ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Institut für Informatik
Autonome Intelligente Systeme
Prof. Dr. Wolfram Burgard

Where to Park?
An In-vehicle Parking Space Occupancy Estimation and Guidance System

Masterarbeit

Igor Bogoslavskyi
März 2014

Betreuer: PD Dr. Cyrill Stachniss
Prof. Dr. Wolfram Burgard

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Arbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

(Igor Bogoslavskyi)
Freiburg, den 22. Februar 2012

Zusammenfassung

Parken ist zu einem der größten Probleme in Großstädten geworden. Man verliert viel Zeit, Brennstoff und Geld beim Versuch einen freien Parkplatz zu finden. Studien zeigen, dass die Parkplatzsuche für mehr als 10% des Verkehrsaufkommens in Städten verantwortlich ist und es bis zu 20 Minuten dauern kann, bis man einen Parkplatz gefunden hat.

Daher beschäftigt sich diese Massenarbeit mit der Vorhersage der Belegung von Parkplätzen. Diese Informationen können die Parkplatzsuche erheblich erleichtern, indem man diese in Planungsmethoden integriert, die häufig in Handys und GPS Navigationssystemen schon vorhanden sind. Wir präsentieren einen automatisierten Ansatz der Sammlung und Interpretation der Belegungsdaten von Parkplätzen mit Hilfe einer mobilen Platform. Dieser Ansatz erlaubt die Routenplanung mit der Hilfe von nicht nur räumlichen Informationen, sondern auch der Belegungsinformation über die Parkplätze.

Der vorgeschlagene Ansatz verwendet die Kamera des Fahrzeuges, um die geparkten Autos zu erkennen und diese den Parkplätzen zuzuweisen. Darüber hinaus wird die Belegungswahrscheinlichkeit jedes Parkplatzes über all Läufe hinweg geschätzt. Weiterhin lässt sich der Ansatz leicht erweitern, um die Belegungsinformationen an bestimmten Tagen oder zu bestimmten Uhrzeiten zu ermitteln.

Außerdem stellen wir auch einen Planer vor, der nicht nur die Position, sondern auch die Belerungswahrscheinlichkeiten mit in Betracht zieht. Der Planner sucht nach dem optimalen Weg, welcher die Zeit, die ist nötig um einen freien Platz zu finden und anschließend ein Ziel zu Fuß zu erreichen, minimiert.

Wir haben das System an Hand von echten Daten, die mit einem mobilen Roboter an mehreren Tagen auf einem Parkplatz gesammelt wurden, ausgewertet.

Abstract

Parking has become one of the biggest problems of the big cities. People lose time, fuel and money when searching for a free parking spot. Studies show, that the people cruising in the search for a parking space can generate more than 10% of an overall city traffic and the times until they eventually find a parking space can reach 20 minutes.

We therefore investigate the problem of how to map and predict the occupancy of parking spaces. This information, when integrated into navigation devices as they are found today in cell phones and GPS navigation systems, has the potential to drastically improve the experience of searching for a free parking space.

We present a proof of concept of an automated approach for gathering and interpreting the parking lot occupancy information using a mobile platform. It allows for planning the route not only with respect to spatial information but additionally taking into account parking spaces' occupancy information.

The proposed approach uses an in-vehicle camera setup to repeatedly detect the parked cars and assign them to the parking spaces in the area of interest. It estimates the occupancy probability of each found position suitable for parking throughout all runs. The framework is easily extendable to account for querying for occupancy information on specific date or even time of day.

Furthermore, we introduce a planner that relies not only on the position but also on the occupancy probability of each parking space in order to find the path that minimizes the expected time spent on searching for an unoccupied parking space and walking from the found one to the destination by foot.

We evaluated our system based on real world data gathered with a mobile robot over several days in a real parking lot.

Acknowledgments

I would like to express my thanks to everyone who supported me throughout the course of this work. In particular I would like to thank Cyrill Stachniss for his invaluable help. Without his experience and advices this work would not be possible. Additionally, I would like to thank Luciano Spinello for his ideas on the prototyping stage. For the help with the Obelix robot and EUROPA framework my thanks go to Bastian Steder, Michael Ruhnke and especially Rainer Kümmerle. They have significantly helped me finding my way around the system. I also thank everyone in the Autonomous Systems Lab for the great atmosphere there and particularly Wolfram Burgard for making such atmosphere possible. I would also like to thank Tobias Domhan for the time and effort that he has invested in proof-reading the draft version of this thesis. And, of course, I want to express my thanks to my parents that support all my decisions throughout my life. I would not be able to be where I am if not for all the effort they have put in me. Last but by far not least I want to thank Olga for her support and valuable critics of this work. She is one of the main reasons why I am able to reach my new horizons.

Contents

1 Introduction

We all have been in a situation, in which we were driving around in a city, trying to find a free parking space. We all know how daunting can it be to drive numerous circles around the center of the city with a high uncertainty about where and when we eventually find a free parking lot. With this problem we are not alone.

Rodrigue et al. [30] argue in their work "The geography of transport systems", that parking in the center of a modern big city with a population bigger than one million inhabitants is one of the most prevalent transport problems. They claim that cruising in the search for a free curb parking space may account for more than 10% of the local circulations as the drivers can spend up to 20 minutes looking for a parking spot.

Several aids for supporting drivers in finding parking spaces have been developed in the past. For example, modern garages often feature systems that help people find a free parking space by offering a counter, that shows the number of free parking spaces available at the moment. online maps usually contain the positions of such parking garages mapped with their GPS coordinates. This allows for these parking garages to be easily found, making use of an ordinary SatNav device or with basically any smartphone. Even though these occupancy counters make it easier to find a free parking spot, they only provide the current occupancy information and not its prediction and are only available for a small number of parking lots.

Furthermore, it is not always convenient to drive to a parking garage as sometimes they lie far from the destination of the driver. In such cases it is a lot more convenient to leave the car in one of the curb parking spaces.

Shoup [34], while addressing the problem of adjusting parking prices in order to reduce time spent on the search for a free parking space, found that it took between 3.5 and 14 min to find a curb space, and that up to 74 percent of the traffic was comprised by the people cruising for parking.

This results in having to spend time and fuel on repeatedly driving along the same route in a desperate hope that someone has left and there is a parking spot available now. There is significant amount of uncertainty in this task. And uncertainty is a state that may lead to frustration.

Rodrigue et al. [30] evaluates the parking difficulty by asking the parkers to express their parking impressions. His results indicate that the amount of parking information parkers had before their trips, was directly related to their parking search time, which in turn, influenced their perceptions of parking difficulty.

Dealing with such uncertainty is annoying for humans. Yet, people deal with it much better then robotized systems. In order to carry out the parking task in an autonomous fashion there needs to be more information provided. There must be a deterministic algorithm, that has a decision what an optimal action is in any moment in time. For this algorithm to function there has to be enough information on spacial representation of the parking spaces as well as a model, that accounts for the occupancy information.

To the extent of our knowledge, there is only a few solutions that provide information on the positions and occupancy of the curb parking spaces. One of the best known examples is the "Parking on demand" system by Pierce and Shoup [28, 29]. They present a system of parking meters that adapt the price of curb parking spaces with relation to current occupancy in the area. Along with intelligent parking meters they also present a smartphone application that provides live information on the occupancy and pricing of distinct parking spaces.

Even though this system proves to save time and money for the people that use it, we still see a lack of autonomy in it. In order for this system to be used in an autonomous fashion one must introduce a planner that considers not only spacial information but also occupancy and pricing.

Another downside of this system is its dependence on the fixed positions of the parking meters. This implies, that in order to use the presented system we first need to install the parking meters for every curb parking space. This can be an effort, time and money consuming task.

We argue, that there has to be a system that will be able to map the parking spaces, estimate their occupancy information and search for one in an optimal and fully autonomous way.

Autonomous vehicles are becoming a topic of a great interest. In recent years, automotive and software companies as well as the universities all over the world have been focusing on developing their own autonomous vehicles. These vehicles can already impressively well navigate in the cities [38, 23, 24, 12] and are able to take people to an arbitrary destination even in difficult urban environments. They also can learn how to adopt behavioral knowledge from the traffic [26, 35]. We believe, that these cars are to become safer, more efficient in the means of fuel consumption and more predictable

than the human drivers. Following the recent success of the Google self-driving car [25] and the fact that it is now officially allowed to use self-driving autos on public roads in California, Nevada and Florida, USA, we predict that in the upcoming years, people will spend less time driving by themselves and will rely more on robotized autos.

When it comes to parking, these vehicles can find out what a parking space is and are able to park there [5, 22, 45]. Despite being able to park in an autonomous fashion, they are yet unable to know where to search for a free parking slot by themselves and, as a consequence, where to search for a good one.

The Contribution of This Thesis

The task of this thesis was to develop a proof of concept system for supporting drivers in finding free parking spaces. The system should perceive cars in the environment, model parking places, and suggest trajectories towards parking lots.

As a result of this task, we designed such a system. Our approach requires only on-board sensing on a vehicle to perceive the environment. So far, we use a camera and a laser range finder to observe the scene and to detect cars from the perceptual input. This task is solved by a supervised learning approach using a support vector machine and histograms of oriented gradients as features. The detections are then fused into a model of the environment that allows to build up a representation of the scene estimating how often a parking space is free or occupied. Based on this representation, we define a Markov Decision Process that allows us to generate the optimal path, given the current knowledge, to find a parking space and at the same time park as close to the desired target location as possible. The novelty of this thesis is a proof of concept of an integrated system, that only relies on on-board perception, models the parking situation and generates optimal trajectories for finding a parking space and reaching the target location as fast as possible.

2 Related Work

The problem of finding free parking spots receives more and more attention with the continuous growth of the cities and their population and different approaches have been proposed to support drivers. We can divide the works in the field in two main categories depending on the sensor setup. The detection of the parking spaces is either carried out via the usage of static-mounted cameras or the on-vehicle sensor systems. We will further focus more on each of these options.

Detection with Static Monocular Cameras

The main purpose of such systems is to simplify the sensor setup for occupancy detection in the modern parking garages. Usually the parking lots' occupancy status is inferred from sensors, placed below or above every parking lot. It therefore comes in handy to replace all these individual sensors with one camera, that observes all parking slots at once. The occupancy status of each parking slot is then inferred from the visual information. These works differ in the features utilized for car detection as well as in methods of clustering these features.

Wu and Zhang [46] present an unsupervised system to monitor the occupied parking slots. The authors use a stationary monocular camera to detect parked vehicles. In their work, car detection is solely based on color. The authors use support vector machine (SVM) to cluster the features to distinguish between occupied and free parking slots. They pre-process the ground truth images of the parking spaces and search for color differences between the measurements and the ground truth measured for an empty parking lot.

True [41] proposes a similar approach. Alike with the previous papers he presents an unsupervised system for parking space detection. The author utilizes a static overhead camera. He uses human-labeled parking slots' positions to define the spacial arrangement of the parking spaces. The author distributes chrominance channels of the images from the camera into bins, storing a histogram of these values for each parking slot, then he classifies these histograms as free or occupied using either k-nearest neighbors or SVM.

He also presents a variant of the work based on classifying color patches situated around corner detector's regions of interest, but he proves it to be less efficient.

Tschentscher and Neuhause [42] utilize a static monocular camera and show a comparison of different features for occupancy analysis. As in the works mentioned above, the authors propose using an overhead camera pointing to the parking lot, where the distinct parking spaces are manually labeled in the images. They study the influence of the visual features such as color histograms, gradient histograms, difference of Gaussian histograms and Haar features on the detection performance. They also explore the variance in methods for training a classifier based on the chosen feature set, such as k-nearest neighbors, linear discriminant analysis and SVM. They achieve detection rate of 98% while performing in real time.

Ichihashi et al. [17] shows a system that uses a set of overhead cameras for occupancy detection in indoor and outdoor scenes. The authors set the focus towards the clustering algorithms, showing that the performance of the detector based on the fuzzy c-means (FCM) clustering and the hyper-parameter tuning by particle swarm optimization (PSO) outperforms SVM both in speed and accuracy of detection.

Huang and Wang [15] and Huang et al. [16], use a static monocular overhead camera and a 3-layer Bayesian hierarchical framework (BHF). The authors of the paper specifically address the challenges of vacant parking space inference that come from dramatic luminance variations, shadow effect, perspective distortion, and the inter-occlusion among vehicles showing great detection rates of up to 99%.

Some works change the focus from detecting the parked cars to detecting the appearance of the markers pre-painted on each parking slot. Yusnita et al. [47] proposes a system to detect occupancy status of the parking spaces in parking garages. They are using an overhead, strictly vertically oriented camera. The parking spaces are manually labeled by a marker that states their occupancy state. For each query parking slot the authors produce a binary image which is then analyzed to find if the parking space is empty or occupied. This process is carried out by comparing the shape of the detection with the shape of the prior that reassembles the circle shape of the marker drawn on the floor of the parking slot.

In-vehicle Detection

The main area of interest for us is the in-vehicle detection of the parking slots. We are specifically interested in the works that present approaches relying on the fusion of range finder data with visual information.

Fintzel et al. [13] and Abad et al. [1] model free parking slots locally based on the 3D sonar sensor mounted on the car with conjunction with a visual 3D estimation setup based on the tracking of the interest points in the images seen from different angles.

In these papers the authors use the ultrasonic sensor and 3D vision sensor that detects points of interest on the bodies of the parked cars and tracks them to compare their positions from different points of view. The cars are then modeled by vertical planes of two different orientations that are fit into the 3D data from the sensors. The authors model the empty parking slots by the empty spaces between these planes.

Coric and Gruteser [9] also move focus from static overhead cameras on a developed parking lot to estimating parking slots' positions for on-street parking. They use an ultra-sonic sensor mounted on the side of the car to estimate the parking possibilities along particular streets. The authors use a straightforward threshold method to distinguish between free and occupied space. They also make use of an idea that the more times a car is observed in any arbitrary place the more likely it is that the space occupied by this car is a valid parking slot. As a result, all measurements are incorporated with the GPS measurements to form a global map of parked cars, which the authors use to detect wrongly parked cars.

Following the focus on the in-vehicle sensor setup, Schmid et al. [32] utilizes an on-board short-range radar. The measurements from three radars are stored into a 3D occupancy grid, that represents the local surroundings of the car. Free parking spaces are detected on the given grid by analyzing occupancy grid cells corresponding to curb and other parked cars. This provides an estimate of the parking space in 3D that allows for precise parking.

Suhr et al. [37] proposes a system that is able to detect parking spaces from 3D data acquired from stereo-based multi-view 3D reconstruction. The authors select point correspondences using a de-rotation-based method and mosaic 3D structures by estimating similarity transformation. While relying solely on this information and not using the odometry they are able to achieve reliable results with the detection rate of 90%.

Some works, however focus not on car detection, but on detecting the parking slots' markings. This approach proves to be useful when searching for free outdoor parking spaces.

Suhr and Jung [36] present a fully automatic system for detection of parking slots' markings. They utilize a tree structure performing a bottom-up and a top-down approach in order to classify parking slots as such and leave out false detections.

In this thesis, we present an in-vehicle detection system that is capable of detecting parked cars, modeling their spacial position as well as occupancy information inferred from multiple observations. For this task we utilize the in-vehicle sensor setup, consisting of a stereo camera and, optionally, a laser range finder. We estimate the occupancy probability for each parking slot in a global map and present a planner capable of finding a route to a free parking space considering spacial and occupancy constraints.

3 In-vehicle Parking Space Occupancy Estimation and Guidance System

3.1 Overview

In this chapter, we describe in detail the three main components of our proof of concept system prototyped for in-vehicle sensor setup: visual car detection, creation of the map of parking lots capable to store their occupancy information and a planner able to find a free parking lot in the minimum expected time. We also present the way of integrating these parts into a single consistent framework. We start by a coarse outline of these different sections and present each of these subparts in detail later in the chapter.

- *Visual Car Detection:* We visually detect cars in the images taken from the stereo camera and estimate how far from the image plane are they situated by fusing the detections with the depth data, received either from the stereo disparity or from the laser range finder (Section 3.2).
- *Parking Lot Modeling:* The positions of the detected cars are fused into one global representation of the world observed by the agent. This representation also contains occupancy information, perceived through multiple runs (Section 3.3).
- *Action Planning:* In the modeled environment we search for actions that minimize the expected time spent on searching for a parking space and walking from it to the goal (Section 3.4).

3.2 Visual Car Detection

We start by motivating the choice of the visual detection framework. There is a number of object detection methods to choose from. We investigate three different approaches, utilizing different visual features: Haar features, local binary patterns (LBPs) and histograms of oriented gradients (HOGs).

Viola and Jones [43] introduce the Haar feature based object detection. Their approach provides the detection rate of up to 95% when applied to face detection. The main drawback of the proposed method for our setup is that it suffers from the changes in the shape of the object, rotations, occlusions and from the illumination changes. For the task of car detection it is important to allow for a certain degree of deviation for the reason that the cars can differ in shape and angle from which they are observed. They also can be occluded by people walking in front of them or other cars. In addition to these variations, we may also observe them in different lighting conditions.

Trefný and Matas [40] utilize the local binary patterns (LBPs). This approach addresses the illumination issues and overall boosts the detection rate and performance speed. It is, however, similarly to Haar features, vulnerable to rotation and shape changes.

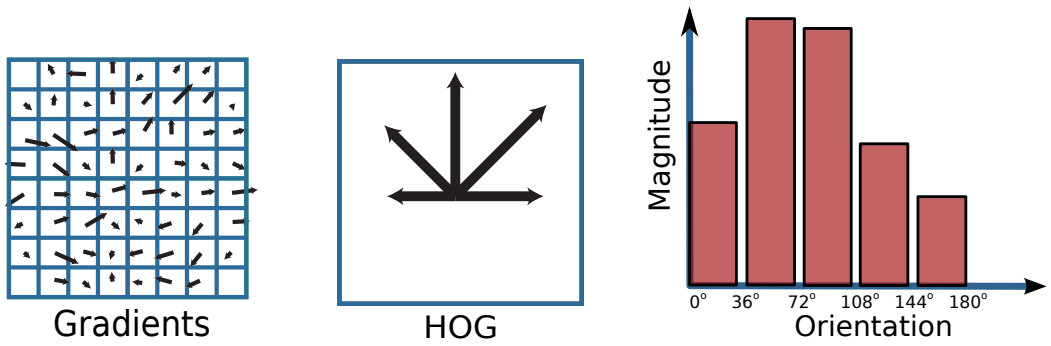
Moreover, to the extent of our knowledge, LBP-based methods are slow at the training stage. Despite not being a crucial measure of an algorithm performance, the training time is still an important argument, as it influences the usability of the method, especially on implementation stage.

Throughout the development of this thesis, we tried all these methods and decided to apply the HOG-based detector following the work of Dalal and Triggs [11]. It relies on the gradient-based features and is therefore invariant to illumination and brightness changes. Additionally, the presented method stays robust to slight changes in rotation and shape.

3.2.1 HOG Detector

The HOG representation has several relevant properties for the problem under consideration. It captures edge or gradient structure that is very characteristic of local shape, and it does so in a local representation with an easily controllable degree of invariance to local geometric and photometric transformations: translations or rotations make little difference if they are much smaller than the local spatial or orientation bin size.

Following Dalal and Triggs [11], we make use of the histograms of oriented gradients placed in a dense grid on the query images. The descriptor is created by computing the gradient magnitude and orientation in each pixel of the detection window and storing the orientations in a histogram. For each 8×8 pixel block in the image we form a histogram by assigning the gradient orientations to distinct histogram bins, that cover all possible orientation angles of the gradient. We do not take into account the direction of the gradient vector, focusing only on its orientation. This allows for the histogram bins to cover an angle of 180° instead of 360° while still storing all possible orientations. A



(a) (b) (c)

Figure 3.1: This figures shows gradient orientations for every pixel of the detection window (a), the corresponding histogram (c) and its visualization as a glyph (b). The directions of the arrows in (c) represent the angle covered by each bin, the lengths of arrows correspond to the magnitude of the gradients that fall into the corresponding bin. In this example, for better visualization, the histogram consists of 5 values, covering the range of 180°, meaning that each bin covers the angle of 36°, whereas the experiments in this thesis, use the histograms consisting of 9 bins.

visualization of an example histogram can be seen in Figure 3.1.

To detect the front and rear sides of the cars, we set the size of the hog descriptor window to 128×128 pixels. The idea behind this decision is based on generally square-like shape of the cars, when seen from front or rear. A square drawn around the car contains higher percentage of useful information in comparison to a rectangle. To detect the side of the car, the descriptor window is changed to 128×64 pixels following the same logic.

These window sizes yield the number of the histograms that fit inside each window which is the number of 8×8 pixel blocks contained in the descriptor window. This results in 256 histograms for front/rear view and 128 histograms for the side view.

We form a vector from all histogram values contained in each of 256 or 128 histograms that describe the detection window. Following Dalal and Triggs [11], we consider that each histogram consists of 9 distinct bins. This results in the vectors of sizes respectively 2304 and 1152 values for front/rear and side view detection windows. We view this vector as a point in the space of same dimensionality and seek for a hyperplane to classify these points into positive and negative ones with the use of the support vector machines. This is illustrated in Figure 3.3.

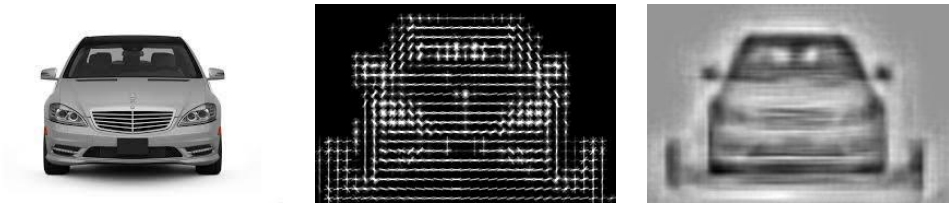
3.2.2 SVM Classifier

In order to carry out the decision in the test data, we train a linear Support Vector Machines (SVM) classifier on top of all HOG descriptors. Alternatively, a more complex decision boundary can be used, but linear SVM, despite its simplicity, provides reasonable results. We show the detection performance in the Experimental Results section.

Cortes and Vapnik [10] presented the support vector machines as a method to solve a two class classification problem. For a dataset of points in n dimensional space SVM finds a hyperplane ($n-1$ dimensional plane) that optimally assigns them to two classes maximizing the distance between the two resulting datasets. In the linear SVM the decision boundary is linear.

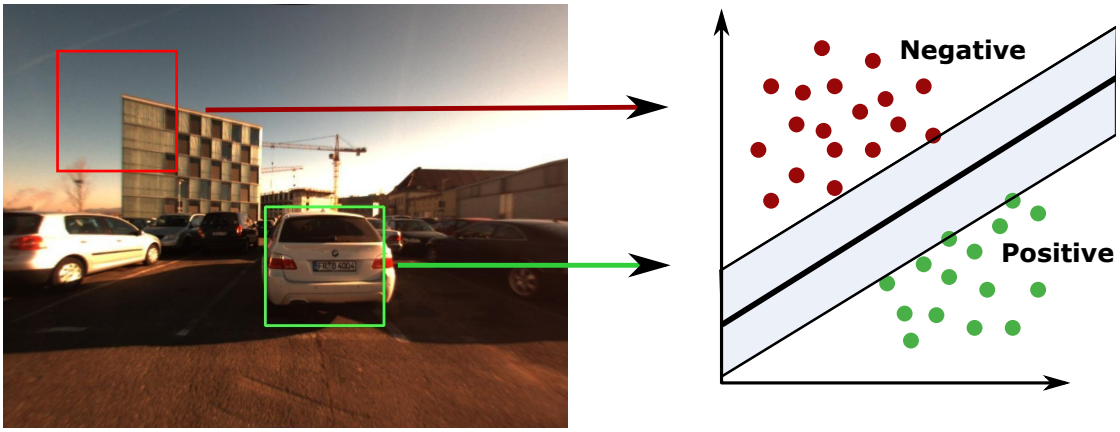
SVM is a well-known and arguably the most popular algorithm in binary clustering, therefore many implementations are readily available. In this work we make extensive use of libSVM library by Chih-Chung and Lin [8].

Based on the trained classifier, we carry out the detection in a cascade fashion via the sliding window approach. That is: for every image we generate a cascade which is effectively a set of images, subsampled with different scaling factors. For every image in a cascade we slide a detection window of the pre-defined size over the image. Every sliding window content yields a HOG descriptor which we test against the pre-trained SVM classifier to find out to which side of the decision boundary it is classified. If the current HOG belongs to the side where the SVM has assigned the cars from the training dataset then the current sliding window is a region of interest and contains a car. We store each detection as a rectangle by saving the pixel positions of the corners of the detection window around the car.



(a) **(b)** **(c)**

Figure 3.2: This figure shows a car image (a), a representation of the histogram values using glyphs (similar to Fig. 3.1b) (b) and a HOG inversion image via HOGles [44] (c). The shadows around the car have a high influence on the corresponding HOG. Such details are smoothed out via learning the descriptor from the training data.



(a) (b)

Figure 3.3: Illustration of HOG detection window (a) and clustering the corresponding multi-dimensional points via fitting a hyperplane using SVM (b) which yields the maximal distance between the datasets.

3.2.3 Depth Information

In this section, we describe in details how the depth information is acquired and combined with the visual detections described in the previous sections.

Stereo Camera

In order to find the depth from the stereo camera, we calculate the disparity image from left and right images taken from the video stream. Following Konolige [18], we carry out the disparity computation making use of block matching along the epipolar lines in the image. Knowing the internal camera parameters we reconstruct the relative 3D position of each pixel from the disparity image.

The distance along the camera z axis: $Z = \frac{fB}{d}$, where f is the focal length of the camera, B is the baseline and d is the disparity. Given Z, we focus on finding X and Y coordinates from the projection equations:

$$X = \frac{uZ}{f} \qquad Y = \frac{vZ}{f} \tag{3.1}$$

where u and v are the pixel position in the $2D$ image and X, Y, Z are the $3D$ coordinates that correspond to the pixel (u, v) in the camera frame. $3D$ position that corresponds to

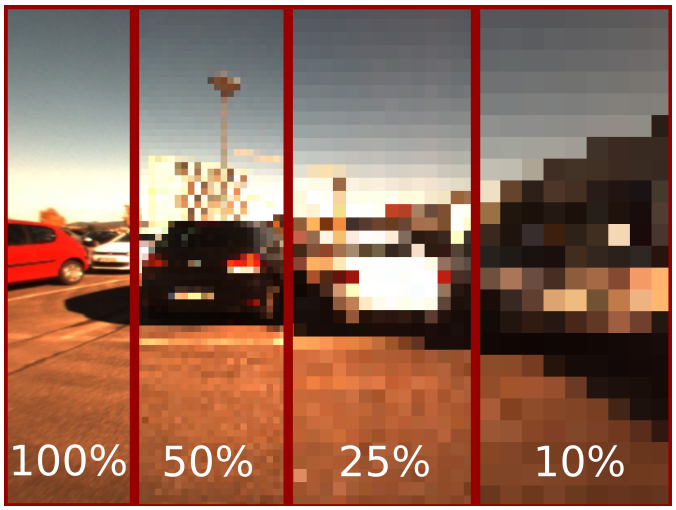


Figure 3.4: Each part of this figure illustrates a cut from the query image, subsampled with the corresponding scaling factor. For example, a 1024×768 pixel image subsampled with the factor of 50% has the size of 512×384 pixels. A set of images with different scaling factors form a cascade and are used as base for the sliding window approach for object detection. This allows us to detect objects of similar structure on different scales.

every pixel in the image allows us to combine depth information with visual detections.

Considering, that each detection is stored in the form of a region of interest in the image coordinates, we are able to accumulate the depth of all pixels that fall into it to estimate the joint depth of the detected car.

To find the distance to the car, we consider taking the median of all depth values that fall into the corresponding region of interest. These depth values contain high variance. The first reason for this is that they originate on the surface of the car and the variance represents the difference in the distance between different parts of the car. The second reason comes from the method in which these depth values are acquired. The depth map from the stereo camera setup proves to be noisy in the urban environments with homogeneous regions like the sides or the hood of the car. Moreover parts of the environment are seen through the windows of the car, adding additional noise to measurements. An example can be found in Figure 4.2.

This high variance of the input data results in the high uncertainty of the precision in results' measurements. We further describe the method that uses the laser range finder for the same task of finding the relative position of the detected car to the camera.

Laser Range Finder

In order to use a laser range finder to detect the parked cars we used a EUROPA robot Obelix which has a laser with an opening angle of 270° mounted approximately on the human knee level. This setup proves to be more robust in terms of finding the exact relative position of the detected car.

We search for the beams, that span through the area covered by the image. The camera is mounted on the same vertical axis with the laser as in Figure 4.3c. This yields the choice of the laser beams, that contain depth values associated with the car:

$$Z = \{\text{beam}_n \mid \forall n : \beta_l < \alpha_{\text{camera}} + \alpha_0 + \alpha_n < \beta_r\} \tag{3.2}$$

Here, α_{camera} is the angle at which the camera points as perceived in the laser frame, β_l and β_r are the angles of left and right margins of the detection bounding box in the camera frame, α_n is the angle of the n-th beam in the laser frame. α_0 is the angle of the first beam in the laser frame. In the setup we use, $\alpha_0 = -3\pi/4$ and $\alpha_{\text{camera}} = -\pi/2$.

The endpoints of the beams in Z provide us with consistent depth information. In order to account for possible occlusions or measurement errors we take the median of these values.

3.3 Parking Lot Modeling

The methods mentioned above provide the cars positions relative to the camera. Integrating this information with GPS coordinates or pose estimates based on simultaneous localization and mapping [20, 19, 21], we calculate the positions of the detected cars in the world frame. In this section we present two approaches for storing these detections. We also describe the approach to modeling the positions of the parking lots and their occupancy probability.

3.3.1 Occupancy Grids

The first approach that we consider utilizes the grid maps. Grid maps partition the space into a grid of rectangular cells. Each grid cell contains information about the corresponding area in the environment. We make use of one particular realization of grid maps — the *occupancy grid maps* as presented by Moravec and Elfes [27]. Occupancy grid maps assume that each grid cell is either occupied by an obstacle or free. In our case, each cell stores the probability that the particular cell is occupied by a car.

The occupancy grid maps are an efficient approach for representing uncertainty. Grid maps allow for detailed representation of an environment without a predefined feature extractor. As they have the ability to represent occupied space, empty space and unknown (unobserved) space, grid maps are well suited for tasks such as path-planning, obstacle avoidance and exploration. In contrast to most representations based on features, grid maps offer constant time access to cells.

Let $P(\mathrm{a}^i)$ be an occupancy probability estimate of a cell i in the environment. Considering the probabilistic nature of occupancy of every distinct cell we integrate multiple measurements, taken in different times into one model. Following the work of Moravec and Elfes [27], we obtain an update rule for $P(a_t \mid z_{1:t})$. We apply Bayes' rule and obtain

$$P(a_t \mid z_{1:t}) = \frac{P(z_t \mid a_t, z_{1:t-1})P(a_t \mid z_{1:t-1})}{P(z_t \mid z_{1:t-1})}. \tag{3.3}$$

We then compute the ratio

$$\frac{P(\mathrm{a}^i \mid z_{1:t})}{P(\neg\mathrm{a}^i \mid z_{1:t})} = \frac{P(z_t \mid \mathrm{a}^i, z_{1:t-1})}{P(z_t \mid \neg\mathrm{a}^i, z_{1:t-1})} \frac{P(\mathrm{a}^i \mid z_{1:t-1})}{P(\neg\mathrm{a}^i \mid z_{1:t-1})}. \tag{3.4}$$

Similarly, we obtain

$$\frac{P(\mathrm{a}^i \mid z_t)}{P(\neg\mathrm{a}^i \mid z_t)} = \frac{P(z_t \mid \mathrm{a}^i)}{P(z_t \mid \neg\mathrm{a}^i)} \frac{P(\mathrm{a}^i)}{P(\neg\mathrm{a}^i)},$$

which can be transformed to

$$\frac{P(z_t \mid \mathrm{a}^i)}{P(z_t \mid \neg\mathrm{a}^i)} = \frac{P(\mathrm{a}^i \mid z_t)}{P(\neg\mathrm{a}^i \mid z_t)} \frac{P(\neg\mathrm{a}^i)}{P(\mathrm{a}^i)}. \tag{3.5}$$

Applying the Markov assumption that the current observation is independent of previous observations given we know that a cell contains a parked vehicle gives

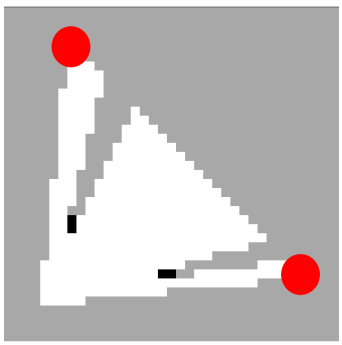
$$P(z_t \mid \mathrm{a}^i, z_{1:t-1}) = P(z_t \mid \mathrm{a}^i), \tag{3.6}$$

and utilizing the fact that $P(\neg\mathrm{a}^i) = 1 - P(\mathrm{a}^i)$, we obtain

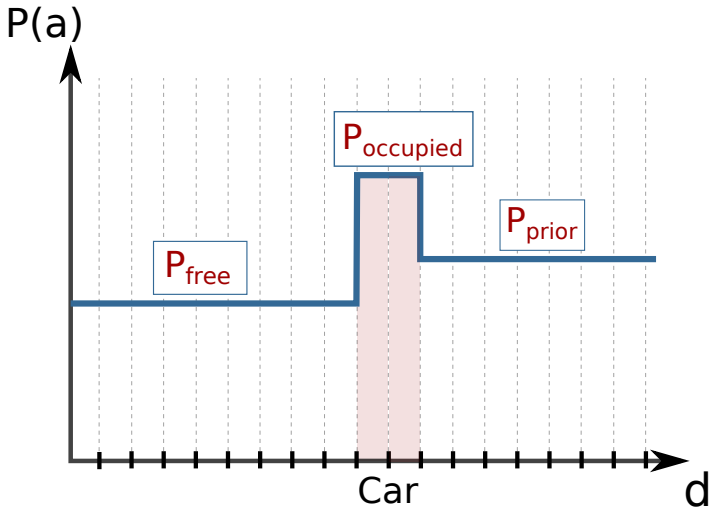
$$\frac{P(\mathrm{a}^i \mid z_{1:t})}{1 - P(\mathrm{a}^i \mid z_{1:t})} = \frac{P(\mathrm{a}^i \mid z_t)}{1 - P(\mathrm{a}^i \mid z_t)} \frac{P(\mathrm{a}^i \mid z_{1:t-1})}{1 - P(\mathrm{a}^i \mid z_{1:t-1})} \frac{1 - P(\mathrm{a}^i)}{P(\mathrm{a}^i)}. \tag{3.7}$$

This equation can be transformed into the following update formula:

$$P(\mathrm{a}^i \mid z_{1:t}) = \left[1 + \frac{1 - P(\mathrm{a}^i \mid z_t)}{P(\mathrm{a}^i \mid z_t)} \frac{1 - P(\mathrm{a}^i \mid z_{1:t-1})}{P(\mathrm{a}^i \mid z_{1:t-1})} \frac{P(\mathrm{a}^i)}{1 - P(\mathrm{a}^i)}\right]^{-1} \tag{3.8}$$



(a) Occupancy grid update of the camera field of view between the margin points marked with red circles containing two detected cars seen in black.



(b) Observation model $P(a^i|z_t)$, updating the cells before the detection with P_{free} probability, the ones that contain a car with P_{occupied} and leaving the ones, covered by the detection untouched.

Figure 3.5

In this approach we do not explicitly model the parking lots, but they are described by groups of cells in spacial proximity, that were repeatedly observed to contain cars.

In order to fully define the equation given above, we need to define the observation model $P(\mathrm{a}^i \mid z_t)$.

As presented in Figure 3.5b, we define the observation model along the rays, that span from the camera position to the pre-defined $d_{\max}$ and along the defined for the camera field of view. In order to find all the cells of the occupancy grid map that fall into the field of view of the camera we compute left and right margin points. Following the work of Bresenham [4] we search for the cells that form the frontier of the camera's field of view. The frontier consists of the cells that lie on the line between the above-mentioned left and right margin points. The margin points are shown with red circles in Figure 3.5a.

We carry out the same Bresenham algorithm from the camera cell c to each query cell c_i in the frontier. Whenever we encounter a cell containing a car, the next cells within the size of the car are also updated as occupied. All the ones that come afterwards are not updated as "not visible". These can be seen in an example in Figure 3.5a as gray

"tails" behind the car detections.

We formally define the observation model $P(\mathrm{a}^i \mid z_t)$ as follows:

$$P(a^i \mid z_t) = \begin{cases} P_{\text{free}}, & \text{if } d(c_i) < d(\text{detection}) \\ P_{\text{occupied}}, & \text{if } d(\text{detection}) < d(c_i) < d(\text{detection}) + s_{\text{car}} \\ P_{\text{prior}}, & \text{if } d(c_i) > d(\text{detection}) + s_{\text{car}} \end{cases} \tag{3.9}$$

where, $d(\textit{detection})$ is the distance to the detection point, $d(c_i)$ is the distance to the i_{th} cell, s_{car} is the length of the car. We consider all distances to be measured along the ray from the Bresenham algorithm.

Considering the current setup of the sensor that generates only few detections, overwhelmed by a number of observations detecting cells as free, we argue for setting P_{prior} to 0.5, as we have no prior knowledge on the occupancy of each cell, P_{free} to 0.45 to introduce only a slight update when observing an unoccupied cell and P_{occupied} to 0.95 which emulates a high certainty when detecting an occupied cell.

3.3.2 Static State Binary Filter in Pre-defined Positions

However, occupancy grids have their drawbacks. One of them, especially for our setup, is the discretization error. Whenever there are multiple detections of the same car from different view points it can happen, that the detection is assigned to different cells of the occupancy grids. This leads to the problem of data association and therefore difficulties in creating a meaningful accumulated map.

To avoid the discretization issues we make use of the pre-defined parking lot positions. These can be set from manual measurements on the ground or from aerial images. The rest of the theory stays untouched. For each parking lot we model the occupancy probability likewise to the occupancy grid maps approach via the static state binary Bayes filter as defined in Eq. (3.8).

The observation model, however, differs significantly from the one used in the occupancy grip maps. Whenever we detect a parked car, we search for the closest parking lot and assign the detection to it. The parking lot occupancy is updated as occupied.

Additionally, we consider the detection of the parked cars to be divided in sessions. For example, we consider driving near every parking space in the whole parking area to be one session. The parking lots that were not updated as occupied during the session are updated as free.

We explicitly stress, that this approach, while needing a pre-processing step of mapping the parking lots, allows for storing both spacial and occupancy information in one distinct structure.

3.3.3 Occupancy Inference

We assume the parking lots to be observed in sessions taking place at different times. We also assume the world to be static during each session that is: the changes in the world occur only from session to session.

To estimate the occupancy probability we count the times a state has been observed as occupied and free. Let state s be observed in a sequence of measurements carried out in times $t_0 \dots t_n$. After each observation the parking lot contains the occupancy probability inferred from the current measurement session. We compute the prediction of the state's occupancy by counting the times the states was seen as occupied (N_{occupied}) and free (N_{free}) through the course of all measurement sessions at times $t_0 \dots t_n$. Therefore, for each state s:

$$N^s_{\text{occupied}} = \sum_{k=0}^{n} i^s_k \tag{3.10}$$

where i^s_k is an indicator function:

$$i^s_k = \begin{cases} 1, & \text{if } p^s_k > P_{\text{prior}} \\ 0, & \text{if } p^s_k \leq P_{\text{prior}} \end{cases} \tag{3.11}$$

Here, p^s_k is the occupancy probability of state s at time t_k. We compute the occupancy probability prediction $P(x = \text{occupied})$ for an arbitrary state s as:

$$P(x_s = \text{occupied}) = \frac{N^s_{\text{occupied}}}{N^s_{\text{occupied}} + N^s_{\text{free}}} \tag{3.12}$$

3.4 Action Planning

In this section we formally define the action planning framework that searches for a path to a free parking lot. It, however, depends on the application what an optimal solution to this problem is. We therefore explicitly define the criterium of optimality that we consider throughout this work.

There are typically subareas of the parking area, which are better in the sense of their distance to a common destination. This also means that the "better" areas of the parking

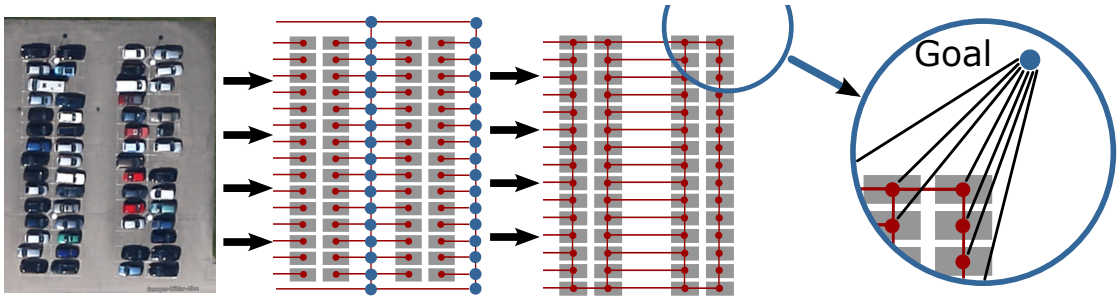


Figure 3.6: Graphs based on the parking area. The first graph representation defines different states for positions where the car can drive and the ones where it can only park. The second graph is a simplification of the one in the middle. The vertices in it correspond to the parking spaces. The edges between the vertices represent possibilities to reach one state from another, while the actual route is not represented explicitly. The correspondent spacial information is instead stored in the edges. Every state is additionally connected to the goal state.

lots are more likely to be occupied than the other. An illustration of such situation can be seen in Figure 4.5a. The right part of the parking lot is significantly more densely occupied than the left one. The relation between the position of the parking lots and their occupancy leads to a trade-off between driving straight to the closest-to-goal, but likely occupied parking lot, and parking in a more distant area, most likely free.

It usually takes less time to drive a car than to walk by foot, but it can take longer to find a free parking lot closer to the goal than walking from a more distant one. Given the constraints described above, the framework that we present focuses on finding the solution that minimizes the expected time spent on finding a free parking lot and walking from the found one to the goal.

3.4.1 Parking Area as a Graph

We represent the parking area as a graph $G = (V, E)$. The states are defined as a set of vertices V, where each vertex represents a point in $\mathbb{R}^2$. Each edge $e \in E$ is a function $V \to V$, that corresponds to the actions that can be carried out from one state to another.

In order to define the set of vertices for modeling the parking area as a graph, we consider the movements of the car that are allowed by the parking area's structure. We present a representation of this structure in Figure 3.6. The graph to the left has two distinct sets of vertices. The ones in blue represent the points in $\mathbb{R}^2$ that model all possible paths around the parking lot, suitable car movements. The red vertices represent the positions of the parking spaces, which are additionally connected to the goal state

(not shown in the figure). The graph to the right represents the reduced version, where all vertices correspond to the parking spaces' positions. By reducing the graph in such way we move the focus from modeling the actual car movements to the observations the agent makes, that is: it is irrelevant for the planner what the actual position of the car is if it is able to observe a parking slot. We define the edges so that they hold the spacial information about the connectivity and distances between the parking lots.

3.4.2 MDP Definition

Tipaldi and Arras [39] show, that Markov decision processes (MDPs) provide a way to maximize joint rewards instead of greedily going for the best possible goal. We follow similar approach, defining rewards and transition probabilities based on the environment specific to our problem.

Following Bellman [3], we define the MDP by the initial state s_{start}, transition function $T(s \mid s', a)$ and a reward function $R(s, s', a)$. Here, $T(s \mid s', a)$ denotes the probability of reaching state s if action a is carried out in state s'. The transitions in the model are assumed to be Markovian. That is, the probability of reaching s from s' depends only on s' and not on the history of earlier states.

Actions

In order to define the transition and reward functions we define the set of actions A that the agent can perform in the vertices of the graph. It consists of five distinct actions: "up": $\uparrow$, "down": $\downarrow$, "left": $\leftarrow$, "right": $\rightarrow$ and "park". Formally:

$$A_{\text{move}} = \{\uparrow, \downarrow, \leftarrow, \rightarrow\} \tag{3.13}$$

$$a_{\text{park}} = \text{“park”} \tag{3.14}$$

$$A = A_{\text{move}} \cup a_{\text{park}} \tag{3.15}$$

Transition Function

For four actions from A_{move}, we consider the probability of moving from any state to the next one (if possible) to be equal to 1, while the probability of parking in a particular state is equal to the occupancy probability of the state:

$$\forall (s, s') \in E, \forall a \in A_{\text{move}} : \quad T(s \mid s', a) = 1 \tag{3.16}$$

$$\forall s \in V, (s, s_{\text{goal}}) \in E : \quad T(s \mid s_{\text{goal}}, a_{\text{park}}) = P(\text{free}) \tag{3.17}$$

where $P(\text{free})$ is a probability of a parking lot to be free. To be consistent, we set the remaining probabilities to the values that guarantee that $\forall s, s' \in V, (s, s') \in E, \forall a \in A : \sum_s T(s \mid s', a) = 1$. This ensures that in each state the outcome of each action the agent can take is defined. It resolves to staying in the same state when trying to carry out an unavailable or improbable action.

For example: it is clear, that staying in the left most corner of the parking lot and carrying an action to go left, should result in staying in the same spot.

Expected Time Minimization to Rewards Maximization

We choose the reward function in such way that it guarantees a close relation between the optimal MDP solution and minimizing the expected time the agent needs in order to find a free parking lot.

Let route R be an arbitrary path to the target location to which the agent needs to travel to. The route is effectively described as a sequence of moves the agent makes on his way to the goal. Given the transition model presented above, we argue, that every move the agent makes can be described by two states s_1, s_2 from V along with an action $a \in A$ that brings the agent from state s_1 to state s_2, that is: such that $T(s_2 \mid s_1, a) > 0$. The agent starts in the initial state s_{start} and travels to the goal state s_{goal}. The route is then a sequence:

$$R = \{(s_i, s_{i+1}), a\}_{i=\text{start}}^{\text{goal}} \mid (s_i, s_{i+1}) \in E, \exists a : T(s_{i+1} \mid s_i, a) > 0 \tag{3.18}$$

This yields the expected time of the route to be defined as:

$$E(t_R) = \sum_{\{(s_i, s_{i+1}), a\} \in R} T(s_{i+1} \mid s_i, a) \cdot t_{s_i, s_{i+1}} \tag{3.19}$$

where $T(s_{i+1} \mid s_i, a)$ is the probability of moving from state s_i to s_{i+1}, carrying out action a and $t_{s_i, s_{i+1}}$ is the time it takes an agent to travel between the defined states.

We seek such route R^* that minimizes the expected time defined in Eq. (3.19):

$$R^* = \underset{R}{\operatorname{argmin}}\, E(t_R) \tag{3.20}$$

Defining the reward function as a decreasing function in time, allows us to consider a dual problem of finding a route that maximizes the expected reward:

$$R^* = \underset{R}{\operatorname{argmax}}\, E(r_R) \tag{3.21}$$

This is a problem that the MDPs are able to solve. We present the reward function as a dual representation of agent's travel time and argue on the differences and similarities between the optimal path provided by MDPs and R^* as defined in Eq. (3.20). The function $R(s, s', a)$, defines the reward for reaching state s from state s' carrying out action a.

We assume, that the agent needs time for driving from one state s to another s'. The time $t^{s,s'}_{\text{drive}}$ it takes the agent to drive from any arbitrary state s to an adjacent one s' depends on the distance between these states and on the agent's driving speed v_{drive} and can be defined as:

$$\forall s, s' \in V, (s, s') \in E: \quad t^{s,s'}_{\text{drive}} = \frac{\sqrt{(s - s')(s - s')^T}}{v_{\text{drive}}} \tag{3.22}$$

The cost $R(s, s', a)$ of moving from state s' to state s carrying out action a grows with the time $t^{s,s'}_{\text{drive}}$ the agent needs to cover the path, that is: the longer it takes for an agent to take an action, the bigger cost he has to pay for it. We therefore define a negative reward for each action in A_{move} between any two adjacent states as follows:

$$\forall s, s' \in V, \forall (s, s') \in E, \forall a \in A_{\text{move}}: \quad R(s, s', a) = -t^{s,s'}_{\text{drive}} \tag{3.23}$$

Failure Costs

In addition to these costs, we also penalize staying in one place. In our system the agent stays in the same state only if he fails to carry out an optimal action. We define this cost in such way, that the agent looses a constant amount of time Δt if he fails to carry out any action $a \in A_{\text{move}}$. However, the cost for failing to carry out the action a_{park} is different. This cost can be used to control the agent's behavioral model. The higher the cost of the failure is, the less aggressive the agent's behavior will be. Setting the failure cost to a high value results in the agent's behavior that avoids carrying out the parking action in a parking lot with a high occupancy probability. We present examples of the impact of this cost in the experimental section.

Success Rewards

We also define the rewards that the agent receives when he successfully carries out parking actions. These rewards are likewise time based. They form a decreasing linear function in order to guarantee that the state closer to the destination gets a bigger reward than the one situated further. We define the reward to be $r_{\max} - r_s$, where $r_s = t_{\text{walk}}^{s,s_{\text{goal}}}$ is the time that the agent needs to walk from an arbitrary state s to the goal, $r_{\max} = \max_{s \in V} r_s$ is the greatest of those times. Formally:

$$\forall s \in V, (s, s_{\text{goal}}) \in E : R(s, s_{\text{goal}}, a_{\text{park}}) = r_{\max} - r_s \tag{3.24}$$

To express r_s, we define $t_{\text{walk}}^{s,s_{\text{goal}}}$ similarly to Eq. (3.22). It depends on the distance between the states s and s_{goal} as well as on the walking speed of the agent:

$$t_{\text{walk}}^{s,s_{\text{goal}}} = \frac{\sqrt{(s - s_{\text{goal}})(s - s_{\text{goal}})^T}}{v_{\text{walk}}} \tag{3.25}$$

The presented reward function $r_{\max} - r_s$ models the win in time between parking in the worst parking lot and the one represented by state s.

3.4.3 Solution to the MDP

The general reward function is fully defined by Eq. (3.23) and Eq. (3.24). It guarantees, that the agent tries to find a trade-off between shorter driving and shorter walking paths, weighted by the occupancy probability of the parking lots.

The key building block of the MDPs is the utility function, which is a measure of how good the state is in the long run:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} R(s_t) \,\middle|\, \pi, s_0 = s\right] \tag{3.26}$$

Given the utility function we define the policy as an expected sum of the rewards obtained, where the expectation is taken over all possible state sequences that could occur, given that the policy is executed. The optimal policy π^* then satisfies:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \left[U^{\pi}(s) \mid \pi, s \in \pi\right] \tag{3.27}$$

This equation is, provided the definition of the cost function, effectively equivalent to Eq. (3.20) leading to π^* being an optimal decision in the means of minimizing the expected time that the agent needs to find a parking lot.

However, as time goes to infinity the sum in Eq. (3.26) also tends to lean to infinitely big values. In order to avoid this, we make use of discounted rewards:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma R(s_t) \mid \pi, s_0 = s\right] \tag{3.28}$$

where $\gamma \in [0, 1)$ is the discount factor. Formulating the utility function in such way allows to avoid infinitely big sums [31] and allows for searching for the optimal policy in the way, that we present further in this section. In order to minimize the deviation of the MDP solution from the one that minimizes the expected time spent on finding a free parking lot, we set γ to a value close to 1. This, however, results in longer planning times. To deal with this issue, we utilize the iterative Policy Iteration algorithm that terminates when there is no update to the policy, omitting estimating exact utility values of all states. This algorithm alternates the following two steps, beginning with some initial policy π_0,

- *Policy evaluation:* given a policy π_i, calculate $U_i = U^{\pi_i}$, the utility of each state if π_i, were to be executed.
- *Policy improvement:* Calculate a new Maximum Expected Utility (MEU) policy π_{i+1}, using one-step look-ahead based on U_i (as in Eq. (3.30)).

The *policy evaluation* is performed via solving a linear system of simplified Bellman equations of the form:

$$U(s) = R(s) + \gamma \sum_{s'} T(s \mid s', a) U(s') \tag{3.29}$$

In order to carry out the *policy improvement* step, we choose the optimal action using the principle of Maximum Expected Utility (MEU) in each state, that is, choose: the action that maximizes the expected utility of the subsequent state. The optimal policy $\pi^*(s)$ can be therefore seen as the one that maximizes the expected utility if followed:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s \mid s', a) U(s') \tag{3.30}$$

The algorithm terminates when the policy improvement step yields no change in the utilities. The given method can be shown to converge in $O(n^3)$ where n is the number of vertices in the graph.

This solution to the MDP guarantees to find a route that maximizes the discounted rewards while searching for the free parking lot and walking from the found one to the goal. Under the assumption that the discount factor leans to 1 this route is also the one that minimizes the expected time for searching for the parking lot.

3.4.4 Re-planning

The agent carries out an optimal strategy to the point when he decides to park. Let us imagine for a moment that the parking lot at which the agent wants to park is occupied. From the point of view of the MDPs the optimal decision would be to wait and try once again, which is clearly a suboptimal decision. Therefore, we also utilize the observations of the world.

Whenever we move past a parking lot, we check if it is occupied and update the occupancy probability in the related state to 1 or 0 respectively. This allows for carrying out further decisions based on the better background, having more knowledge about the state of the world.

The update of the occupancy of the state s consequently changes the probability of the action a_{park} that can be carried out in this state, updating the transition function $T(s_{\text{goal}} \mid s, a_{\text{park}})$ to either 1 or 0. We carry out the policy iteration algorithm again to define the new optimal actions for each state taking into account the changes in the transition model. We keep re-planning until we find a free parking lot or prove that there is none.

4 Experimental Results

Through the course of this work, we make extensive use of three core components: visual car detection, parking lot modeling and planning. In this section, we present the experimental evaluation of each of these parts.

4.1 Visual Car Detection

We start the visual detection section by describing the dataset that we utilize to train the car detector. A training dataset plays a crucial part in any detection algorithm. In order to train the visual car detector we assemble a training dataset that consists of a number of positive (containing cars) and negative (images not containing cars) examples.

For the reason that the car's view depends on the angle of view, we assemble positive training data for different angles of the cars. We prepare training samples for four distinct views of the cars — front, back and both left and right sides.

The sides of the car are invariant under the mirror transformation. This allows to use the same image for training a classifier for each side which simplifies the training data generation. Front and back of the car, though different from human point of view, seem to be similar in the HOG visualization. Therefore, the corresponding datasets can be

Figure 4.1: Examples of positive and negative training examples for front/rear car detector.

Figure 4.2: Depth acquired from disparity between stereo images.

combined into one.

These datasets contain respectively images of sizes 128×128 pixels for front/rear car view and 128×64 pixels for side view. We utilize 682 positive images that contain cars for training the front/rear detector. In addition to these positive examples we also gather 7972 negative images. Some examples from the front/rear positive and negative training datasets are presented in Figure 4.1.

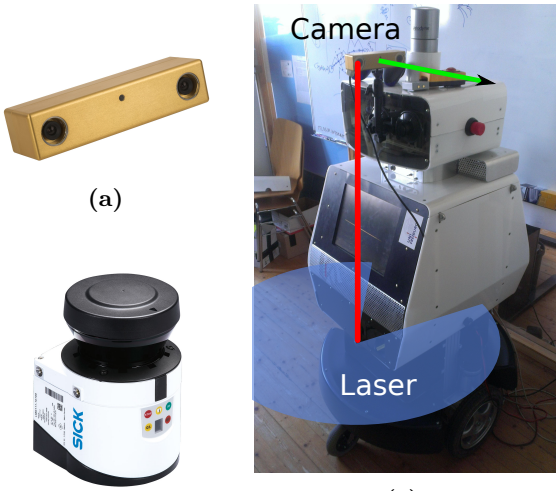
The positive examples contain exclusively views of the cars. Each positive example shows one and only one car. The car occupies the full area of the image. The photos of the cars depict them centered and seen from the same angle of view. This is crucial for training a meaningful classifier.

The positive dataset is assembled from different sources: INRIA Car Dataset by Carbonetto and Dorkó [7], Motorway Car Dataset by Caraffi et al. [6], UIUC Image Database for Car Detection by Agarwal et al. [2], KITTI car dataset by Geiger et al. [14] and various car images found in a public domain via a web searching engine.

In order to create the negative set we randomly sample different sized patches of the given aspect-ratio (128×128 and 128×64 respectively) from the images that contain no cars. Following the work of Dalal and Triggs [11] we cut false detections that appear in the first runs of training the classifier with a clipping tool [33] and add them to the negative training dataset.

We use the trained classifier on the images from the camera mounted inside of the car facing sidewards or, possibly, onto any other moving platform. During our experiments we use a Bumblebee stereo-camera (Figure 4.3a) pointed to the side of the Europa robot Obelix (Figure 4.3c).

We are interested in detections accumulated over time. The same car is seen from different angles, throughout different (usually consequent) images. We consider the car to be detected if it has been detected at least once in the sequence of images, featuring this



(a) (b) (c)

Figure 4.3: A Bumblebee stereo camera (a) and a laser range finder (b) mounted on the Obelix robot on the same vertical axis (c).

particular car. With this in mind, the detection rate for our realization of the approach is approximately 95%.

The examples of detected cars can be seen in Figure 4.4.

4.2 Parking Lot Modeling

The occupancy data is based on the 2D data obtained either from the stereo cameras (Figure 4.3a) or from laser range finder, mounted on the robot (Figure 4.3b). As can be seen in the figure, the stereo camera is mounted on the same z-axis with the laser range finder. We search for the endpoints that represent the cars as described in the section Visual Car Detection of chapter In-vehicle Parking Space Occupancy Estimation and Guidance System.

Figure 4.2 shows that the depth information computed from the disparity image between two stereo images lacks precision and has a significant amount of noise. Not only this noise is caused by the method of depth estimation and errors in point association, but it also suffers from the windows present in almost any car. The windows are transparent which results in wrong depth measurements present in the detected regions of interest.

To avoid the high noise from the stereo camera, we use the depth measurements from a laser range finder. Each laser measurement has high precision in measuring the distance.

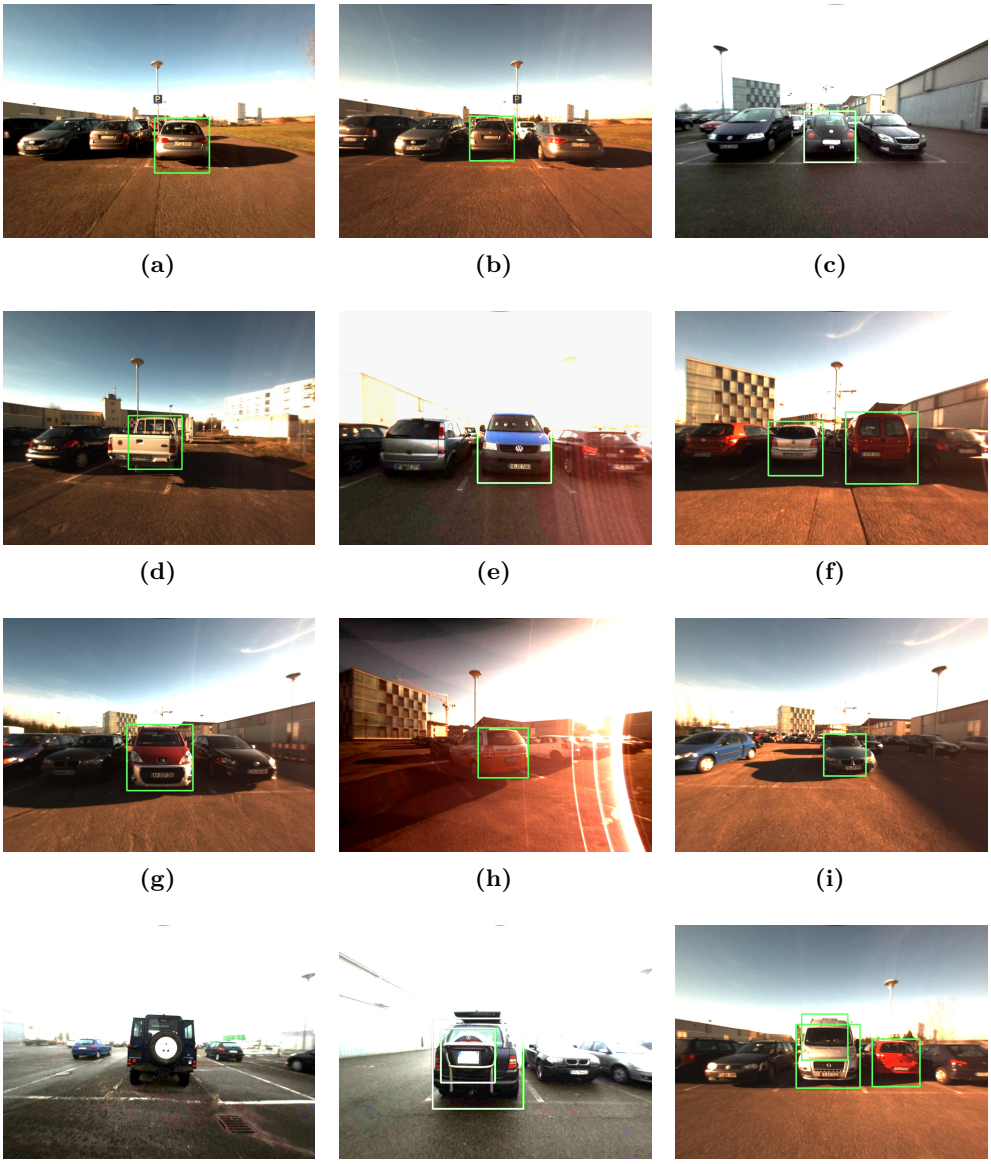
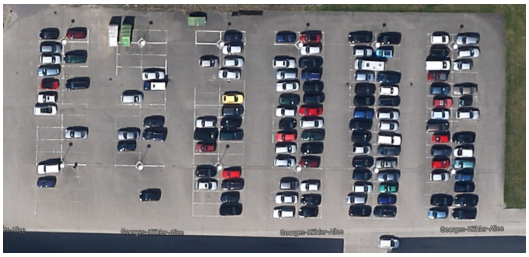
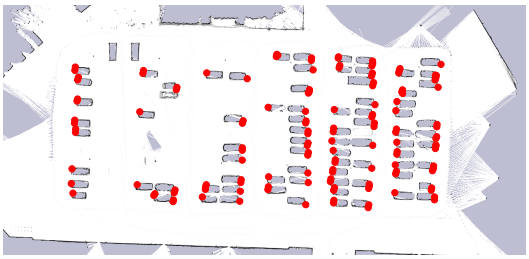


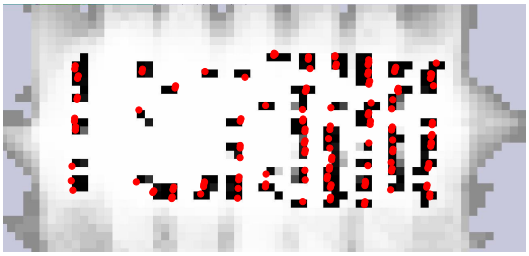
Figure 4.4: Examples of detections of different cars under different light conditions. The cars vary in shape (d), (c), (e), (i) in color and lightning (h), (g), (c). The cars on the sides of the images are often not detected. This, however, does not cause errors given the fact that data acquisition is carried out in continuous manner, and the missed cars are detected in the frames taken at adjacent times (a), (b). The bottom row presents some detection failures. The main cause is the shape of the car that significantly differs from an average one.



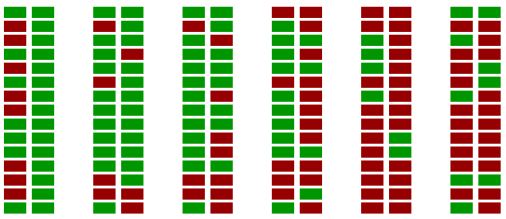
(a) Aerial view.



(b) Car detections over the map built with the laser-based SLAM procedure.



(c) Occupancy grid map acquired as described in Section 3.3.1.



(d) Cars positions mapped to pre-defined parking lot positions.

Figure 4.5: Representations of the campus parking lot at the University of Freiburg. Images (b), (c), (d) represent the same occupancy state of the parking lot.

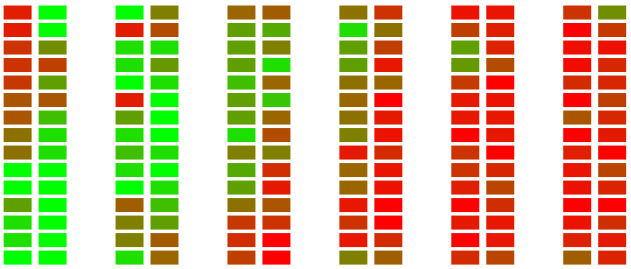


Figure 4.6: Occupancy probability accumulated during a sequence of observations shown in color. The greener the parking lot is — the bigger the probability for it to be free.

Because of its mounting position on the human knee level, it also does not suffer from the partial transparency of the cars as individual laser beams never encounter the cars' windows.

We present occupancy data aggregated to the occupancy grids as well as into pre-defined parking lot positions. The illustration of differences in the maps representations can be seen in Figure 4.5. The figure shows car detected cars positions as red dots. We produce these detections fusing the laser range finder measurements with the visual detections as described in Chapter 3.2.3.

There are two distinct ways in which we model the parking lots: occupancy grid maps and pre-defined parking lots positions.

As can be seen from the Figure 4.5c the occupancy grid based representation lacks precision. The main problem are the discretization errors. It is a common situation when the detections on different sides of the same car fall into different grid cells, causing erroneous map representation. Another problem of the occupancy grid based approach is an imperfect approximation of the car orientation. We argue, that using the observation model, as seen in Figure 3.5a, yields the orientation of the detected car to be influenced by the angle from which it is observed.

Because of these issues, we mostly rely on the pre-defined positions of the parking lots for occupancy estimation. An example of the occupancy information from the data gathered during one run is presented in Figure 4.5d. The exact positions of the parking lots and deterministic association of detections to them allows us to quantify the parking lot positions estimation precision. For this system we measure the detection rate throughout the course of 21 data acquisition sessions. A session is a sequence of

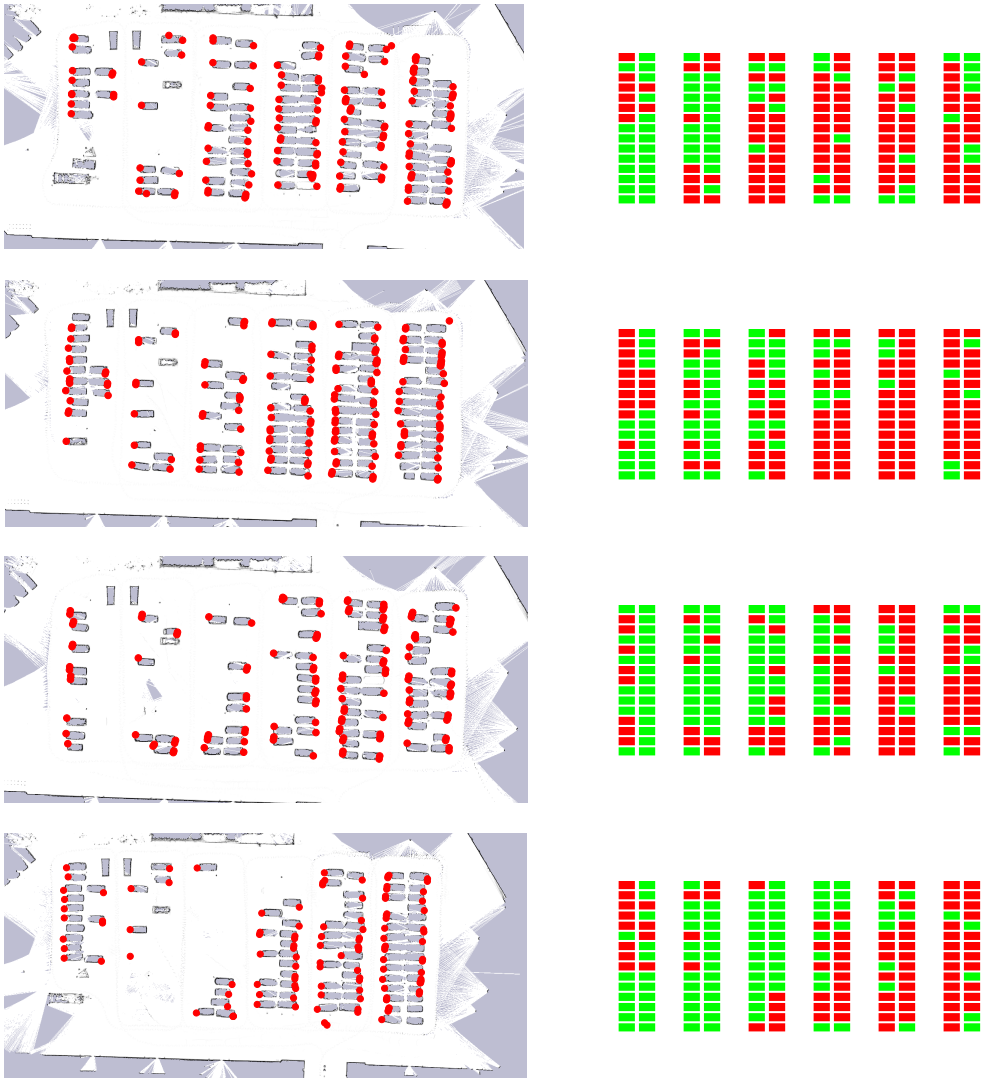


Figure 4.7: This figure shows the precision of occupancy modeling. Each pair of images represents the occupancy state of the parking lot on a specific day. The images on the left show a map of the environment build by laser range measurements. The detected cars are represented in red dots. The right images show the occupancy of each parking space, where green stands for "free" and red for "occupied". Part 1.

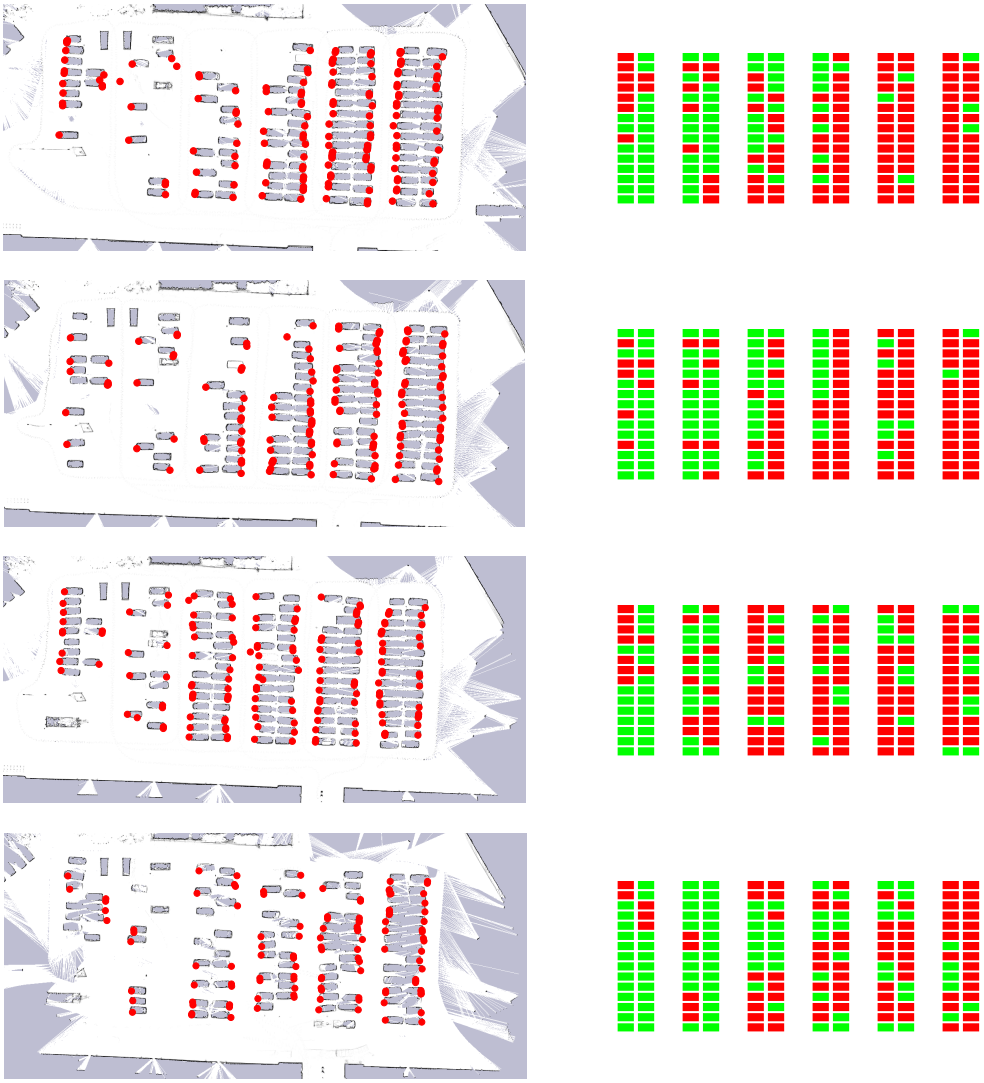


Figure 4.8: This figure shows the precision of occupancy modeling. Each pair of images represents the occupancy state of the parking lot on a specific day. The images on the left show a map of the environment build by laser range measurements. The detected cars are represented in red dots. The right images show the occupancy of each parking space, where green stands for “free” and red for “occupied”. Part 2.

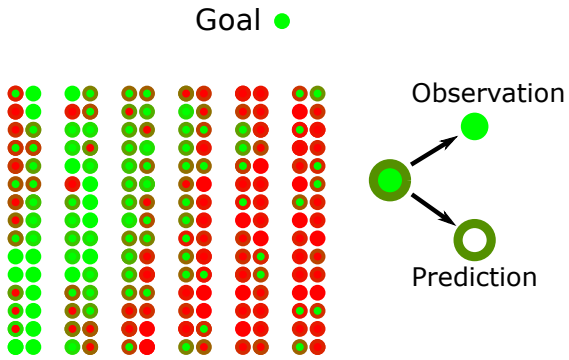


Figure 4.9: A visual representation of the parking lot graph useful for planner visualization. The edges are defined similarly to 3.6. Each state contains prediction and observation information. This can be seen as combining occupancy information in Figure 4.5d with Figure 4.6.

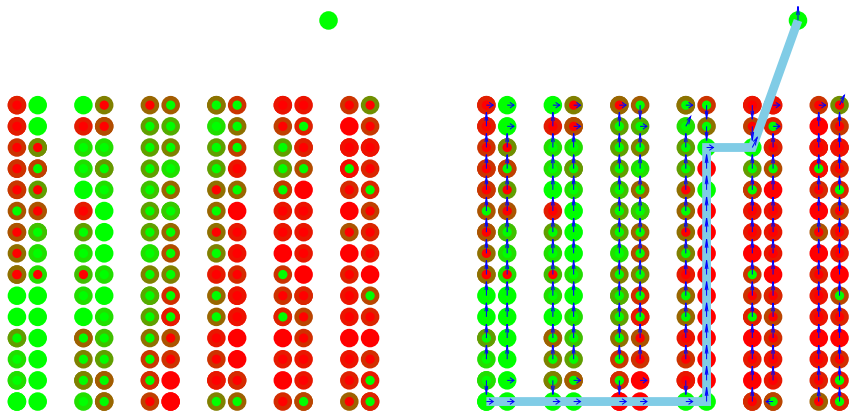
measurements that provide the occupancy information of every parking lot in the area of interest. Figure 4.7 and Figure 4.8 each show 4 examples of such sessions. We estimate the precision of parking lot occupancy detection as a relation of the number of correctly labeled parking lots to the total number of parking lots. This results in: $\frac{\text{correct}}{\text{total}} \approx 90\%$ of all parking lots to be labeled correctly as occupied or free respectively. The exact precision, if calculated on the subset of sessions presented in Figure 4.7 and Figure 4.8, equals 90.83% as a result of 132 wrongly labeled parking lots out of the total number of 1440.

There are two main sources of errors. The first one is failing to detect the cars or assigning them to the wrong parking lot. Failing to detect a car logically propagates from the visual detection part. The data association error is usually due to abnormal sizes of the cars or improper position in the parking lot. The second source of error is an assumption that the world stays static while the dataset is being acquired.

4.3 Action Planning

In this section we present the evaluation of the work of the MDP based planner, that searches for a free parking space. As a testing environment we use the parking lot at the campus of the University of Freiburg.

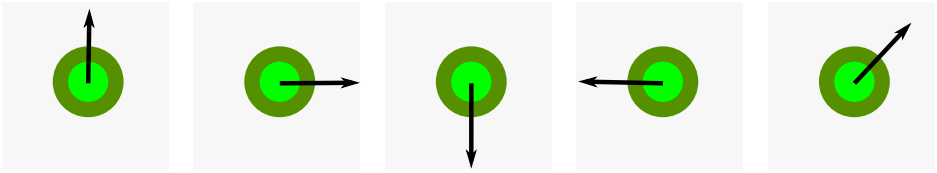
We make use of the occupancy information gathered from integrating the measurements of the parking lots' occupancy (see Figure 4.5d) as observed during a period of 21 days. In order to be able to show the results of the planner we introduce a number of visualizations.



(a) (b)

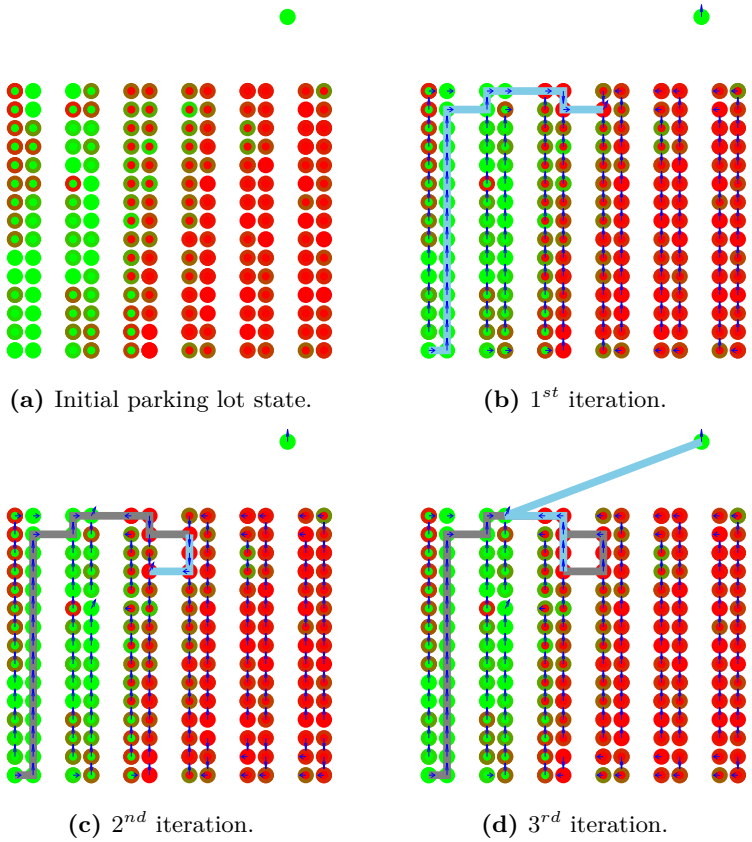
Figure 4.10: An optimal strategy for $v_{\text{walk}} = 4$ km/h, $v_{\text{drive}} = 10$ km/h and the failure cost of 10 seconds. Each state is presented by an inner and an outer circles. The inner circle depicts the real occupancy status on the day the experiment was held. The outer one presents the occupancy estimate based on the data accumulated from multiple runs. Both circle's colors vary from red for occupied parking lots to green for free ones.

We visualize every parking space as a state in which the agent can perform an action to park as presented in Figure 4.9, where each state shows both — the prediction of the occupancy probability as an outer circle and the actual free or occupied state as the inner one. This representation is useful to see in a glance what the state of the parking was on the specific date, while being able to analyze the reasoning behind the agent's actions. However, only the information shown as an outer circle — the occupancy prediction is available to the robot on the planning step. The occupancy prediction is only then



(a) Drive up. **(b)** Drive right. **(c)** Drive down. **(d)** Drive left. **(e)** Try to park.

Figure 4.11: Visualization of the actions considered optimal by a policy on the graph.



(a) Initial parking lot state. **(b)** 1^{st} iteration.

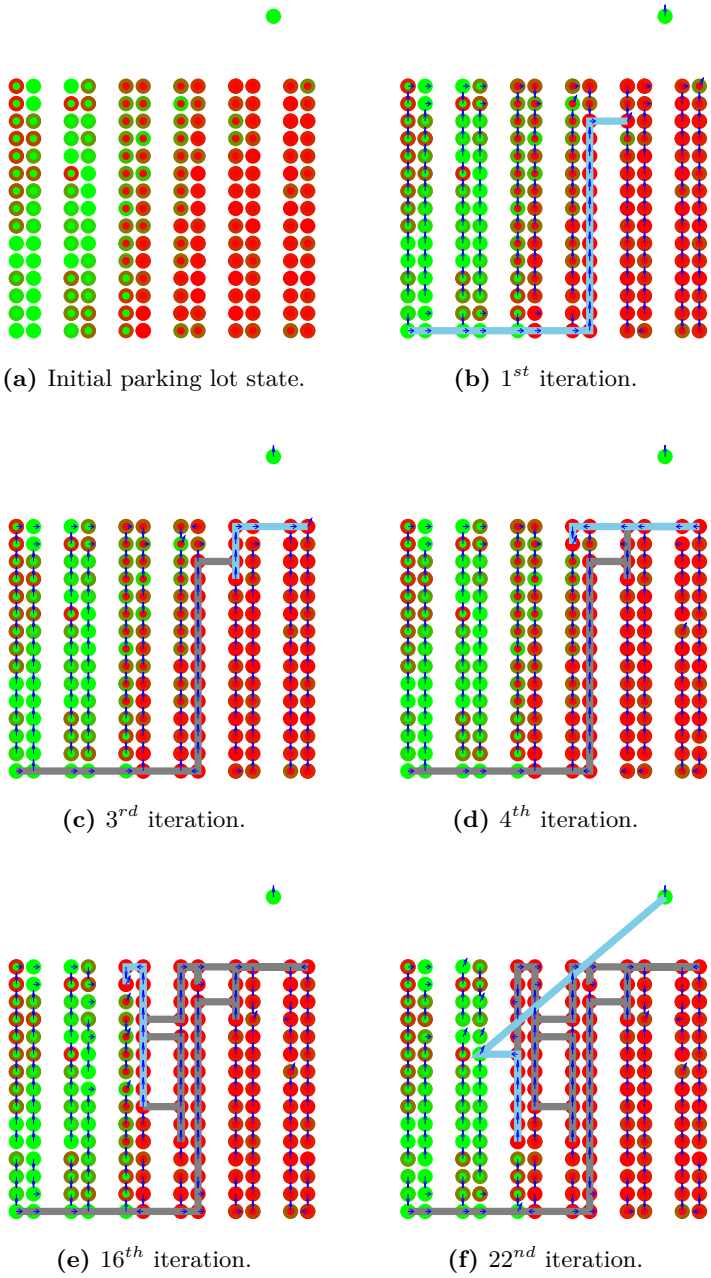
(c) 2^{nd} iteration. **(d)** 3^{rd} iteration.

Figure 4.12: An optimal strategy for $v_{\text{walk}} = 4$ km/h, $v_{\text{drive}} = 10$ km/h and the failure cost of 90 seconds carried out on an artificial dataset. The optimal actions are different for each iteration and are depicted as presented in Figure 4.11.

updated to the real occupancy status, when the parking lot is observed by the agent as he drives by. The actions, that the agent is able to carry out, as presented in Eq. (3.15) are visualized in the way shown in Figure 4.11.

As described in Section 3.4 there are 3 core parameters that influence the performance of the planner: v_{walk}, v_{drive} and the failure cost. In this section we present the differences in the behavior of the agent with respect to different values of these parameters. All the experiments share the same actual positions of the parked cars for the ease of comparison.

For these parameters we consider a number of sets of their values. We further present the differences in the approach picked by the agent depending on the different sets of the parameters.

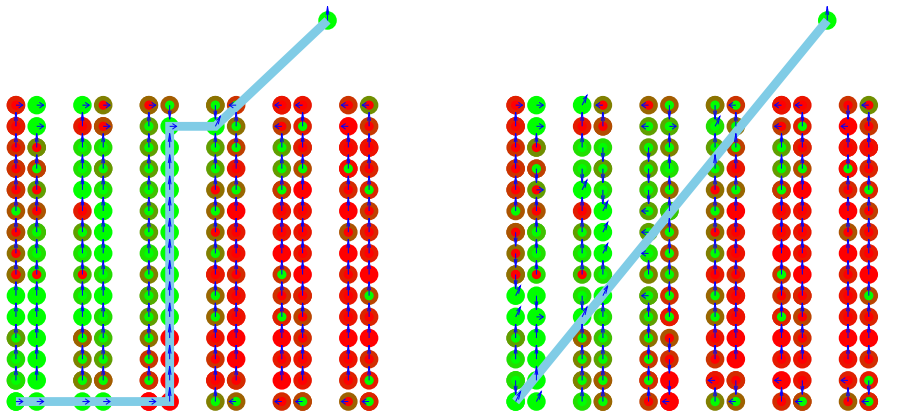


(a) Initial parking lot state. **(b)** 1^{st} iteration.

(c) 3^{rd} iteration. **(d)** 4^{th} iteration.

(e) 16^{th} iteration. **(f)** 22^{nd} iteration.

Figure 4.13: An optimal strategy for $v_{\text{walk}} = 4$ km/h, $v_{\text{drive}} = 10$ km/h and the failure cost of 10 seconds carried out on an artificial dataset.



(a) Optimal strategy for $v_{\text{walk}} = 4$ km/h, $v_{\text{drive}} = 10$ km/h and the failure cost of 90 seconds tested on the real data.

(b) Optimal strategy for $v_{\text{walk}} = 10$ km/h, $v_{\text{drive}} = 10$ km/h and the failure cost of 90 seconds tested on the real data.

Figure 4.14

Figure 4.10 shows the path considered optimal by the agent, given the driving speed of 10 km/h and walking speed of 4 km/h. Additionally, a failure cost of 10 seconds ensures, that the agent considers that it loses 10 seconds for every unsuccessful parking action. This results in the strategy that finds a trade-off between the distance from the parking lots to the goal and their occupancy. The figure shows that the agent decides for a state that combines close proximity to the goal state with locally high probability of being free.

Looking at Figure 4.13 that shows the performance of the search with the same core parameters on the artificial dataset we see, that in a rare case of extremely high abnormal occupancy in the area, the agent is too confident that the parking action would succeed and it results in extremely long paths and numerous re-planning steps. The agent tends to pick the places, situated closer to the goal. To deal with this issue we set the failure cost to a high value, which results in penalizing the choice of the parking lots in which the agent is likely to fail the parking action. Setting the failure cost to 90 seconds results in a much more conservative behavior as presented in Figure 4.12.

Such conservative strategy also proves to be efficient in the real world scenarios as can be seen in Figure 4.14a, showing only a rare need for re-planning. However, this strategy keeps the agent from making overly optimistic decisions and he hardly ever observes the most occupied parts of the parking lot.

Another interesting example that is worth mentioning is the test where human speed is set to a high value. In the Figure 4.14b we observe, that the agent makes the decision that the most practical strategy is to park in the closest to the starting state parking space and walk to the goal from there. This is a result of setting v_{walk} to be equal to v_{drive}. It may be explained as follows: if the walking speed is equal to the driving speed it makes no sense driving around the parking lot when it is possible to walk straight to the goal.

5 Conclusion and Further Work

The task of this thesis was to develop a proof of concept framework for supporting drivers in finding free parking spaces. The system should solve this task by perceiving cars in the environment, modeling parking places, and suggesting trajectories towards free parking lots.

As the result of this task we present a proof of concept of an integrated system, that only relies on on-board perception, models the parking situation and generates optimal trajectories for finding a free parking space and reaching the target location as fast as possible. Our system automatically quantifies the occupancy probability of each parking lot in an area of interest and provides a behavior that minimizes the expected time for reaching the target destination.

Our approach requires only on-board sensing to perceive the environment. So far, we use a camera and a laser range finder to observe the scene and to detect cars from the perceptual input. We solve this subtask by a supervised learning approach using a support vector machine and histograms of oriented gradients (HOG) as features. The detections are fused into a model of the environment that allows for building up a representation of the scene depicting the occupancy probability of each parking lot. Based on this representation, we define a Markov Decision Process that allows us to generate the optimal path, given the current knowledge, to find a parking lot and at the same time park as close as possible to the desired target location.

We present a series of experiments carried out with the Obelix robot at the campus of the University of Freiburg. We evaluate the HOG based visual detection of the cars, the assignment of the detections to the parking lots, and the trajectory planning using MDPs. The visual detection fused with the laser measurements shows steady car detection rates, where approximately 90% of the parking lots are labeled correctly. The MDP based planner was evaluated on the real occupancy data acquired from the parking lot at the University Freiburg campus. It is shown to find the optimal route to the goal, while adopting to current system parameters, such as walking and driving speeds.

Although the system solves the given task of this thesis, there is space for the future work. The visual detection rates can be improved by adjusting the parameters of the

HOG descriptors and adjusting the training datasets. This, however, requires extensive additional study.

Whereas the perception and modeling part was written in C++ and runs on the robot, the planner is a stand-alone application, written in Octave. Thus, for an integrated system, all components need to be implemented on the same platform. Though it was not an explicit task of this thesis, it may be subject to the future work to integrate it with the robot's planner system.

The improvements can also be made to the parking lots modeling. Our current approach relies on the pre-defined parking lot positions as this gave the best performance in our settings. It is tempting to improve the performance of the occupancy grid maps in order to map the parking lots in an autonomous fashion based on the observations of the parked cars at different times. In our work we rely on a simultaneous localization and mapping approach in order to precisely estimate the position of the robot while taking measurements. This part is crucial for data association. It would, however, be an interesting improvement to find a way of estimating the parking lots' occupancy information relying only on GPS measurements.

Overall, the system, presented in this thesis is a proof of concept implementation. The described system, however, has the potential to be used in autonomous vehicles as it provides a way to deal with searching for a parking lot in a fully autonomous fashion.

Bibliography

[1] F. Abad, R. Bendahan, S. Wybo, S. Bougnoux, C. Vestri, and T. Kakinami. Parking space detection. 2013. URL http://vestri.free.fr/data/ITS%202007.pdf.

[2] S. Agarwal, A. Awan, and D. Roth. The uiuc image database for car detection, 2002. URL http://cogcomp.cs.illinois.edu/Data/Car/.

[3] R. Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518.

[4] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965. ISSN 0018-8670. doi: 10.1147/sj.41.0025.

[5] W. Burgard, D. Fox, and S. Thrun. Probabilistic state estimation techniques for autonomous and decision support systems. *Informatik-Spektrum*, 34(5):455–461, 2011. ISSN 0170-6012. doi: 10.1007/s00287-011-0561-8. URL http://dx.doi.org/10.1007/s00287-011-0561-8.

[6] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas. A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera. In *ITS Conference*, pages 975–982, Sep. 2012.

[7] P. Carbonetto and G. Dorkó. Inria car dataset, 2004. URL http://www.cs.ubc.ca/~pcarbo/objrecls/data/cars.tar.gz.

[8] C. Chih-Chung and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[9] V. Coric and M. Gruteser. Crowdsensing maps of on-street parking spaces. In *Proceedings of the 2013 IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS '13, pages 115–122, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-0-7695-5041-1. doi: 10.1109/DCOSS.2013.15. URL http://dx.doi.org/10.1109/DCOSS.2013.15.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL `http://dx.doi.org/10.1023/A%3A1022627411411`.

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,*, volume 1, pages 886–893 vol. 1, June 2005. doi: 10.1109/CVPR.2005.177.

[12] K. Eisele. Driverless long-distance car journey., 2014. URL `http://technicity.daimler.com/en/driverless-long-distance-car-journey/?csref=_sm:in_140319-technici_gog_kw12_pc`.

[13] K. Fintzel, R. Bendahan, C. Vestri, S. Bougnoux, and T. Kakinami. 3d parking assistant system. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 881–886, June 2004. doi: 10.1109/IVS.2004.1336501.

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

[15] C. C. Huang and S. J. Wang. A hierarchical bayesian generation framework for vacant parking space detection. *IEEE Transactions on Circuits and Systems for Video Technology.*, 20(12):1770–1785, Dec 2010. ISSN 1051-8215. doi: 10.1109/TCSVT.2010.2087510.

[16] C.C. Huang, S.J. Wang, Y.J. Chang, and T. Chen. A bayesian hierarchical detection framework for parking space detection. In *2008. ICASSP 2008. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2097–2100, March 2008. doi: 10.1109/ICASSP.2008.4518055.

[17] H. Ichihashi, T. Katada, M. Fujiyoshi, A. Notsu, and K. Honda. Improvement in the performance of camera based vehicle detector for parking lot. In *IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1–7, July 2010. doi: 10.1109/FUZZY.2010.5584554.

[18] K. Konolige. Small vision systems: Hardware and implementation. In Yoshiaki Shirai and Shigeo Hirose, editors, *Robotics Research*, pages 203–212. Springer London, 1998. ISBN 978-1-4471-1582-3. doi: 10.1007/978-1-4471-1580-9_19. URL `http://dx.doi.org/10.1007/978-1-4471-1580-9_19`.

[19] H. Kretzschmar, G. Grisetti, and C. Stachniss. Lifelong map learning for graph-based SLAM in static environments. *KI – Künstliche Intelligenz*, 24:199–206, 2010. doi: 10.1007/s13218-010-0034-2.

[20] H. Kretzschmar, C. Stachniss, and G. Grisetti. Pose graph compression for laser-based SLAM. In *Proc. of the Int. Symposium of Robotics Research (ISRR)*, Flagstaff, AZ, USA, 2011. URL `http://www.informatik.uni-freiburg.de/~stachnis/pdf/stachniss11isrr.pdf`. Invited presentation.

[21] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard. Large scale graph-based SLAM using aerial images as prior information. *Journal of Autonomous Robots*, 30(1):25–39, 2011. doi: 10.1007/s10514-010-9204-1.

[22] C. K. Lee, C. L. Lin, and B. M. Shiu. Autonomous vehicle parking using hybrid artificial intelligent approach. *Journal of Intelligent and Robotic Systems*, 56(3): 319–343, 2009. ISSN 0921-0296. doi: 10.1007/s10846-009-9319-9. URL `http://dx.doi.org/10.1007/s10846-009-9319-9`.

[23] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception-driven autonomous urban vehicle. In Martin Buehler, Karl Iagnemma, and Sanjiv Singh, editors, *The DARPA Urban Challenge*, volume 56 of *Springer Tracts in Advanced Robotics*, pages 163–230. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03990-4. doi: 10.1007/978-3-642-03991-1_5. URL `http://dx.doi.org/10.1007/978-3-642-03991-1_5`.

[24] D. Lima and G. Pereira. Navigation of an autonomous car using vector fields and the dynamic window approach. *Journal of Control, Automation and Electrical Systems*, 24(1-2):106–116, 2013. ISSN 2195-3880. doi: 10.1007/s40313-013-0006-5. URL `http://dx.doi.org/10.1007/s40313-013-0006-5`.

[25] J. Markoff. Google cars drive themselves, in traffic. *The New York Times*, 10:A1, 2010.

[26] J. Maye, R. Triebel, L. Spinello, and R. Siegwart. Bayesian online learning of driving behaviors. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[27] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation.*, volume 2, pages 116–121, Mar 1985. doi: 10.1109/ROBOT.1985.1087316.

[28] G. Pierce and D. Shoup. Sfpark: Pricing parking by demand., 2013. URL `http://escholarship.org/uc/item/0j41t7rz`.

[29] G. Pierce and D. Shoup. Getting the prices right. *Journal of the American Planning Association*, 79(1):67–81, 2013. doi: 10.1080/01944363.2013.787307. URL `http://www.tandfonline.com/doi/abs/10.1080/01944363.2013.787307`.

[30] J. P. Rodrigue, C. Comtois, and B. Slack. *The geography of transport systems.* Routledge, 2013. URL `http://people.hofstra.edu/geotrans/eng/ch6en/conc6en/ch6c4en.html`.

[31] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 edition, 2003. ISBN 0137903952.

[32] M. R. Schmid, S. Ates, J. Dickmann, F. von Hundelshausen, and H.-J. Wuensche. Parking space detection with hierarchical dynamic occupancy grids. *IEEE Intelligent Vehicles Symposium*, 2012.

[33] N. Seo. Imageclipper - a tool for fast image cropping, 2008. URL `https://code.google.com/p/imageclipper/`.

[34] D. C. Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006.

[35] L. Spinello, R. Triebel, and R. Siegwart. Multiclass multimodal detection and tracking in urban environments. *Int. Journal on Robotics Research (IJRR)*, 2010.

[36] J. K. Suhr and H. G. Jung. Fully-automatic recognition of various parking slot markings in around view monitor (avm) image sequences. In *15th International IEEE Conference on Intelligent Transportation Systems (ITSC),*, pages 1294–1299, Sept 2012. doi: 10.1109/ITSC.2012.6338615.

[37] J. K. Suhr, H. G. Jung, K. Bae, and J. Kim. Automatic free parking space detection by using motion stereo-based 3d reconstruction. *Machine Vision and Applications*, 21(2):163–176, 2010. ISSN 0932-8092. doi: 10.1007/s00138-008-0156-9. URL `http://dx.doi.org/10.1007/s00138-008-0156-9`.

[38] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the darpa grand challenge. In M. Buehler, K. Iagnemma, and S. Singh, editors, *The 2005 DARPA Grand Challenge*, volume 36 of *Springer Tracts in Advanced Robotics*, pages 1–43. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73428-4. doi: 10.1007/978-3-540-73429-1_1. URL `http://dx.doi.org/10.1007/978-3-540-73429-1_1`.

[39] G. D. Tipaldi and K. O. Arras. I want my coffee hot! learning to find people under spatio-temporal constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011. URL `http://www.informatik.uni-freiburg.de/~tipaldi/papers/tipaldiICRA11.pdf`.

[40] J. Trefný and J Matas. Extended set of local binary patterns for rapid object detection. *Computer Vision Winter Workshop*, 2010.

[41] N. True. Vacant parking space detection in static images. 2007.

[42] M. Tschentscher and M. Neuhause. Video-based parking space detection. 2012.

[43] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518 vol.1, 2001. doi: 10.1109/CVPR.2001.990517.

[44] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. In *ICCV*, 2013.

[45] Z.J. Wang, J.W. Zhang, Y.L. Huang, H. Zhang, and A. Mehr. Application of fuzzy logic for autonomous bay parking of automobiles. *International Journal of Automation and Computing*, 8(4):445–451, 2011. ISSN 1476-8186. doi: 10.1007/s11633-011-0602-4. URL `http://dx.doi.org/10.1007/s11633-011-0602-4`.

[46] Q. Wu and Y. Zhang. Parking lots space detection. 2006. URL `http://pdf.aminer.org/000/346/665/event_classification_for_automatic_visual_based_surveillance_of_parking_lots.pdf`.

[47] R. Yusnita, F. Norbaya, and N. Basharuddin. Intelligent parking space detection system based on image processing. *International Journal of Innovation, Management and Technology*, 3rd, 2012.