehicles, which are the sources of information, cloud server, which is where the data is stored and processed and clients, which will use the acquired information.

4.1.1 Vehicles

The vehicle are the sources of information for the presented system. For this concept, we consider only vehicles equipped with ultrasonic and camera sensors. These sensors are mounted on the left and right sides of the vehicle as well as incorporated in the front and back vehicle's bumpers.

This vehicles have separate ECU's for processing ultrasonic and camera sensor information. The parking processing ECU is running algorithms that use ultrasonic sensors information for detecting parked vehicles, their lateral boundaries and corner positions and can measure the parking space in depth and also possible obstructions on the path of the vehicle. While the vehicle is passing near a free parking space, the ECU generates a message containing length

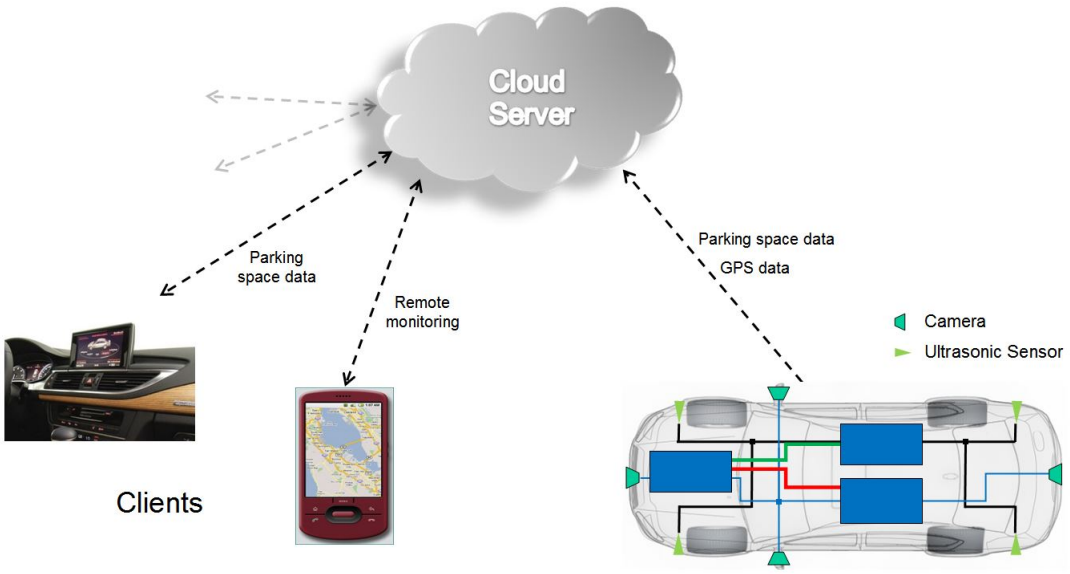

Figure 4.1: Basic Architecture consists of vehicles that provide data, cloud server that collects and processes data and the clients who use the data

and depth information, type of parking (cross or parallel), reference type (curbstone, wall), distance to corner of the parking space and many other parameters. We add GPS information to this data and send it to the cloud server in order to build a database of available parking spaces by the means of a mobile Internet connection. This ECU also controls the automatic parking function. It also sends a message when the automatic parking process was completed successfully and also when the pull out manouver was finished. This information can also be sent to the cloud server. The camera ECU uses multiple cameras mounted on the vehicle to build a virtual top image of the vehicle. We want to use this images for remote monitoring of the vehicle while it is parked by sending video information towards the owner of the vehicle using the smartphone, in case the alarm starts or even on demand. The camera system can be also used in the parking space detection by running line detection or other algorithms and can add even higher value to the concept.

4.1.2 Cloud Server

The cloud server has the task of collecting data from the vehicles. The data has to be processed and labeled so that only valid, accurate and consistent data is sent to clients. These clients make queries to get access to information according to certain parameters like their location and prefered options. The sever can be a dedicated server configured exactly for this application or can be hosted by a service provider like Amazon S3, Google App Engine or many others.

4.1.3 Clients

The clients are other drivers that are searching for available parking spaces and vehicles owners that want to make sure that their vehicles are safe while parked. They make queries and the information has to be given to them in a proper way. A good user interface is important in order for such a system to be launched on the market. One possibility is to make the parking space data available through the human machine interaction (HMI) displays of the vehicles. Many services that use cloud support are already available into production vehicles (see Chapter 1) by using integrated 3G modems that offer the possibility of inserting a SIM card with mobile Internet connection. The HMI offers services like navigation, media streaming, email and traffic information. We must think of how to integrate parking space availability information into existing displays and into current services prezent in vehicles. Another possibility is to make data available through smartphone applications. Users could use it at home in their houses or in their cars by using docks to support their smartphones.

For designing a good interface we must consider the user needs and behaviour. We have to analyze what kind of information must be displayed to clients and in what way.

4.2 Use Cases

Let's consider our "hero", John is driving in the morning to the city center to solve some issues with his credit card at his bank. He wants to be there early thinking it won't be so crowded and won't have to wait at a long line. But when he arrives there with his car, he notices that there aren't any availalbe parking spaces in the area. So he starts the application on his car that can help him find a free parking spot. Luckily, Lucy is driving to work just 100 m in front of John and her car detects a parking space on the side of the road. John is informed through his car display that a parking spot is free in front of him. He parks his car there and he can now go solve his credit card problems. Since he parked there, on the smartphone application, the parking spot is not available anymore. After an hour, he solved his issues at the bank and he drives out of the parking space. Now the parking space is made available for other drivers that search parking spaces in that area.

Now John wants to drive to work. Since he does not have a fixed parking location near his work place, he wants to know in advance where free parking spaces are located near that location. He opens the smartphone application that can help him view the parking situation in that area. He sees that there are enough free spaces there, so he is not worried that he might not find a parking spot.

Driving to work, John hears at the radio that in the last week there have been reported many car thefts in the city. Now he is frightened that someone might want to steal his new car. But he remembers that his car has the option for remote parking monitoring. If anything happens with his car, he will see it through his smartphone and he can than intervene. He is again releaved of his worries.

4.3 Challenges

Though the idea of collecting parking space data and sharing it with other drivers is simple, there are some challanges that we need to overcome.

First of all, we need to considered the validity of the acquired information. Though the software running on the vehicles is able to detect free parking spaces, these may not necessarily be valid. The algorithms use ultrasonic sensor data to measure distance from one corner of a parked vehicle (or object) to another parked vehicle (or object). These measured spaces could be anything from garaje entrances to illegal stoping zones. That is why it is necessary to process the collected data on the server side in such a way that only valid data is sent to clients. For this purpose there has to be a filtering algorithm that can tag data accordingly into valid and invalid.

Another challange is the freshness of data. A parking space can be free now, but the next minute a car can occupy it. So we must use timestamps to keep track of when the data was acquired.

GPS accuracy can also be a problem. Putting an accurate location tag on the detected parking space it is very important. GPS devices provide coordinate readings accurate to a few meters, but this is not always the case. Accuracy depends on the number of sattelites that are available. That is influenced by the clearness of the sky and the surroundings near the location of interest. Improvement can be achieved by combining GPS data with map-based services.

An aspect that we must not neglect is the anonimity of the collected data. Users would most likely not accept to use this system if the collected data would contain their personal information, like where they drove and parked their vehicles. Also back-tracking must not be possible i.e. users must not be able to access journey or personal information of other users.

5 User Interface Investigation

In this chapter we investigate how the parking availability information can be presented to the end-users. Paper prototyping and focus group methods have been used with the purpose of finding a proper user interface for our concept.

5.1 Paper Prototypes

Paper prototyping is a method used in the design process of a software product that helps developers to better understand what are the user's needs and expectations of that software. To make other people understand our concept and to generate new ideas, three paper prototypes were prepared. This are basically map overlays that contain parking space availability information.

The first prototype shown in Fig. 5.1 (top) shows parking spots as round coloured dots. Each dot reprezents a parking space and its color shows its availability. If the user presses on a dot, an information bubble appears that displays additional information regarding that parking spot. Next one (middle) presents parking areas and not single parking spots. A green rectangle means that it is very probable that there is a parking spot into that area. Unavailable parking area are represented in red. And the last prototype (bottom) represents parking spots as green rectangles, which means that it is very probable that there is a parking spot into that area.

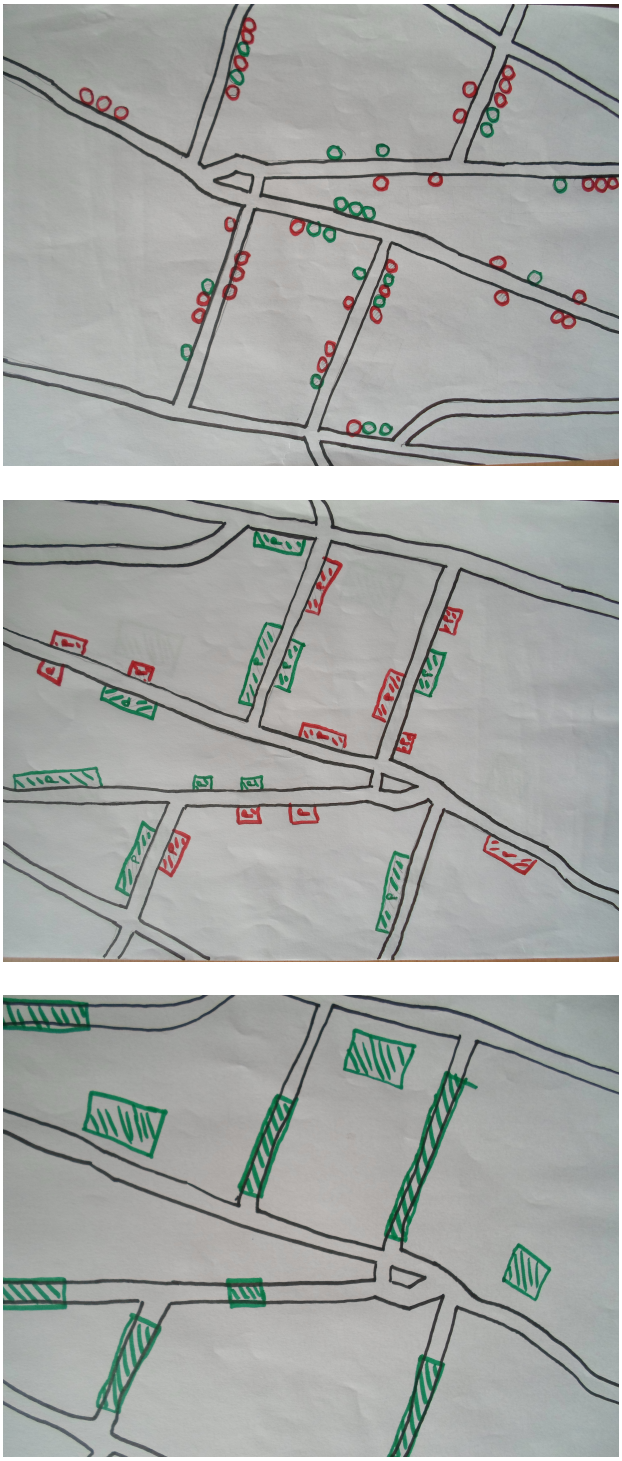

Figure 5.1: User interface prototypes - the top figure shows parking spaces as round colored dots according to their availability; in the middle figure parking spaces are represented as red and green rectangle areas; in the bottom figure only available parking spaces are represented green rectangle areas

5.2 Focus Group

A focus group was held in order to get new ideas for our system while also getting feedback for the concepts we already had. The discussion was conducted in order to get the following ideas from the involved people:

- To understand what problems people are facing when looking for a parking space
- To find out if they already use or know about such a system.
- To get ideas on how they would envision such a system.
- To see what information would they find interesting to have.
- To know what they think of the concept.
- To get feedback and new ideas for the paper prototypes.
- To know in what situations would people use the system.

5.2.1 Participants

The participants of the focus group are all with technical background either from software or hardware domains: 2 managers, 1 engineer, 2 students and 2 discussion conductors.

5.2.2 Discussion

All participants agreed that it is a common problem not knowing where to park your vehicle in crowded cities and they would be interested in having some kind of help regarding this matter. One problemis that in large cities, there are different costs for parking from street to street and you can not know the prices only if you know the area or if you have been there before. Also, the tickets are not available from one area to the other. It would be good to have this kind information available in advance.

A tool that provides information related to parking would be useful to everyone. The participants would want to know where parking spaces are available, the costs and other details about the parking zones. For example, it would be interesting to know in what type of area is the parking space located to make sure it is safe to park there (dark street, underground) or if it is directly under the sun etc. Some people also prefer certain parking types (parallel over cross parking). Another case where such a parking space application would be useful is when planning a car trip to a new location. It would certainly be helpful to know in advance where parking spots are located. Map based applications have been used for this purpose so far.

An interesting idea generated in the gocus group discussion is that the user interface of the available parking search system can be just an extra add-on to the navigation software of the vehicle. While driving, the users do not care of having too much information and just want to be guided to the nearest parking space. They would prefer having as few text as possible

and information displayed by symbols. The system could remember personal preferences for parking spots and make automatic recommendations while driving. Accuracy of the provided information is a very important aspect. Users would not want to be guided to a parking space and when they get there to see that it is already occupied or it does not match their interest.

Another inportant aspect is related to privacy. It is important to collect anonymous information from the vehicles. Users are generaly not against the collection of information from their vehicles as long as privacy is maintained.

5.3 Results

One of the most important aspects is that people would prefer having a single system that integrates all the functions: parking, navigation, road information etc. and they probably would not use different platforms for each of these. Another resulting conclusion that we can draw is that we need to differentiate between two use cases:

- When people plan their drive in advance, from their home maybe and they want to find suitable parking spaces near their destination
- While driving, when people want to find the nearest parking space.

Based on the results of the focus group, a new user interface prototype was designed (Fig. 5.2).

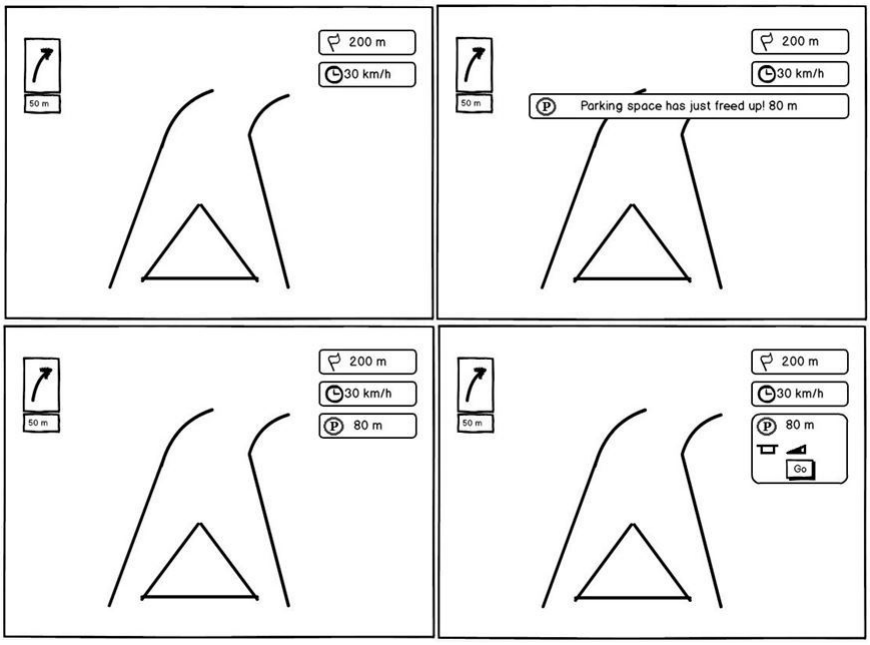

Figure 5.2: Pre study resulting prototype - parking space availability information integrated into the navigation software of the vehicle

The parking space availability information is integrated into the navigation software of the vehicle. The default use case is like this: the driver starts the navigation software and selects its destination. When he is close to its destination, maybe a few hundred meters, he will receive some popup messages on the display to inform him that some parking spaces where detected by other vehicles in front of him. He can then press on such a parking space and further information will be displayed, like the type of parking space, length and other caracteristics, offering also the possibility to reroute its destination towards that selected parking space.

6 Implementation

This chapter describes a first prototypical implementation of the concept. The prototype is basically a complete system that is able to collect parking space information, send it to a server database where it is also processed and classified and clients can make queries for accessing this data. Also real-time video communication has been implemented for remote monitoring of parked vehicles. For this, a vehicle equipped with a semi-automatic parking assistance system and multi-camera system as the data source, Google App Engine platform as the cloud server and an Android based client were used.

6.1 Distributed Architectures

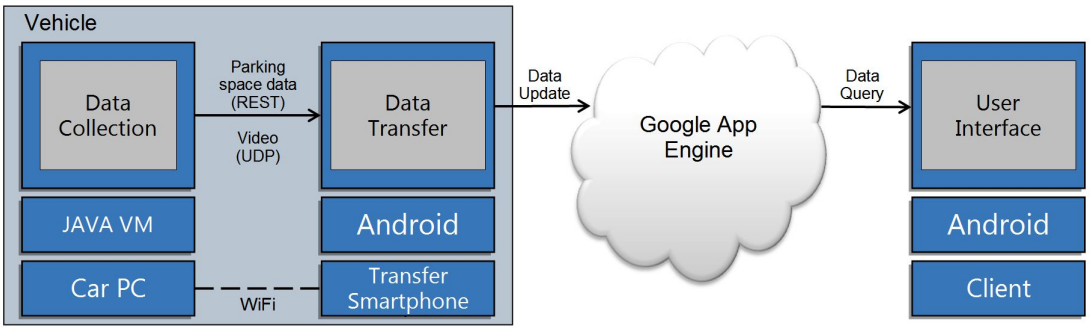

Figure 6.1: Distributed architecture of the prototype includes the vehicle equipped with the car PC that acquires data from the CAN bus and smartphone used for its GPS and fortrasferring data to the server, App Engine server that stores and processes the data and the Android client that displays parking space information

The prototype architecture consists of several parts. The car PC grabs the data from the CAN bus and sends it to an intermediate smartphone via a WIFI connection. The smartphone is used as a gateway to the server for its Internet connection and also for the integrated GPS. The App Engine server collects and processes the data and accepts queries from Android clients.

6.2 Vehicle

The implementation relies on a vehicle that is specially equipped by Bosch. The vehicle is a Mercedes B class and the main components that are important for the implementation are:

- *Automatic parking assist aystem*: using ultrasonic sensors the system is able to detect parking spaces while the vehicle is moving. When the system finds a suitable parking space, this is reported to the driver via a display mounted in the vehicle. The driver then has the option to let the system park the vehicle on its own into that parking space. Also, the option to park out from the same parking space is available when the driver wants to leave. The software that this system uses is running on an car integrated PC.
- *Multi-camera system*: the system consists of four externally mounted (front, back and sides) VGA cameras together with an ECU that is able to construct a virtual top view image of the vehicle.

6.2.1 Car PC

The car PC is used in the implementation to retrieve the parking space data from the CAN bus and the video signal from the camera ECU. It is a Windows 7-based PC, specifically designed for vehicle integration.

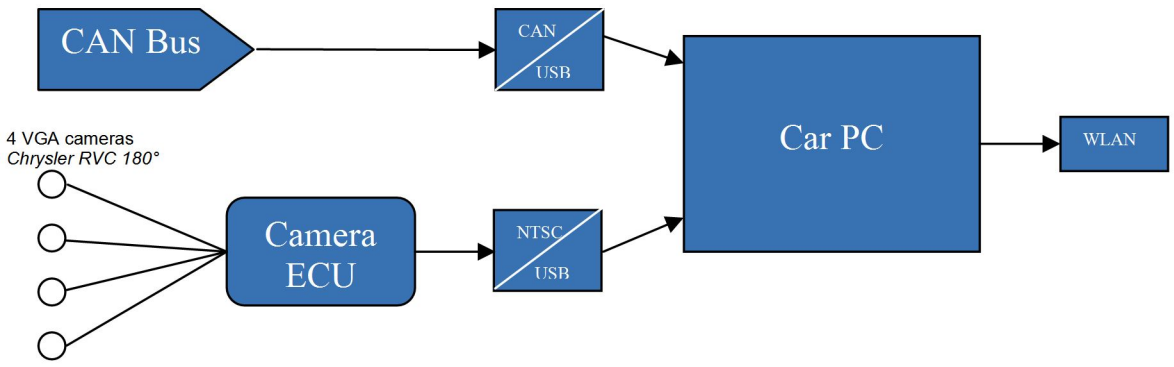

Figure 6.2: Car PC connected to vehicle CAN bus through the CAN2USB interface and to the camera ECU through the NTSC2USB interface

A CAN to USB interface[1] is used for fetching the messages from the bus. The PC runs a HTTP server that can distribute this messages to incoming connectrions through the WIFI receiver module. The NTSC signal coming from the multi-camera ECU is sent to the PC through a NTSC to USB converter. We used an Elgato Video Capture device[2], capable of digitizing the NTSC signal in real-time.

6.2.1.1 CAN Messages Acquisiton

For this prototype we chose to use an intermediate smartphone for communicating with the server and also for its GPS capabilities, so we used an HTTP RESTful server implementation that is able to read the CAN messages acquired through the CAN to USB converter. The server cyclically reads the signal values from the CAN bus and sends the values to the HTTP clients when they make requests. The reason for choosing this implementation is that, since the project is in collaboration with Bosch, this part of the project was available from a previous one.

There are three type of messages that are acquired from the CAN bus for the implementation:

- *Detected parking space messages*: This messages provide information about parking space that are detected by the ultrasonic sensors of the automatic parking assist system. Length, width, type and other parameters are included.
- *Parked-in messages*: When using the park-in option of the automatic parking assist, a CAN signal is issued after the maneuver has successfully completed.
- *Parked-out messages*: When using pull-out control of the automatic parking assist, a CAN signal is issued when this maneuver is completed.

6.2.1.2 Video Transmission

The cloud server receives the requests for starting the transmission, either from the vehicle or from the smartphone, and sends back the necessary information for the connection to be established, like illustrated in Fig. 6.3.

The video that is transmitted is a top view image of the vehicle constructed by the multicamera system ECU of the vehicle. The image is virtually constructed using four VGA cameras mounted on the vehicle. A sample of the image can be seen in fig. 6.4.

Several solutions were tested to find what is the best choise for transmitting live video between the vehicle's car PC and an Android client. The results are resumed in table 6.1.

The video acquisition server runs on the car PC. This grabs the video from the USB device and sends it remotely thorugh WIFI or mobile connection to an Android client. Using the

[1] http://vector.com/vi_cancase_xl_log_en.html
[2] http://www.elgato.com/video/video-capture

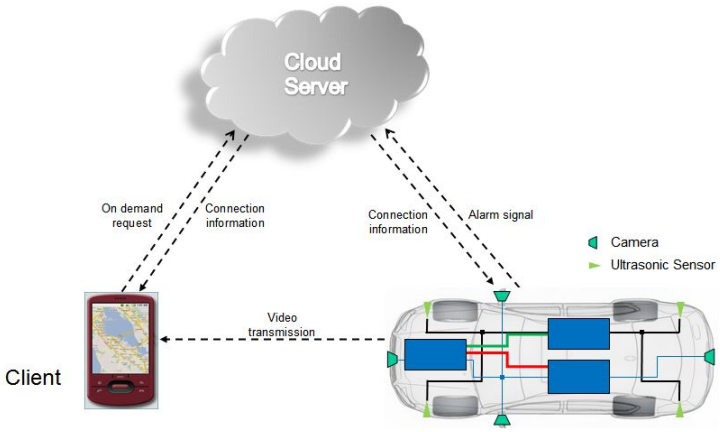

Figure 6.3: Remote monitoring principle - vehicle sends request to the server in case the alarm is triggered, the server establishes the connection between smartphone client and the vehicle so that live video is transmitted

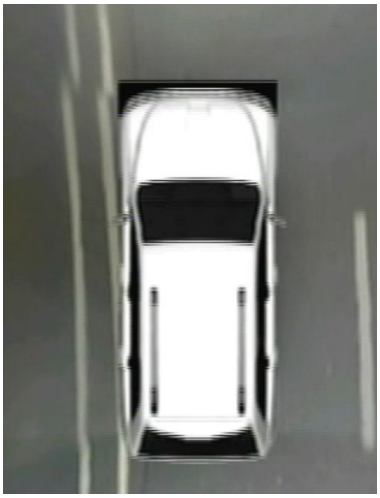

Figure 6.4: The top view image is a virtually constructed image using the four VGA cameras mounted on the vehicle

JavaCV library[3] it is very easy to process the video from any USB device. Frames can be captured with defined rates and encoded to almost any video format. We chose mpeg format because it proved to work the best in real-time with regard to latency. The encoded video can then easily accessed through a UDP connection on the defined socket. The simplified Java

[3]https://code.google.com/p/javacv/

	VLC	Ffmpeg + Wowza	Flash Encoder + Media Server	JavaCV
Capabilities	-Easy to setup stream from camera device -encoding to many video formats -many options available -open source	-many available configurable options -encoding to many video formats -ffmpeg is open source	-friendly interface -many options and formats	-Java library -also for Android -wrapper for Ffmpeg and openCV -possible to capture frame by frame -send as JPEG or encode as video
Disadvantages	-must capture the stream as RTSP in Android	-ffmpeg doesn't provide crossbar support; -Wowza is not open source	-uncertainty about Android HTTP Live Streaming support -or the need to use a Browser environment for Flash support	-documentation not available
Test conclusions	-high latencies even for low quality formats -not working well for Android -hard to find a suitable format for Android -distorted images	-workaround for crossbar problem by using AmeRec software -high latencies -hard to find a suitable format for Android	-low latencies for .flv format (tested on local PC only) -HTTP Live Stream obsolete as of Android 4.2	-low latencies for JPEG UDP stream -requires high bandwidth (600 KB/s for 600x500@15fps) -MPEG encoded stream possible with low latencies but only for lower quality (300x300@10fps) -high latencies for h264 encoded stream

Table 6.1: Comparison of different real-time video transmission software and libraries

code is shown in listing 6.1.

```
// start the frame grabber
grabber.start();
IplImage img;

// create frame recorder
FFmpegFrameRecorder stream = new FFmpegFrameRecorder("udp://46.115.82.158:6000", 150,
    150);

 stream.setFormat("mpegts");
// set other encoding options
// ...

stream.start();

while (running) {
        // grab frame
        img = grabber.grab();
        // record video
       stream.record(img);
}
// delay for modifying frame rate
Thread.sleep(80);
```

Listing 6.1: Simplified Java code of the video capture based on the JavaCV library

6.2.2 Transfer Smartphone

An intermediate Android smartphone is used because the GPS signal is not accessible on the bus of the test vehicle. So for the prototype we are using the GPS of the smartphone and also its mobile Internet connection for transferring the data to the cloud. Another alternative would have been to use a separate GPS device and a mobile Internet USB stick connected to the car PC. The smartphone performs the operations shown in fig. 6.5.

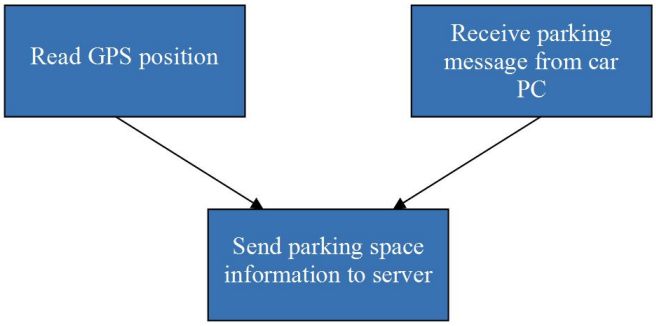

Figure 6.5: The transfer smartphone listens for parking events and when such an event occurs the GPS location is added and the data is sent to the server

A HTTP client cyclically sends requests to the HTTP server that runs on the car PC for reading the CAN message values that are of interest. When a parking space is detected, or a parked-in or parked-out message is issued, a separate async task is created that sends the information to the server. A separate thread constantly reads the GPS position of the smartphone device and when such a parking event occurs, the position information is added. The communication with the server is performed using an API created using the server part of the code. This is described later in section 6.3.

6.3 Server

The server stores the parking space data sent by the vehicles that are equipped like described in the previous sections. This parking space data is sent by through the smartphone's Internet connection and stored into a database. The server also performs some processing on the collected data. First of all, the collected parking space locations are map matched with the help of open-source map-based services. Secondly, a filtering algorithm is applied for distinguishing the valid entries from invalid ones.

The server implementation relies on Google App Engine. Google App Engine[4] allows developers to build their own applications on the same infrastructure that is being used by Google's own applications. It supports applications written in several programming languages. Java Runtime Environment allows code written for JVM, Java servlets, Java programming language, JavaScript or Ruby. App engine also provides a Python runtime environment which includes a Python interpreter and the Python standard library and also features a PHP runtime for Google Cloud SQL and Google Cloud Storage support. App Engine provides a straightforward way of creating an Andoid connected project using Eclipse IDE[5].

Java Persistance API (JPA) is an interface available in App Engine used for storing objects that contain data into relational databases. Using JPA it is easy to create object entities that will contain the data to be stored. An entity is a lightweight persistence domain object. Typically an entity represents a table in a relational database, and each entity instance corresponds to a row in that table.[6]. These objects are also available in the client code in order to keep the data to a consistent format.

After the entities are defined, App Engine provides a straight forward way of creating the endpoint code coresponding to each entity, used to process insert, update, retreive and delete requests. Google Cloud Endpoints allow to generate APIs and client libraries from an App Engine application, i.e. the API backend in order to simplify client access to data from other applications[7].

6.3.1 Entities

Each entity propriety is a column in the table that the entity represents. For each propriety, *get* and *set* methods need to be defined.

6.3.1.1 ParkingSpace entity

For storing parking space information we created the ParkingSpace entity. This contains the proprieties of the detected parking spaces that we need for the processing and filtering of the data. The entity is shown in listing 6.2.

The parking space location is stored using *latitude* and *longitude* coordinates and also *orientation*. *lastLength* and *lastWidth* store the length and width of that parking space. When detecting a parking space, a certain length is measured, according to the objects that delimit this space. If the same parking space is detected some other time, the length of this parking space can be different. For example, the vehicle in front has left and now the area is larger.

[4]https://developers.google.com/appengine/?hl=ro
[5]http://www.eclipse.org/
[6]http://docs.oracle.com/javaee/5/tutorial/doc/bnbqa.html
[7]https://developers.google.com/appengine/docs/java/endpoints/

```
@Entity
public class ParkingSpace {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "parkingSpaceId")
	private Key parkingSpaceId;
	private int accumulatedLength;
	private boolean availability;
	private long lastDetected;
	private long lastParked;
	private int lastLength;
	private int lastWidth;
	private String lastUser;
	@Latitude
	private double latitude;
	@Longitude
	private double longitude;
	@Geocells
	private List<String> geoCell;
	private String locationType;
	private OSMStreet osmStreet;
	private int orientation;
	private String parkingType;
	private String side;
	private int timesDetected;
	private int timesParked;
	private int usersParkedNumber;
	private int usersDetectedNumber;
	private boolean validity;
	private long validationTimestamp;
	@ManyToMany(fetch = FetchType.EAGER)
	@Unowned
	private List<User> usersThatDetected;
	@ManyToMany(fetch = FetchType.EAGER)
	@Unowned
	 private List<User> usersThatParked;

	//getters
	 ...
	//setters
	...
```

Listing 6.2: ParkingSpace entity is used for storing parking space information

We keep track of the total length of the parking space using *accumulatedLength*, and so the parking spaces will grow in size. To keep track if a parking space is free or occupied we use the *availability* variable. *lastDetected* and *lastParked* are the timestamps for when the parking space was last detected, respectively when a vehicle last parked in it. We also keep track of the users that detected each parking location. This users are not the end clients, but the providers of information i.e. vehicles equipped with ultrasonic sensors. *lastUser* stores the id of the user that provided the information, *usersThatDetected* and *usersThatParked* store every user id that detected and parked into a certain parking space. *validity* will be set to true if the filtering algorithm determines that a certain stored parking space is valid. The filtering

algorithm uses some counters for determining the validity: *timesDetected* is incremented each time a parking space is detected near the same location, *timesParked* is incremented when someone parks in the same parking space, *usersParkedNumber* keeps track of the total users that parked in that parking space and *usersDetectedNumber* stores the total number of users that detected the respective parking space. *OSMStreet* is a separate entity that stores street information from OpenStreetMap. Using this we can asociate a parking location with the street on which is located.

geoCell is used to overcome one limitation of the App Engine database queries: "Inequality filters are limited to at most one property"[8]. For bounding box queries this becomes a big problem. Bounding boxes queries are the ones where the latitude and longitude values both have to be between a minimum and a maximum value. To overcome this we used the *geomodel* library[9]. This provides a generalized solution for performing basic indexing and querying of geospatial data in Google App Engine.

6.3.1.2 ValidParkingSpace entity

After a parking space is validated by the filtering algorithm, a new entity is asociated with it. We use this so that the time for processing a client query is reduced, by only searching for these valid parking spaces into the database.

```
public class ValidParkingSpace {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	@Latitude
	private double latitude;
	@Longitude
	private double longitude;
	@Geocells
	private List<String> geoCell;
	private ParkingSpace parkingSpace;

	//getters
	...
	//setters
	...
```

Listing 6.3: ValidParkingSpace entity - is created when the filtering algorithm validates a ParkingSpace entry and associates with it

[8] https://developers.google.com/appengine/docs/java/datastore/queries
[9] https://code.google.com/p/geomodel/

6.3.1.3 User entity

The user entity stores the id of the user. Each vehicle that provides information has to have its own id, so that we can count the number of users that detects a certain parking space.

6.3.1.4 OSMStreet entity

In order to map each detected parking space to a certain street, OpenStreetMap (OSM) information is used. OSM data is available in many different formats, among which is .xml. The architecture of the OSM model consists of *nodes*, *ways* and *relations*[10].

Nodes are the core elements of this model. They represent single points in space defined by latitude, longitude and node id. They can define standalone point features by using *tags* (e.g. a fuel station can be tagged with amenity=fuel) or can be used to define the shape of a way. A way is an ordered list of nodes which should also contain at least one tag or it is part of a relation. A relation consists of one or more tags and an ordered list of nodes and/or ways. These are used to model logical or geographic relationships between objects.

Streets are defined by OSM as ways. For this, we use in the OSMStreet entity two lists: *centerLatitude* and *centerLongitude*. These are order lists that represent the latitude and longitude of each node in the way. To simplify the query of a street according to its GPS position we also use *centerLatitude* and *centerLongitude*. These represent the position of the middle node in the way (e.g. if the way consists of five nodes, the center node would be node 3). *infoName* represents the name of the street and *type*, the type of the street (e.g. highway, residential street).

OSM street data was extracted from large "planet" .xml files which provide the entire world map information. Smaller files exist for each country or even smaller areas. This files are available for download from many servers (e.g. http://download.geofabrik.de/) and most of the files are updated regulary. Specific data (like the streets in our case) can be extracted from these files using a small tool called *Osmosis*[11]. After only the street information was extracted, the data was inserted into the App Engine database using the bulkloader tool[12]. This can insert large amounts of data using .csv files as input.

6.3.2 Processing Detected Parking Spaces

Parking space data is received from vehicles that make insert requests to the App Engine server in the form of ParkingSpace entities. The basic structure of an insert request is described in fig. 6.6. First step after a new ParkingSpace object is received is to map its location coordinates to a street from OSM. New latitude, longitude and orientation values are calculated, after which it is checked if that parking space is already in the database. This is done by checking if

[10] http://wiki.openstreetmap.org/wiki/OSM_XML
[11] http://wiki.openstreetmap.org/wiki/Osmosis
[12] https://developers.google.com/appengine/docs/python/tools/uploadingdata

the location coordinates of the respective parking space are approximately the same with one of a parking space from ParkingSpace table. If this is the case, then they are merged together and a check is done to see if the old entry was already validated. If the entry was validated, this new entry will also be considered as valid and the database will be updated. In the case that the parking space is not validated, the validation algorithm is applied. For every new validated ParkingSpace object, a ValidParkingSpace object is created so that the data of the validated parking spaces can be accessed easier.

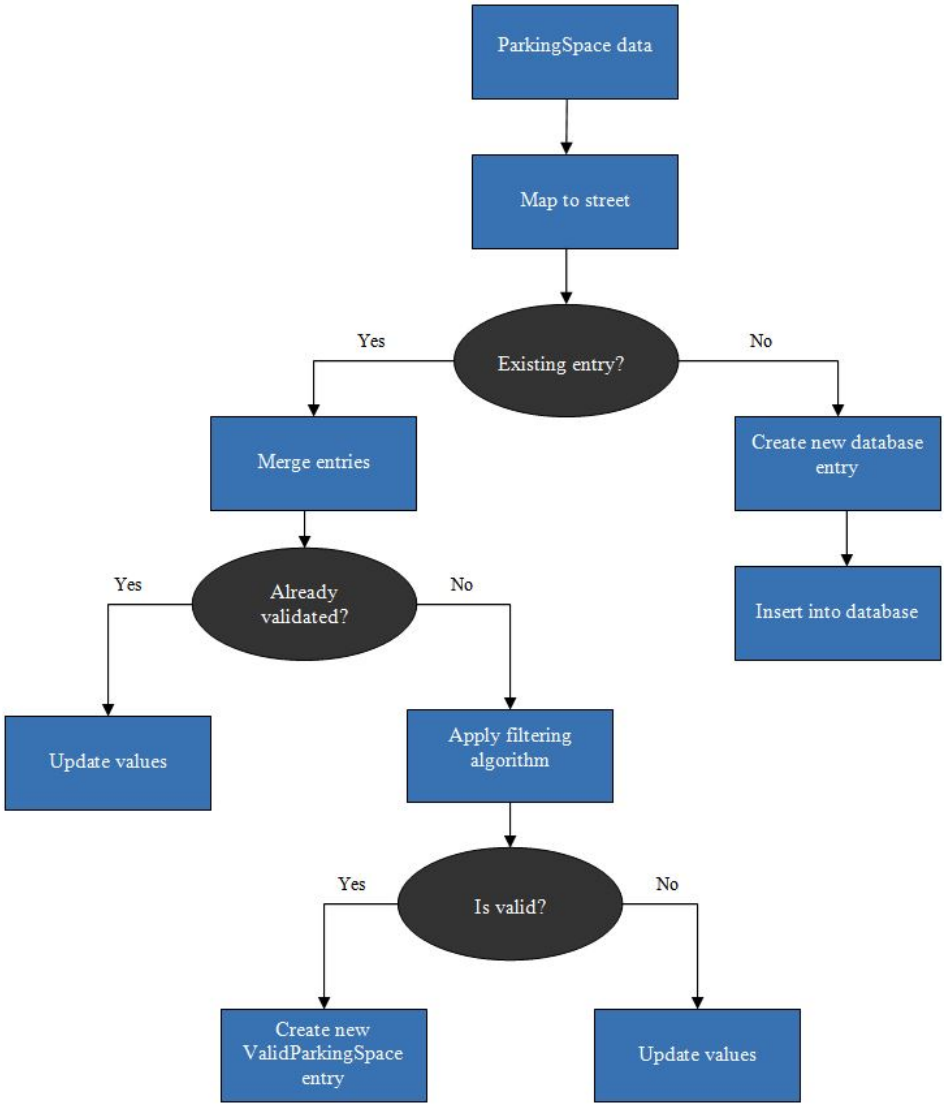

Figure 6.6: The parking space data is received by the server and certain processing steps are taken according to the data that is contained

6.3.2.1 Street Mapping

Street mapping consists of asociating the position of a detected parking space with a street from OSM database. This is helpful to correct a GPS reading that is not so accurate. First all the streets in a range of 500 meters are retrieved from the database. Since OSM streets are actualy ordered lists of nodes that define the path of the street, we search for the closest distance between the position that we want to map and the line that ties every two nodes, after which we map the position onto that line, according to fig. 6.7.

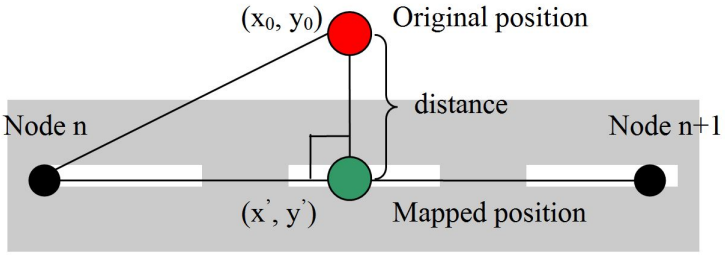

Figure 6.7: The position of the detected parking space is corrected and mapped to the closest street

Since every point is defined in latitude and longitude we need to somehow convert to a planar system and for this we use the Mercator projection[13]. This projects the Earth to a cyilinder and GPS coordinates can be converted to planar values. To convert a latitude and longitude point to a planar point we use the following formulas:

$$x = longitude \tag{6.1}$$

$$y = \ln\left(\tan\left(\frac{\pi}{4} + \frac{latitude}{2}\right)\right) \tag{6.2}$$

To calculate the distance between a point and a line we use the following formula:

$$distance = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \tag{6.3}$$

,where $ax + by + c$ defines the line equation, $a = y_1 - y_2$, $b = x_2 - y_1$ and $c = (x_1 - x_2) * y_1 + (y_2 - y_1) * x_1$, x_1, x_2, y_1 and y_2 being the point coordinates of the two delimiting nodes and

[13]http://wiki.openstreetmap.org/wiki/Mercator

x_0 and y_0 represent the coordinates of the original point. After we find the street segment which is closest to the point that we test, we must find the coordinates of that point on the street. We use the following formulas for calculating the planar coordinates using the distance to that street segment:

$$x^{'} = x_0 - \cos\left(\arctan\frac{|b|}{|a|}\right) * distance \tag{6.4}$$

$$y^{'} = \frac{-c - a * x^{'}}{b} \tag{6.5}$$

6.3.2.2 Merging Parking Spaces

When inserting a new parking space into the database we check to see if other parking space are close enough to that one. The approximation value that is used is the $accumulatedLength/2$ of the parking space to which we check. If two or more parking spaces are close enough, we merge these into a single ParkingSpace entity. This is shown in fig. 6.8.

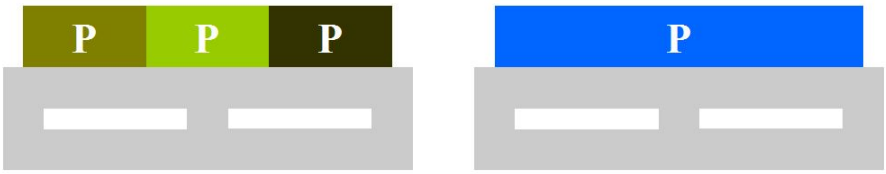

Figure 6.8: Parking spaces that are close enough are merged into a single parking space

For the parking spaces that are merged together, certain values need to be calculated and others taken from one of the spaces that are merged. New *latitude*, *longitude* and *orientation* values are determined, *accumulatedLength* is set to the total length of the merged spaces. If one of the parking spaces is validated, then the new merged space will also be valid.

6.3.3 Filtering Algorithm

A good processing of the collected parking space data is necessary so that only valid information is made available to clients. Determining the model for the classification of data is an important part of the concept. A learning algorithm must be implemented that will be capable of learning to distinguish between valid and invalid detected parking locations. Valid spaces are the ones which are detected by the vehicles and are trully parking spaces, while the invalid spaces are ones which are wrongfully detected as parking spaces i.e. false positives. The quality of the provided data towards clients depends on the algorithm that is implemented. The algorithm

uses parking space data collected from multiple vehicles. The process is illustrated in Fig. 6.9.

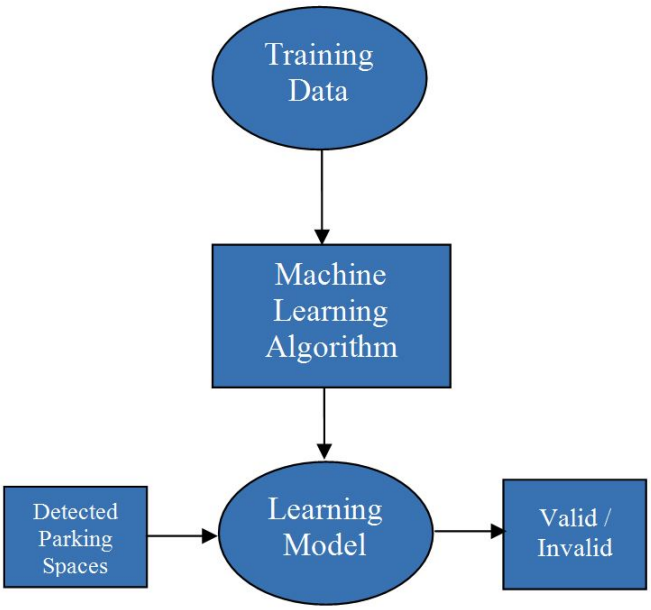

Figure 6.9: The learning algorithm for parking space validation uses the labeled training data for establishing the learning model afterwhich new data can be labeled accordingly

The parking space information consists of length, depth, road side, type, certain distance values, GPS data etc. We will count the number of times each parking space is detected at the same GPS location. But this information is not enough for determining if a detected parking space is valid or not. In this concept, we use also the information provided by the automatic parking systems and collect the messages from the CAN bus regarding the completion of automatic parking in and parking out maneuvers. This is useful because we can also count the number of times vehicles park in and out at the same location. For better determining the correct GPS position of the parking spaces, map services data can be used. There are open source providers that allow their information to be used for purposes like this. Using this, we can easily determine if a detected parking spaces is located on a street, parking lot or other location. In [Mathur et al., 2010], it is used an environment fingerprinting technique for accurate determination of a parking location. In this concept, we can accurately determine the position of the parking spaces using GPS combined with map services information.

First, we need a lot of parking space data that we can use as the training data for the algorithm. Example of features that we can use are:

- *timesDeteted*: Number of times a parking space was detected.

- *timesParked*: Number of times a vehicle parked into a parking location.
- *usersDetectedNumber*: Number of users that detected a certain parking location.
- *usersParkedNumber*: Number of users that parked into a certain parking location.
- *accumulatedLength*: Length of the parking location.
- *width*: Width of the parking location.
- *type*: Cross or parallel parking type.

Once it is decided on the features to use, the learning algorithm can be applied so that parameters are found i.e. the learning model that will help decide when a parking space becomes valid, when new data is collected. After we determine the learning model, new parking data can be classified accordingly. When a parking space is detected, we can determine if it is valid or not according to the model parameters.

6.3.4 Processing Park-in and park-out messages

Parked-in messages are counted for each ParkingSpace entry and used as a feature in the filtering algorithm. The number of times a vehicle parks in at a certain location has a great impact on the validity of that parking location. Parked-in and parked-out messages that come from the automatic parking assist system are also used in our system for changing the availability of a parking space in our database. This is the case only for the validated parking spaces. When a park-in message is received after a successful automatic parking assist maneuver, an update request is sent to the server with the GPS location of where the message was created. If a validated parking space is found at that location, its availability is changed. Park-in messages set the availability to false and park-out messages to true.

6.4 Client

The clients are the ones that finally use the information that is provided by our system. They can search for available parking spaces and monitor their parked vehicle wih the help of live video transmission.

The client prototype is implemented on an Android smartphone. We used a Samsung Galaxy S4 for development. The implemented Android application is able to display parking information by making queries to the App Engine server and also display live video directly from the multi-camera system installed on the test vehicle.

6.4.1 Parking Space Information Display

For displaying parking space information Osmdroid[14] library is used. This is a substitute for Google's MapView that uses OpenStreetMap data. Its main advantage is that it offers mainly the same possibilities with an open source license.

6.4.1.1 Parking Spaces Overlay

Using Osmdroid it is possible to add overlays on the standard map tiles according to GPS coordinates. Available parking spaces are displayed as green rectangles. They are scaled according to their length and a standard width, according to the zoom level of the map.

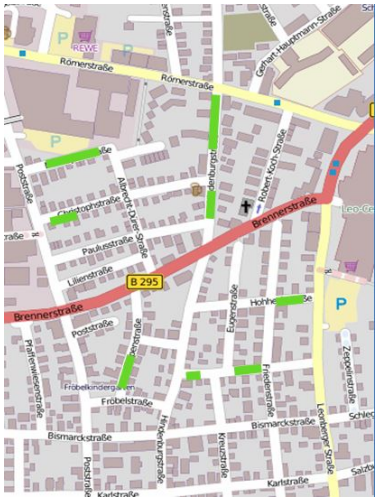

Figure 6.10: User can view available parking spaces as green rectangles on the map (©OpenStreetMap contributors)

6.4.1.2 Bounding Box Queries

The parking space data is queried from the App Engine database according to bounding GPS coordinates of the screen. The minimum and maximum latitudes together with tthe minimum and maximum longitudes determine the box. With every scroll and zoom movement on the map, this box is recalculated and a *QueryTask* is created for retriving the parking spaces from the database. This query task uses the created backend API to retreive the ValidParkingSpace objects that are in the bounding box. The query request is built using the API and the results are retreived through HTTP. The latitude and longitude values that define the box are passed as parameters to *queryValidParkingSpace* server method.

[14]https://code.google.com/p/osmdroid/

```
@Override
       protected Long doInBackground(Context... contexts) {

              // backend API code for creating the endpoint communication
              Validparkingspaceendpoint endpoint = ...
              ...

              try {
                     // calling the server method for retreiving the entries from the
                         database that satisfy the bounding box
                     CollectionResponseValidParkingSpace result =
                         endpoint.queryValidParkingSpace(mMinLatitude,
                                                         mMaxLatitude, mMinLongitude,
                         mMaxLongitude).execute();

                     // ValidParkingSpace result list
                     List<ValidParkingSpace> spaces = result.getItems();

              } catch (IOException e) {
                     ...
              }
       }
```

Listing 6.4: Query task code - uses the generated backend API for retreiving ValidParkingSpace objects from the database

6.4.1.3 Information Bubble

Additional information is displayed if the user touches a parking space on the map. Also, the user can press the navigate button in order to open navigation directions towards the selected parking location. The bubble is illustrated in fig. 6.11

6.4.2 Video Client

The clients that will use this option are the owners of the monitored vehicles. The idea is to send real-time video on the owner's smartphone when the alarm of the vehicle starts or when the owner wants to access this video data (on-demand). The Android prototype code for displaying the live video transmitted by the car pc uses the same JavaCV library as for the transmission. The mpegts frames are received through a UDP connection and the are decoded and ransformed into bitmaps. These bitmaps can then be displayed in using the ImageView class provided by the Android system. This process is resumed in fig. 6.12

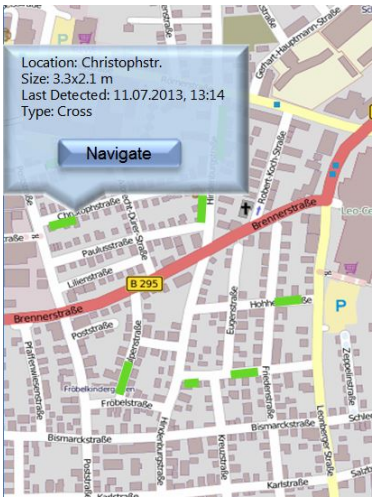

Figure 6.11: When the user touches a parking space on the screen, an information bubble will pop-up containing information about that parking space and the possibility to open navigation directions towards that parking space location (©OpenStreetMap contributors)

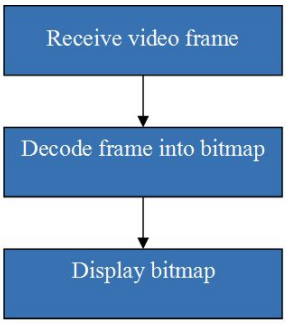

Figure 6.12: The video client receives the mpegts frame from the car pc through UDP, it is decoded it into a bitmap and displayed

7 Evaluation

This chapter presents several evaluations of the implemented system. Severel analysis were made in order to see how the system behaves in the real world.

7.1 Data Acquisition

Parking space data was acquired in order to analyze how the system behaves. For this, a separate setup was arranged that is able to grab parking space data from the ultrasonic sensors and also video data from the multicamera system of the test vehicle. The video is intended for visually analysing the captured parking space data from the ultrasonic sensors and

7.1.1 Goals

The goal of this campaign is to get data that can be used for testing the filtering algorithm as well as cost estimations for the implemented system. Also, the accuracy of the ultrasonic sensor system for detecting parking spaces is analysed. Another goal is to get an insight on the dynamic parking situation in a crowded city.

7.1.2 Required Hardware and Software

For the acquisition of the above mentioned data the following items were used:

- Mercedes B-Class test vehicle equipped with the automatic parking assist system, multi-camera system and Windows 7 PC

- Laptop with CANAlyzer[1] software for viewing and logging the messages that are passed through the CAN bus.
- Video grabber device used for transforming the NTSC video signal to USB signal that can be processed on the laptop
- Webcam for recording the front view of the vehicle while driving
- GPS device for logging the position
- Camera for taking pictures of the setup

7.1.3 Hardware Setup

The setup schematic is illustrated in fig. 7.1. Parking space data is acquired from the CAN bus and the top view video is recorded from the multicameara system and a windshield mounted webcam. A USB GPS device (Wintec G-Rays 2[2]) is used for logging the positions of the parking space data.

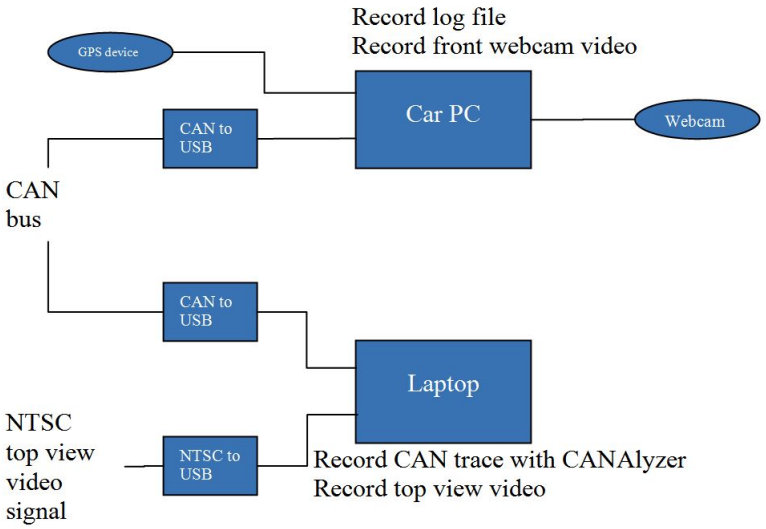

Figure 7.1: For data acquisition the car PC and a laptop is connected to the CAN bus for getting access to the parking space messages and two separate videos are recorded, the top view of the multicamera system and a front view using a webcam mounted on the windshield

[1] http://vector.com/vi_canalyzer_en.html
[2] http://www.wintec.com.tw/en/Products/gps/wbt_202.html

The camera mounting and the display of the car PC can be seen in fig 7.2.

Figure 7.2: The webcam that records the frontview is mounted on the windshield of the vehicle and the code that logs and records the video is running on the car PC

7.1.4 Code for Acquiring Data

A Java program was developed for acquiring video and parking space data. Its operation is illustrated in fig. 7.3.

The video frames are grabbed from the webcam that is installed on the windshield using the JavaCV library. When a parking space is detected by the ultrasonic sensor system, information about that parking space is added to the frame (timestamp, GPS location, width, length, type). The detected parking space entry is also saved in a .CSV log file together with the GPS location. For reading the GPS data in Java from the Wintec G-Rays 2 USB device, the RXTX library was used[3]. Also a separate GPS log file is saved where locations are acquired with a period of 500 ms. The video frames are recorded to .MPEG format with a frame rate of about 15 frames per second. A sample frame can be seen in fig. 7.4.

7.1.5 Itinerary

For data acquisition two separate areas were chosen. Both areas were scanned at different times to evaluate the system's functionality during different day times. The first location is in Leonberg, Germany, where four smaller, not so crowded residential streets were scanned.

[3]http://rxtx.qbang.org/wiki/index.php/Main_Page

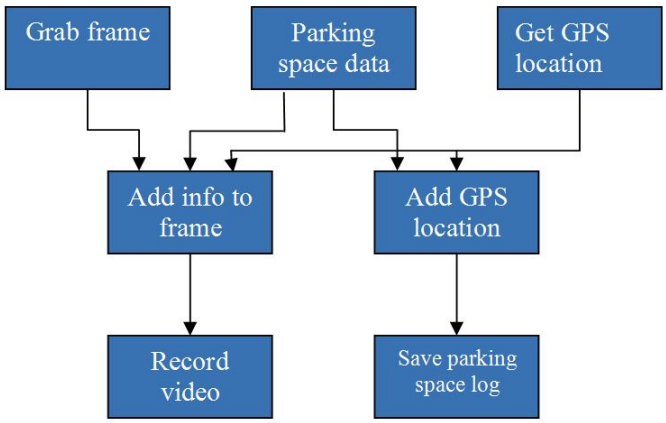

Figure 7.3: The Java program grabs each frame from the webcam and each time a parking space is detected, GPS location is read and textual information is added to the frame; also the parking space data is saved in a separate log file

Figure 7.4: The video frame contains information about each detected parking space and its GPS location

The area was selected to evaluate the behavior of the system in an area with a number of available parking spaces during the entire day. The total length of the path is around 500 m. We acquired data for one hour in the morning, between 10:00 and 11:00 and in the afternoon, between 13:00 and 14:30. The second location is close to the center of Stuttgart, a very crowded area, with a path length of around 1 km. Here we acquired data in the evening between 15:00

and 16:30. In addition, data was acquired in the morning between 8:00 AM and 9:00 AM, which is the morning rush hour where many residents leave for work or grocery shopping or students arrive to the nearby school, resulting in a more dynamic parking behavior. The two itineararies can be viewed in fig. 7.5.

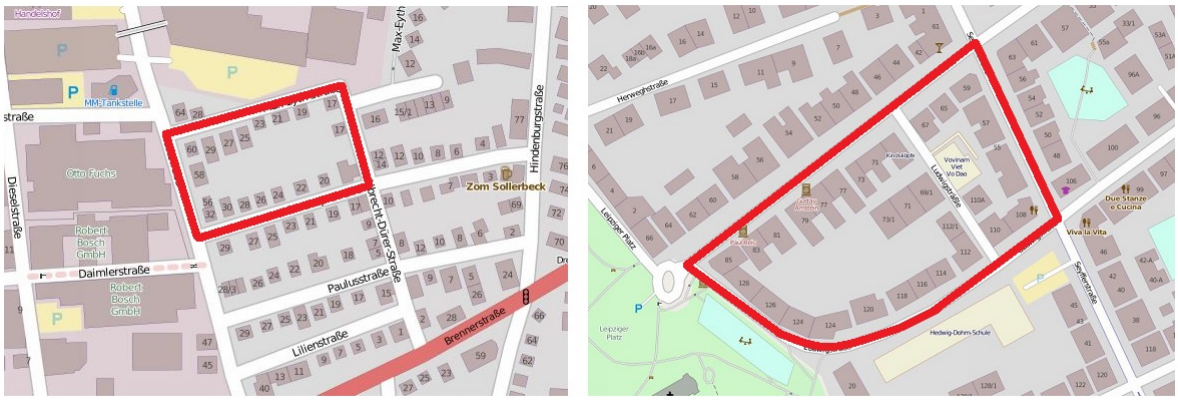

Figure 7.5: Two locations were chosen for acquiring parking space data: Leonberg -left and Stuttgart West - right (©OpenStreetMap contributors)

7.2 Analyzing the Performance of the Ultrasonic Parking Space Detection

After the data was collected from the specified locations, an analysis was performed in order to examine how accurate the detection of parking spaces using ultrasonic sensor system actually is. For this purpose, the recorded videos and log files were analyzed and the following information was extracted:

- *Number of parking spaces*: parking spaces that were present on the street from which the data was collected. This were extracted visually from the recorded video files.
- *Number of detected parking spaces*: spaces that are detected by the ultrasonic sensor system
- *Number of valid detections*: detected parking spaces that are actually parking spaces (true positives)

The acquisition of data consisted of multiple runs (laps) on each of the two locations. The above values were extracted from each run and average values were calculated. The results are resumed in table 7.1. It can be seen that the detection accuracy using ultrasonic sensors is pretty low at around 40-50%. In Stuttgart West, the accuracy is a bit higher since it was more crowded, and the cars were parked almost everywhere. The ultrasonic sensors were

able to detect the spaces between the vehicles. In Leonberg, since there were a lot of empty areas where nothing was present on the side of the road, the sensors did not have so good landmarks and the processing algorithms produced more errors than in the other case. Based on the detection accuracy, it is clear that a filtering algorithm is needed for splitting the valid detections from invalid ones.

		Leonberg	**Stuttgart West**
Parking spaces	Average detected	17	33
	Average valid	12	26
	Average valid detected	7	18
	Detection accuracy (%)	41%	54%

Table 7.1: Video and log files where analysed for each run on each location and average values were calculated for measuring the performance of the ultrasonic sensor detection

As mentioned, in Stuttgart-West data was collected in the morning between 8:00 and 9:00 where around 30

7.3 Street Matching Algorithm Test

A few GPS coordinates of parking locations acquired from Leonberg were used for testing the street matching algorithm. As seen in fig. 7.6 the positions are accurately corrected to match the street to which they are close to.

Figure 7.6: The left picture shows the unmatched parking location, their position being the one read from the GPS device. The right picture shows the position matched on the street (©OpenStreetMap contributors)

7.4 Data Filtering Algorithm

As mentioned before, using only the information provided by the ultrasonic sensors when detecting a parking space (width, length, type etc.) is not enough for distinguishing valid from invalid detections. The proposed concept relies on using also the information from the automatic parking assist system. From this we collect the messages that are created when a vehicle is parked in or out at a certain location. For testing how a learning algorithm could be applied to the system, the following features for each detected parking location are used:

- Number of times detected
- Number of times parked
- Users that detected parking spaces
- Users that parked into parking spaces

When vehicles are parked at a certain location a multiple number of times, the probability of that parking space to be valid should be higher. User information is also important since private parking spaces can be distinguished from public ones. If only one or two users park at the same location multiple number of times, it is most likely for that location to be a private parking space. On the other hand, if many users park at the same location then it is very probable to be a public parking space. Private parking spaces should not be validated since the information is not useful for users if they are not allowed to park at that location. The training data set is constructed from the log files of the collected parking space data. From these files the above mentioned *number of times detected* feature was extracted. The other three features have to be generated. *Number of times parked* has been added to the collected data set at certain parking locations (labeled as valid). Also, five users have been simulated, even though we used only one vehicle for collecting the data. *Number of times parked* values where added to the parking locations which we considered as valid by analyzing the videos.

Values between 1 and 7 where added. The data set was labeled by visually analyzing the video files. Data was labeled as valid at the parking locations where the vehicle parked more than three times and where at least two users detected and parked in a parking space.

The *libsvm*[4] library was used in Matlab to create a learning model for classifying the detected parking spaces by using support vector machine learning. This library provides a way to create the learning model by applying the *svmtrain* function to the provided training data set. Afterwards, new values can be predicted using this model by using the *svmpredict* function. The model "learned" that at the locations where the users also parked and where the user number is higher, the respective parking space is valid.

7.5 Feasibility Discussion

From the filtering algorithm tests was concluded that a parking space would be needed to be detected around 10 times and at least 2 drivers would need to park at least 5 times in a parking space to validate it. This would imply a large number of vehicles that are able to collect this data for creating a parking space map of an entire city. A solution would be to have a number of designated vehicles to drive around the city for a certain amount of time in order to detect but also park in and out into each parking space around that city. For a city with 1000 streets, 10 vehicles could realiye this map in about four months. If the automatic parking assist systems would become a standard option for production vehicles, then the process of building parking spaces maps in cities would be not so time consuming.

[4]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

8 Summary and Conclusion

In this chapter a summary of the work is presented and a conclusion from the performed work is drawn. The implementation in this paper is compared to previous works that were presented in chapter 2.

8.1 Summary

We presented at the beginning of the paper the main challenges of parking in crowded cities. Lost time, fuel consumption, vandalism and theft are only a few of these problems. Solutions that rely on crowdsourcing or third party providers try to bring parking space information to drivers, but they have their own problems, mainly related to the fact that the performance relies on the participation rate or, respectively on the information provided by other sources. The system presented by [Mathur et al., 2010] collects parking space data using ultrasonic sensors and uses an envirmonment fingerprinting technique for accurately determining the correct location of the detected parking spaces. However, this needs extensive processing and many passes over the same area. Our proposed solution relies not only on ultrasonic sensors, but on information provided by the vehicles that park in and out into/from parking spaces. All these data is collected, map matched and filtered so that an accurate map of valid parking locations is build and afterwards parking availability can be determined. Also we propose a solution for remote vehicle monitoring while the vehicle is parked. The solution is based on a multi-camera system integrated in the vehicle that can construct a top-view image of the vehicle. Live video can be transmitted to the owner's smartphone in the case the alarm is triggered or on demand.

We implemented a prototypical system that consists of a vehicle that is able to collect parking space data, a server that stores and processes the data and an Android client that can visualize parking space availability information. The server uses OSM data for accurately matching the position of each detected parking space to a certain street. Also, real-time transmission solutions have been tested for sending video images between the vehicle and an Android

smartphone. A JavaCV implementation has been chosed that uses UDP as the transmission protocol and .mpeg as the video format.

We sustained a data acquisition campaign for testing how accurate the ultrasonic sensor system is in detecting parking spaces and for collecting parking space data that can be used for testing filtering algorithm. We chose two locations on which we recorded video from the front-view of the vehicle in order to later analyse and count the parking locations. We also saved the CAN messages related to parking space detection in a log file. The analysis of the data proved that the accuracy of the ultrasonic parking space detection is pretty low for our intended purpose and for that a filtering algorithm is necessary. We tested how such an algorithm could work for filtering this parking space data.

8.2 Conclusion

With the more increasing traffic in crowded cities it is important to find better solution that assist drivers in finding parking spaces. We showed that such a system that can provide parking space availability to drivers can be constructed. Even though parking space detection by ultrasonic sensors is not accurate enough, this can be increased by using parked-in and parked-out messages that are already available on the vehicles that are equipped with automatic parking assist systems and by using map-based services for map-matching the parking space locations. A simple filtering algorithm with only four features can easily filter the valid parking locations from the invalid ones.

CD-ROM Content

Abstract

- Abstract of the thesis in English and German

Master Thesis

- MT_Radu_Georoceanu_Extending_Parking_Assistance_for_Automotive_User_ Interfaces

Code

- ParkConnect
- ParkConnect-AppEngine
- VideoStreamServer
- DataAcquisition
- VideoRecorder

Bibliography

[Armbrust et al., 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58. (Cited on page 22)

[Bose et al., 2011] Bose, R., Brakensiek, J., Park, K.-Y., and Lester, J. (2011). Morphing smartphones into automotive application platforms. *Computer*, 44(5):53–61. (Cited on page 5)

[Caliskan et al., 2006] Caliskan, M., Graupner, D., and Mauve, M. (2006). Decentralized discovery of free parking places. In *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*, VANET '06, pages 30–39, New York, NY, USA. ACM. (Cited on page 3)

[Campolo et al., 2012] Campolo, C., Iera, A., Molinaro, A., Paratore, S., and Ruggeri, G. (2012). Smartcar: An integrated smartphone-based platform to support traffic management applications. In *Vehicular Traffic Management for Smart Cities (VTM), 2012 First International Workshop on*, pages 1–6. (Cited on page 6)

[Chen et al., 2010] Chen, L.-C., Hsieh, J.-W., Lai, W.-R., Wu, C.-X., and Chen, S.-Y. (2010). Vision-based vehicle surveillance and parking lot management using multiple cameras. In *Proceedings of the 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IIH-MSP '10, pages 631–634, Washington, DC, USA. IEEE Computer Society. (Cited on page 7)

[Chen et al., 2012] Chen, X., Santos-Neto, E., and Ripeanu, M. (2012). Crowdsourcing for on-street smart parking. In *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, DIVANet '12, pages 1–8, New York, NY, USA. ACM. (Cited on pages 7 and 9)

[Coric and Gruteser, 2013] Coric, V. and Gruteser, M. (2013). Crowdsensing maps of on-street parking spaces. In *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, pages 115–122. (Cited on page 8)

[Foresti et al., 2004] Foresti, G., Micheloni, C., and Snidaro, L. (2004). Event classification for automatic visual-based surveillance of parking lots. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 314–317 Vol.3. (Cited on page 8)

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Springer series in statistics. Springer. (Cited on pages 24 and 26)

[Jung et al., 2010] Jung, H. G., Lee, Y. H., and Kim, J. (2010). Uniform user interface for semiautomatic parking slot marking recognition. *Vehicular Technology, IEEE Transactions on*, 59(2):616–626. (Cited on page 18)

[Kim et al., 2011] Kim, S., LEE, E., Choi, M., Jeong, H., and Seo, S. (2011). *Design Optimization of Vehicle Control Networks.* IEEE Trans. on Vehicular Technology, vol. 60, no. 7, pp. 3002 –3016. (Cited on page 13)

[Leitner-Fischer and Leue, 2011] Leitner-Fischer, F. and Leue, S. (2011). *The QuantUm Approach in the Context of the ISO Standard 26262 for Automotive Systems.* University of Konstanz and Steinbeis Transfer Center for Complex Systems Engineering. (Cited on page 14)

[Mathur et al., 2010] Mathur, S., Jin, T., Kasturirangan, N., Chandrasekaran, J., Xue, W., Gruteser, M., and Trappe, W. (2010). Parknet: drive-by sensing of road-side parking statistics. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 123–136, New York, NY, USA. ACM. (Cited on pages 7, 50 and 63)

[Montini et al., 2012] Montini, L., Horni, A., Rieser-Schüssler, N., and Axhausen, K. (2012). *Searching for Parking in GPS Data.* 12th Swiss Transport Research Conference. (Cited on page 7)

[Na et al., 2009] Na, K., Kim, Y., and Cha, H. (2009). Acoustic sensor network-based parking lot surveillance system. In *Proceedings of the 6th European Conference on Wireless Sensor Networks*, EWSN '09, pages 247–262, Berlin, Heidelberg. Springer-Verlag. (Cited on page 8)

[Ng and Chua, 2012] Ng, L. L. and Chua, H. S. (2012). Event recognition in parking lot surveillance system. In *Proceedings of the 12th Pacific Rim International Conference on Trends in Artificial Intelligence*, PRICAI'12, pages 875–878, Berlin, Heidelberg. Springer-Verlag. (Cited on page 8)

[Pruckner et al., 2003] Pruckner, A., Gensler, F., GRAEF, H., SPANNHEIMER, H., GRESSER, K., et al. (2003). Der parkassistent-ein weiteres innovatives fahrerassistenzsystem zum thema connecteddrive aus der bmw-fahrzeugforschung. *FORTSCHRITT-BERICHTE VDI. REIHE 12, VERKEHRSTECHNIK/FAHRZEUGTECHNIK*, (525). (Cited on page 16)

[Shoup, 1997] Shoup, D. C. (1997). *The High Cost of Free Parking*. Transportation Center, University of California, Berkeley, Berkeley, USA. (Cited on page 3)

[Talbot S.C., 2009] Talbot S.C., R. S. (2009). *Comparison of FieldBus Systems, CAN, TTCAN, FlexRay and LIN in Passenger Vehicles*. Distributed Computing Systems Workshops. ICDCS Workshops '09. 29th IEEE International Conference. (Cited on page 15)

[Velte et al., 2010] Velte, T., Velte, A., and Elsenpeter, R. (2010). *Cloud Computing, A Practical Approach*. McGraw-Hill, Inc., New York, NY, USA, 1 edition. (Cited on page 22)

[Vestri et al., 2005] Vestri, C., Bougnoux, S., Bendahan, R., Fintzel, K., Wybo, S., Abad, F., and Kakinami, T. (2005). Evaluation of a vision-based parking assistance system. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 131–135. (Cited on page 17)

[Winner, 2012a] Winner, H. (2012a). *Handbuch Fahrerassistenzsysteme Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort - Section B, Chapters10,11,12,13,14,15*. Vieweg + Teubner, Wiesbaden. (Cited on pages 11, 12 and 13)

[Winner, 2012b] Winner, H. (2012b). *Handbuch Fahrerassistenzsysteme Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort - Section F, Chapter 31*. Vieweg + Teubner, Wiesbaden. (Cited on pages 16 and 17)

[Yang et al., 2013] Yang, B., Fantini, N., and Jensen, C. S. (2013). ipark: identifying parking spaces from trajectories. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 705–708, New York, NY, USA. ACM. (Cited on page 7)

[Zaldivar et al., 2011] Zaldivar, J., Calafate, C. T., Cano, J. C., and Manzoni, P. (2011). Providing accident detection in vehicular networks through obd-ii devices and android-based smartphones. In *Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks*, LCN '11, pages 813–819, Washington, DC, USA. IEEE Computer Society. (Cited on page 6)

[Zimmermann and Schmidgall, 2010] Zimmermann, W. and Schmidgall, R. (2010). *Bussysteme in der Fahrzeugtechnik - Protokolle, Standards und Softwarearchitektur - 4. Auflage*. Vieweg Teubner. (Cited on page 14)

All links were last followed on January 05, 2014.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature