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
???????????WHOISHERE??????????????

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

Abstract

As there are more and more cars on the streets from day to day it gets more and more complicated to find a parking space, especially in the center of the city, where one has spacial constraints on building new parking lots.

Therefore it comes in handy to map and predict the occupancy of parking spaces. This information, when integrated into planners commonly found in cell phones and GPS navigation systems, can drastically improve the experience of searching for a parking lot.

In this thesis we present an automated approach of 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 also taking into account parking lot occupancy information.

The proposed approach uses an in-vehicle camera setup to repeatedly detect and map the parked cars in the area of interest and to estimate the occupancy probability of each found position throughout all runs. The framework is easily expendable to account for searching for occupancy on specific date or even time of day.

Furthermore, we also introduce the planner that relies on the occupancy probability of each found parking lot to find the optimal path in the means of time spent on searching for a parking place as well as getting from the found parking lot to the destination by foot.

Contents

1 Introduction

We all have been in a situation, when one was driving around a city, trying to find a free parking space. One can drive numerous circles around the center of the city with absolute uncertainty about where and when he can eventually find a free parking lot.

In Chapter 6 of his work — "The geography of transport systems", Rodrigue et al. [26] argues, that parking in the city centers of modern big cities with population bigger than one million inhabitants is one of the most prevalent transport problems. He claims that "cruising" in the search for a free curb parking space may account for more then 10% of the local circulations as the drivers can spend up to 20 minutes looking for a parking spot.

There are systems integrated into parking garages, that make help people to find a free parking space by offering the number of free parking lots available at the moment. The modern maps usually contain the positions of such parking garages mapped with their GPS coordinates. These allows these parking garages to be easily found with an ordinary GPS navigation device that one can nowadays find in most cell phones. 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.

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

Shoup [29], 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 between 8 and 74 percent of the traffic was 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 the thing that may lead to frustration.

Rodrigue et al. [26] 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 such uncertainty much better then the 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 one the best known examples is the "Parking on demand" system by Pierce and Shoup [24, 25]. 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 automatism in it. In order for this system to be used in an autonomous fashion one has to introduce a planner that considers not only spacial information but also occupancy and pricing.

Another down side 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 to be able to map the paring spaces, estimate their occupancy information and search for one in an optimal and fully autonomous way.

Recently autonomous vehicles have become a topic of great interest. In the latest years, auto companies and software companies as well as the universities all over the world have focused on developing their own autonomous vehicles. These vehicles can already impressively well navigate in the cities [33, 19, 20] and are able to take people to an arbitrary point even in difficult urban environments. We believe, that these cars are to become safer, more efficient in the means of fuel consumption and more predictable than an average human driver. They can already learn how to adopt behavioral knowledge from the traffic [22, 30]. Following the recent success of the Google self driving car [21] and the fact that it is now officially allowed to use self-driving autos on public roads in

California, Nevada and Florida, US, we predict that with time 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 [4, 18, 39]. Despite being able to park in an autonomous fashion, they are yet unable to know where to search for a free one by themselves and as a consequence where to search for a good one.

We address precisely this issue. We present an approach that provides a system for mapping the parking spaces while predicting their occupancy via integrating information from multiple observations. Not only we are able to estimate this information but we also present a way to plan the route in such way that it eliminates uncertainty in finding the free parking place and guarantees to minimize the time spent for cruising while searching for a free parking space and walking from the found one to the destination.

2 Related Work

The problem of finding free parking lots receives more attention with the continuous growth of the cities and their population. As far as we have found out, we can divide the works in the field in two main categories depending on the sensor setup. The detection of the parking lots 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.

2.1 Detection with static monocular cameras

To the best of our knowledge, the main purpose of such systems is to simplify the sensor setup for occupancy detection in the modern parking garages. These works differ in the features utilized for car detection as well as in methods of clustering these features.

Wu and Zhang [40] presents an unsupervised system to monitor the occupied parking slots. The authors were using a stationary monocular camera to detect parked vehicles. In their work, car detection is solely based on color. The authors use the support vector machines (SVM) to cluster the features to distinguish between occupied and free parking lots. They pre- process the ground truth images of the parking lots and search for color differences between the measurements and the ground truth measured for an empty parking lot.

their or his?

True [36] shows a similar approach. Alike with the previous papers he presents an unsupervised system for parking lots detection. The author uses an overhead static camera. He uses human-labeled parking lots' positions to define the spacial arrangement of the parking lots. The author distributes chrominance channels of the images from the camera into bins, storing a histogram of these values for each parking lot, 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 [37] utilizes a static monocular camera and shows 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 lots, which 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. [14] shows a system that uses a set of overhead cameras for occupancy detection in indoor and outdoor scenes. The focus is set on 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 [12] and Huang et al. [13], use a static monocular overhead camera and a 3-layer Bayesian hierarchical framework (BHF) the authors of the paper specifically addressed 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 lot.

Yusnita et al. [41] proposes a system to detect occupancy status of the parking lots of indoor parkings. They are using an overhead, strictly vertically oriented camera. The parking lots are manually labeled by a marker that states their occupancy state. For each query parking lot the authors produce a binary image which is then analyzed to find if the parking lot 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. (end)

2.2 In-vehicle detection

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

Fintzel et al. [11] and Abad et al. [1] model free parking lots 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 lots by empty spaces between the planes.

he????

Coric and Gruteser [8] also moves focus from static overhead cameras on the developed parking lot to estimating parking lots for on-street parking. They use an ultra-sonic sensor mounted to 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 time a car was observed in a place the more it is likely that the space occupied by this car is a valid parking spot. As a result all measurements were incorporated with the GPS measurements to form a global map of parked cars, which the authors used to detect wrongly parked cars.

Following the focus on the in-vehicle sensor setup, Schmid et al. [27] 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 lots are then detected on the given grid by analyzing occupancy grid cells on curb and on other parked cars. This provides an estimate of the parking lot in 3D that allows for precise parking.

Suhr et al. [32] proposes a system that is able to detect parking lots 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 which can prove to be useful when analyzing a free outdoor parking.

A fully automatic system for detection of parking slots' markings is presented by Suhr and Jung [31]. The authors utilize a tree structure performing a bottom-up and a top-down approach in order to classify all parking slots as such and leave out all false detections.

2.3 The contribution of this thesis

In this thesis, following the works in In-vehicle detection, we present a system that is capable of detecting parked cars utilizing the in- vehicle sensor setup, consisting of a stereo camera or a monocular camera fused with laser range finder. We also extend this functionality by estimating the occupancy probability for each parking lot in a global

map and additionally present a planner capable of efficiently finding a route to a free parking lot.

My own contribution???

3 Our Approach

3.1 Overview

In this chapter we describe in detail the three main components of our system: visual car detection, creation of the map of parking spaces 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 combining these parts into one system.

We start by a coarse outline of the way these different parts are interconnected. Later in the chapter we show each of the sub-parts in detail.

- *Perception:* We visually detect cars in the images from the camera and find out how far from the image plane are they situated by fusing the detections with the depth data, received either from the stereo camera or from the laser range finder (Section 3.2).
- *Mapping:* 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).
- *Planning:* In the modeled environment we search for actions that minimize the time spent on searching for a parking space and walking from it to the goal (Section 3.4).

3.2 Perception

We start by motivating the choice of the visual detection framework. There is a number of object detection methods to choose from. We focus on three approaches, utilizing different visual features: Haar features, local binary patterns (LBPs) and histograms of oriented gradients (HOGs). We further provide a short comparison of these with relation to our problem.

The work of Viola and Jones [38], introduces the Haar feature based object detection. It provides good detection rates 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.

Utilizing the local binary patterns (LBPs) as presented by Trefný and Matas [35] addresses the illumination issues and overall boosts the detection rates and performance. 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 on the training stage, which is, while not being a big problem, still is an important argument, especially on implementation stage.

not sure about this part

Considering the above, we apply the HOG-based detector following the work of Dalal and Triggs [10]. 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 advantages. 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 the work of Dalal and Triggs [10], we make use of the histograms of oriented gradients placed in a dense grid on the query images. For each 8×8 pixel block in the image a histogram is formed by assigning the gradient orientations to 9 bins, that cover the angle of $180°$. One HOG descriptor consists of several such histograms.

To detect the front and read 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. Therefore, the square drawn around the car will contain higher percentage of useful information in comparison to a rectangle. To detect the side of the car, the descriptor window should be 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

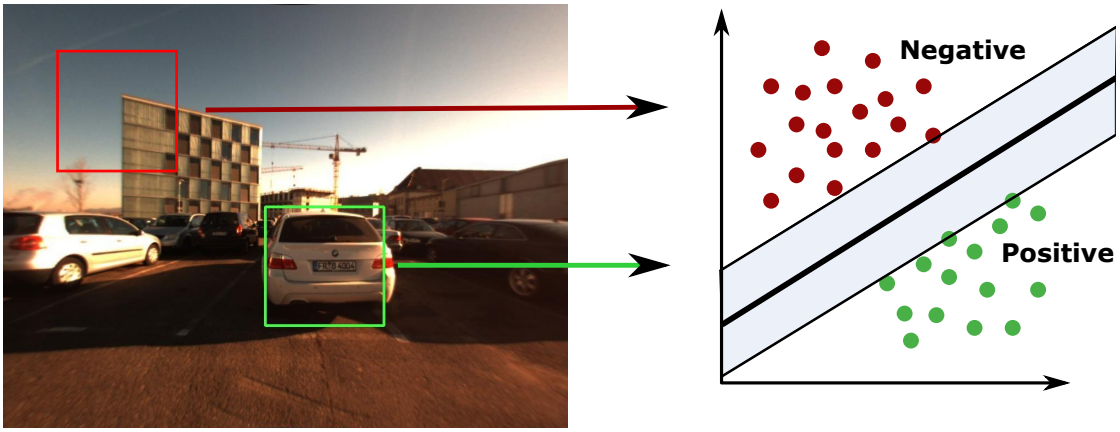

Figure 3.1: Illustration of HOG-based detection and clustering with SVM.

in 256 histograms for front/rear view and 128 histograms for the side view.

We form a vector from all the 9 bin values from each of 256 or 128 histograms contained in the detection window. 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 similar dimensionality. We further search for a hyperplane to divide these points into positive and negative ones with the use of the support vector machines. This is illustrated in Figure 3.1

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 [9] 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 [7].

Based on the trained classifier, we carry out the detection in a cascade fashion via the sliding window approach. For each sliding window content we create a HOG descriptor which we test against the pre-trained SVM classifier to find out on which side of the decision boundary it is. If the current HOG belongs to the side where the SVM has assigned the cars then the current sliding window is a region of interest and contains a car. We store each detection as a rectangle by storing the pixel positions of the corners of the detection window around the car.

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. Knowing the internal camera parameters we reconstruct the relative position of each pixel of 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 location in the $2D$ image and X, Y, Z is the $3D$ position that corresponds to the pixel (u, v) in the camera frame. $3D$ position that corresponds to every pixel in the image allows us to combine this 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, that is contained in this particular region of interest we consider taking the median of all depth values that fall into the given 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 were 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.

This high variance of the input data results in somewhat unreliable results. 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.

The laser mounted on the robot provides a sufficient opening angle in order to cover all the space related to the image from the camera. We search for the beams, that span through the area covered by the image. The camera is mounted on the same Z axis with the laser. This yields the choice of the laser beams, that contain depth values associated with the car:

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

where α_{camera} is the angle of the camera with respect to the direction of the laser, 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, this angle is equal to $-\frac{3\pi}{4}$.

The endpoints of the beams in Z provide us with consistent $3D$ information. In order to account for possible occlusions or measurement errors we take the median of these values.

3.3 Model

The methods mentioned above provide the cars positions relative to the camera. Integrating this information with GPS measurements or SLAM-based pose estimates, we calculate the positions of the detected cars in the world frame. In this sections 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 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 [23]. 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 [23], 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:

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

add log odds

In order to gain efficiency, we furthermore use the log-odds formulation of Moravec and Elfes [23], so that the operations in Eq. (3.8) are realized via addition and subtractions in the log-odds space.

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

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

$$P(a^i|z_t) = \begin{cases} 0.45, & \text{if before detection} \\ 0.9, & \text{if cell stores detection} \\ \text{prior}, & \text{if after detection} \end{cases} \tag{3.9}$$

Furthermore, we define the observation model along the rays, that span from the camera position to the defined $z_{\max}$ 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 [3] we search for the cells that form the frontier of the camera's field of view that consists of the cells that lie on the line between the above-mentioned left and right margin points.

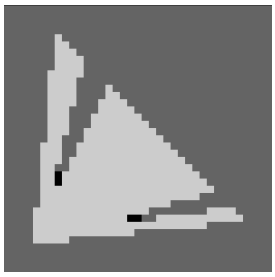

Figure 3.2: An example of Bresenham based update from a single image with two detected cars.

For each cell q we carry out the same algorithm from the camera cell c to the query cell. Whenever we encounter a cell, where a car is situated, the next cell is also updated as occupied, while all the ones that come afterwards are not updated as "not visible". This can be seen in an example in Figure 3.2.

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 angles 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 with the occupancy grid maps approach via the static state binary Bayes filter as defined in Eq. (3.8).

The occupancy information can be measured and accumulated at any needed time, for example on particular days of week.

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 a driving near every parking space in the parking to be one session. The parking lots that were not updated as occupied during the session are updated as free.

We would like to stress, that this approach allows storing both spacial and occupancy information in one distinct structure.

3.4 Action Planning

In this section we formally define the action planning framework that searches for a path to a free parking lot. We want to find the optimal one, but it depends on the application what an optimal solution to this problem is. We therefore define what we consider optimal in this work.

There are typically parts of the parking, which are better in the sense of their distance to a common destination. This also means that the "better" parts of the parking lots are more likely to be occupied than the other, which 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 then walking

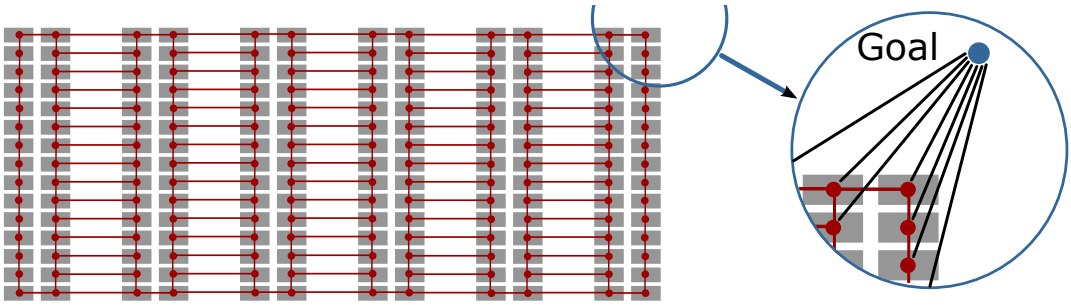

Figure 3.3: Graph based on the parking. The vertices correspond to the parking spaces. The edges between the vertices represent possibilities to reach one state from another. Every state is additionally connected to the goal state

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.

We formally describe the parking as a graph. The states are defined as the set of vertices of the graph V, each of which represents a point in space. The graph also has a set of edges E — a function $V \to V$, that corresponds to the actions that can be carried out from one state to another.

In the grid based situation, the states are the grid cells and the edges are connecting the adjacent cells.

In the case, where we explicitly map the parking lots — they, along with their spacial information, form the vertices of the graph. In this case we define the edges so that they hold the spacial information of the parking.

In both cases all vertices are connected to the goal state. The probability of the action that allows the agent to follow such edges relies on the occupancy probability of the related parking lot and is formally defined later.

We also define the set of all actions that can be carried out in any state. It consists of five distinct actions: “up”: $\uparrow$, “down”: $\downarrow$, “left”: $\leftarrow$, “right”: $\rightarrow$ and “park”. Formally:

$$A_{\text{move}} = \{\uparrow, \downarrow, \leftarrow, \rightarrow\} \tag{3.10}$$

$$a_{\text{park}} = \text{“park”} \tag{3.11}$$

$$A = A_{\text{move}} \cup a_{\text{park}} \tag{3.12}$$

As stated in the work of Tipaldi and Arras [34], 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 for our problem.

Following the work of Bellman [2] we define the MDP by the initial state S_0, transition function $T(s \mid s', a)$ and a reward function $R(s, a, s')$.

Here, $T(s \mid s', a)$ denotes the probability of reaching state s' if and 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.

For four actions that describe driving around the parking, 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:

$$\forall (s, s') \in E, \forall a \in A_{\text{move}} : P(s \mid s', a) = 1 \tag{3.13}$$

$$\forall s \in V, (s, s_{\text{goal}}) \in E : P(s \mid s_{\text{goal}}, a_{\text{park}}) = P(\text{free}) \tag{3.14}$$

where $P(\text{free})$ is a probability of a parking lot to be free, observed through a period of time. To be consistent, we set the remaining probabilities to the values that guarantee that $\forall s, s' \in V, (s, s') \in E : \sum_{s'} P(s \mid s', a) = 1$. This ensures that 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 in any state.

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 and while carrying out an action of parking, with the probability of 0.6 there should be a 0.4 chance to stay in the same place.

The function $R(s, a, s')$ likewise defines the reward for reaching state s' from state s carrying out action a. S_0 denotes the state in which we initially put the agent.

The motivation behind the rewards is based on the time it takes the agent to carry out an action. We assume, that the agent needs a constant time for moving from one state s to another s'. If he cannot carry out the action and is forced to stay in the same position, he regardlessly loses time. We therefore define a negative reward for carrying out any action from A_{move} that penalizes longer paths:

$$\forall (s, s') \in E, \forall a \in A_{\text{move}} : R(s, s', a) = R(s, s, a) = \text{-const} \tag{3.15}$$

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 — therefore 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 $r_s = \sqrt{(s - s_{\text{goal}})(s - s_{\text{goal}})^T}$ is a distance between the arbitrary state $s \in \mathbb{R}^2$ and $r_{\max} = \max_{s \in V} r_s$ is the greatest of those distances. Formally:

$$\forall s \in V, (s, \text{goal}) \in E : R(s, s_{\text{goal}}, a_{\text{park}}) = r_{\max} - r_s \tag{3.16}$$

The proposed reward function guarantees, that the agents tries to find a trade-off between shorter driving and shorter walking paths, weighted by the occupancy probability of the parking lots.

Transition function and reward function are necessary to define the utility function, which is a measure of how good the state is in the long run:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right] \tag{3.17}$$

We choose the optimal action in each state using the Maximum Utility Principle, that is, choose: the action that maximizes the expected utility of the subsequent state:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s \mid s', a) U(s') \tag{3.18}$$

Provided, that the utility of the state is the expected sum of discounted rewards from that point onwards, we define the utility via Bellman Equation:

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s \mid s', a) U(s') \tag{3.19}$$

To solve the MDP we make use of the policy iteration algorithm which we briefly outline here.

The policy iteration algorithm alternates the following two steps, beginning from 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 policy π_{i+l}, using one-step look- ahead based on U_i (as in Equation 3.18).

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 minimizes the expected time spent on searching for the free parking lot and walking from the found one to the goal.

explicitly state about replanning

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 a decision based on the better background.

Because the graph is updated, we carry out the policy iteration algorithm once more to define the new optimal actions for each state.

We repeat this re-planning step until we find a free parking lot.

4 Experimental Results

We make extensive use of three components: visual car detection, parking lot modeling and planning.

4.1 Visual Car Detection

We start this 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 different views of the cars — front, back and both left and right sides.

The sides of the car are invariant under the mirror transformation. Front and back of the car, though different to human point of view, seem to be similar in the HOG visualization. Therefore, the corresponding datasets can be 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

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.

training the front/rear detector. In addition to these positive examples we also convene 7972 negative images. The examples of the 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 [6], Motorway Car Dataset by Caraffi et al. [5] and various car images found in a public domain via Google.

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 [10] we cut false detections that appear in the first runs of training the classifier with a clipping tool [28] and add them to the negative training dataset.

We use the trained classifier on the images from the camera mounted on the car or, possibly, 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 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 4.5.

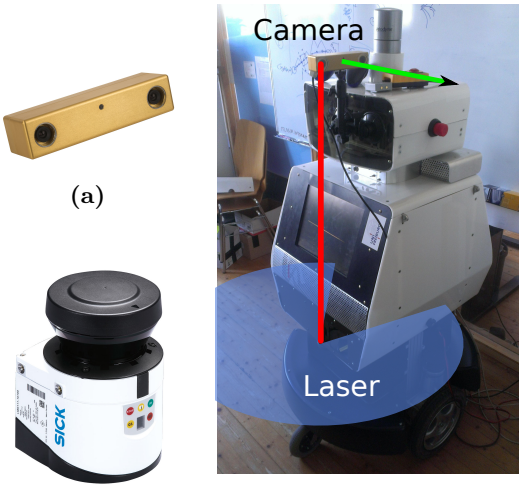

Figure 4.3: A Bumblebee stereo camera (a) and a laser range finder (b) mounted on the Obelix robot on the same vertical axis (c).

4.2 Parking Lot Modeling

We present occupancy data aggregated to the occupancy grids as well as into the predefined parking lot positions. The illustration of differences in the maps representations can be seen in Figure 4.4.

The occupancy data is based on the 2D data obtained either from the stereo cameras (Figure 4.2) 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 as the laser range finder. We search for the endpoints that represent the cars as described in the section Perception of chapter Our Approach.

Figure 4.2 shows that the depth information computed from the disparity information between two stereo images lacks precision and overall is noisy. 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 falling into the detected region of interest.

As the result of this, we use the laser range finder based depth measurements. Laser is precise in measuring the distance. Because of its mounting position on the human knee level, it also does not suffer from the partial transparency of the cars because the lasers never encounter the cars' windows.

Figure 4.4 shows car detections as red dots. We produce these detections fusing the

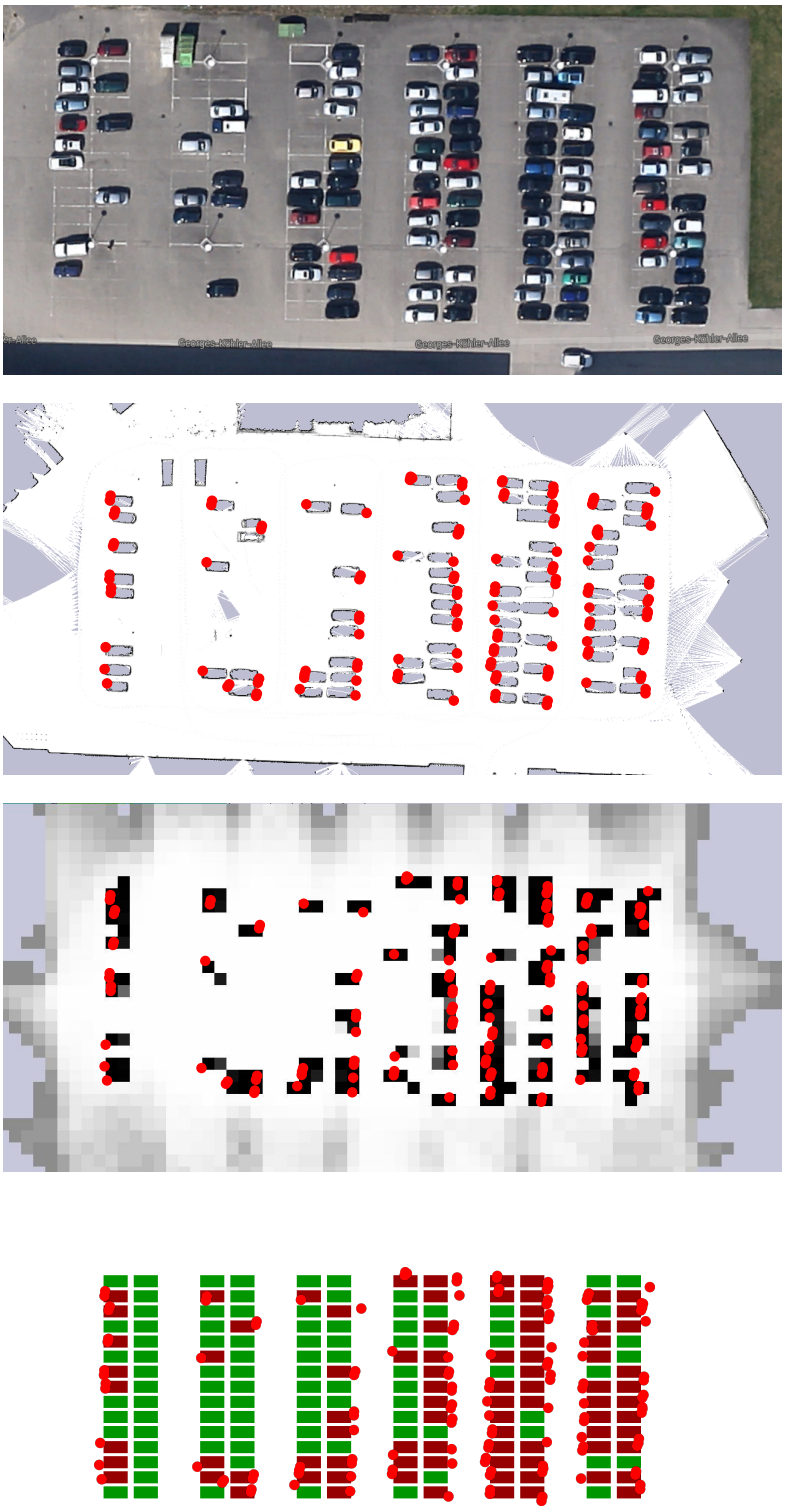

Figure 4.4: Illustration of different mapping approaches.

laser range finder measurements with the visual detections as described in Chapter 3.

The precision of car position modeling can be seen in Figure 4.4. The occupancy grid maps suffer from discretization errors. We have observed situations when the detections of the same car fall into different grid cells, causing erroneous map representation.

Another problem with occupancy grid approach is the ad-hoc modeling of the car orientation. We argue, that using the observation model, as seen in Figure 3.2, yields the orientation of the detected car to be influenced by the angle from which it is observed. This is the second source of the errors in the map representation via occupancy grid maps.

4.3 Planning

As can be seen in the previous section, the question of how to use the occupancy grids in order to build a planner on top of the occupancy information in them is still open. We thus currently focus on a planner based upon the pre-defined parking lots positions. However, the planner itself is generic, which allows to use it in any situation alike. For now we can manipulate the constant time for taking the next action. It can be seen, that if this value is set to a very big value, the optimal action for each state is to park right there just to keep the overall reward positive. If we on the contrary set this constant to 0, the optimal decision will be to move around the parking lot until we have found the best (closest to the goal) parking lot and then try to park there.

The most optimal result is achieved if the cost of moving between the parking lots by car is the distance between them, divided by the speed of the car. This makes the algorithms consider a trade-off between shorter routes while searching for a place and better parking lots, situated closer to the goal, weighted by the occupancy probability. The next example shows the behavior of the agent under different current occupancy.

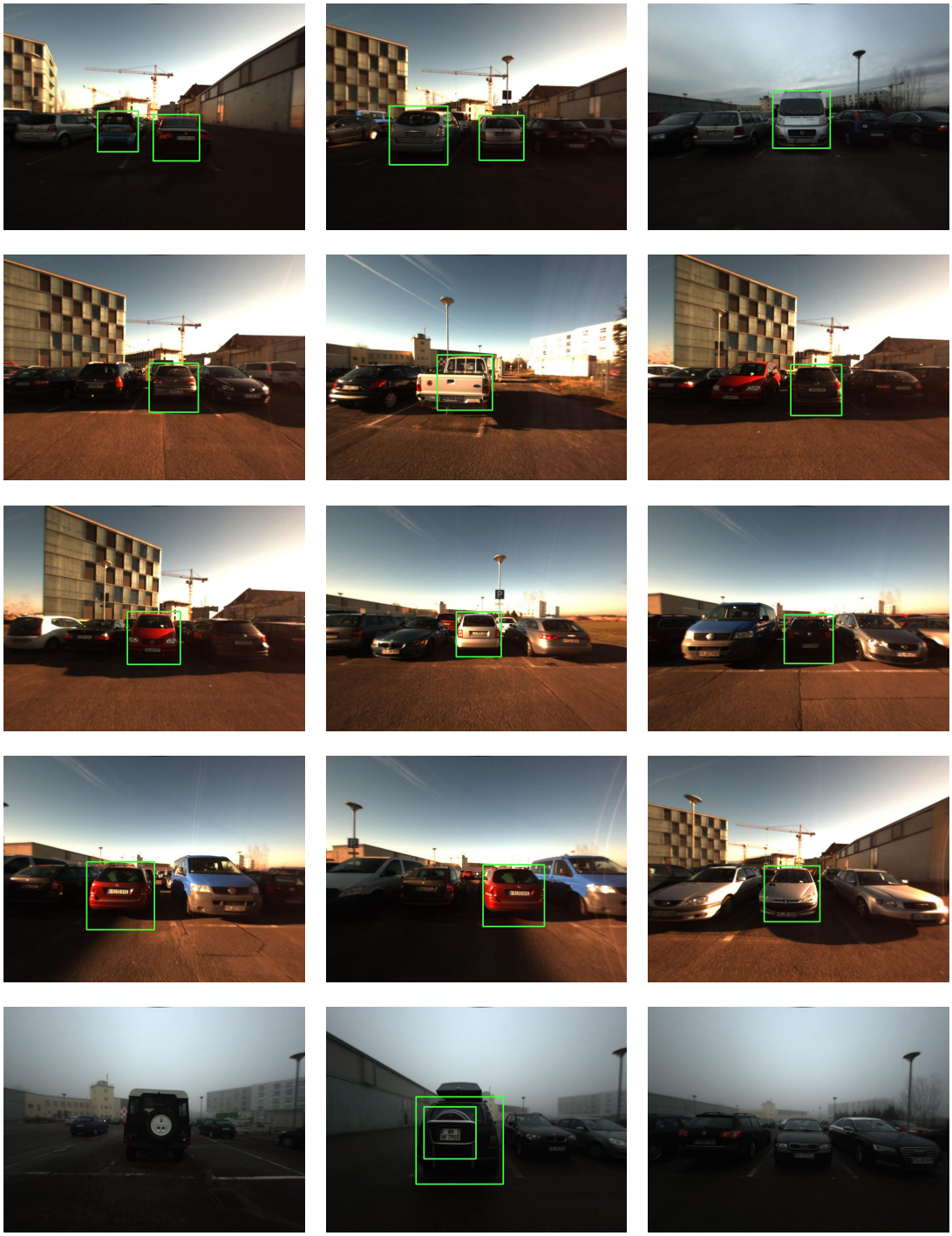

Figure 4.5: Examples of detections of different cars under different light conditions.

5 Conclusion and Further Work

We have build a system that integrate perception, mapping and planning for solving a task of efficiently finding a free on-street parking lot.

The algorithm presented in this work relies on visual detection of the cars in the streets. As no reliable fast car detector was found we have written one, which is going to be made open-source in the near future.

Throughout the course of the work we have experimented with different methods for combining metric information with visual detections such as depth from stereo cameras or lasers. Currently the laser range scanners have proven to provide a better result, however it is of great interest to build a system that would provide the same level of detail using purely visual information.

It is important to have a reliable depth estimate only in case of having a good estimate of the position in the world. We use SLAM ([16, 17, 15]) for this purpose.

Not only we have experimented with perception, but also with mapping. We have implemented the the occupancy grids based approach, which however suffers from the discretization errors as well as from ad-hoc estimation of the position (including orientation) of the detected cars in the occupancy grid. Eventually we stick to the system where we provide our system with additional knowledge about where the parking lots are situated. This information has to be obtained only once for a new environment.

This system then provides a good estimate of the occupancy information that is received via multiple observations of the same spot on different days or simply in different times. It may be subject for future work to estimate the occupancy information for particular day of week or particular time of a working day, etc. The occupancy information is basically a probability of a parking lot to be free when observed.

This occupancy information along with the position of the parking lots in the world allows for a planner to be introduced. This planner, unlike greedy A* and alike searches not for the best parking lot in the means of position or occupancy, but takes into account the trade-off between these two quantities, minimizing the expected time it takes us to park a car and get to the final goal by foot. It is still possible to experience problems finding the best parking lot, but our approach eliminates the high uncertainty

of this process and provides an opportunity to carry out the parking process in a fully-autonomous fashion, while guaranteeing to find the best action in the means of expected time needed for the whole process.

There is however place for future work. To see a fully integrated and easily accessible system we will further focus on making the detection of the parked car position purely based on visual data (via improving stereo-camera depth acquisition). The other vector of interest is improving the approach for mapping in order to leave pre-defined parking lots' positions behind and to be able to efficiently map any given environment without any prior knowledge about it.

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] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957.

[3] 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.

[4] Wolfram Burgard, Dieter Fox, and Sebastian 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`.

[5] Claudio Caraffi, Tomas Vojir, Jura Trefny, Jan Sochman, and Jiri 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.

[6] Peter Carbonetto and Gyuri Dorkó. Inria car dataset, 2004. URL `http://www.cs.ubc.ca/~pcarbo/objrecls/data/cars.tar.gz`.

[7] Chang Chih-Chung and Chih-Jen 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`.

[8] Vladimir Coric and Marco 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`.

[9] Corinna Cortes and Vladimir 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`.

[10] 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.

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] 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.

[18] Chen-Kui Lee, Chun-Liang Lin, and Bing-Min 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.

[19] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, Olivier Koch, Yoshiaki Kuwata, David Moore, Edwin Olson, Steve Peters, Justin Teo, Robert Truax, Matthew Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan 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.

[20] 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.

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

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

[23] 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.

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

[25] 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.

[26] 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`.

[27] 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.

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

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

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

[31] 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.

[32] 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`.

[33] 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 Martin Buehler, Karl Iagnemma, and Sanjiv 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`.

[34] Gian Diego Tipaldi and Kai 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`.

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

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

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

[38] 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.

[39] Zhao-Jian Wang, Jian-Wei Zhang, Ying-Ling Huang, Hui Zhang, and AryanSaadat 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`.

[40] 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`.

[41] 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.