ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

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

Inferring Symbolic Actions from Demonstrations for Solving Manipulation Tasks in the Real World

Masterarbeit

Nichola V. Abdo
Februar 2012

Betreuer: PD Dr. Cyrill Stachniss
Henrik Kretzschmar

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

1 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 repeatedly detects and maps the parked cars in the area of interest and estimates 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 lot, while considering spacial constraints like the distance to the end destination.

2 Introduction

We believe everyone has been in a situation, when you are driving around a city, especially the one you have never been to, and try to find a free parking space. One can drive many circles around the center of the city with absolute uncertainty about where and even more importantly when a free parking lot will eventually be found.

Nowadays there are more and more cars in the streets every day, the cities grow and more people use their car to get from place to place.

There is a lot done to make it easier for people to park their cars — most indoor and some outdoor parkings offer a number of free parking lots available at the moment. The positions of all open parkings are mapped with their GPS coordinates and can be found with an ordinary navigator that one can nowadays find in any cell phone.

However, it is not always convenient to go to an underground parking as usually they are situated further from center. When one takes a car to go to any place of interest, one does this for convenience and therefore wants to park as close as possible to the destination. There are also some places that have no parking with occupancy information estimate in the neighborhood.

There are currently no solutions that provide any information on the positions of the parking lots in the streets and their availability. Therefore, we are on our own with this problem. This results in having to spend valuable time on driving along the same street for the third time in a desperate hope that someone has left and there is a parking spot available now.

This problem is annoying for humans, but people are dealing with uncertainty relatively well which is not true for robotized systems. In the latest years, several auto companies, such as Mercedes, BMW, Nissan, Volvo or even some software companies like Google or universities all over the world have developed their own autonomous vehicles. These vehicles can already impressively well navigate in the cities and are able to take people to arbitrary point even in hard to analyze urban environments. These cars are safer, more efficient in the means of fuel consumption and are a lot more predictable, causing therefore less traffic jams. We believe, that with time people will spend less time driving by themselves and will rely significantly 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. However, they are yet unable to find a free one by themselves.

We want to address precisely this issue. We present an approach that provides a system for predicting the occupancy in any area as well as an algorithm for on-ground mapping of the parking lots with the this additional information. Not only we are able

to estimate this information but we also present a way to plan the route in such way that is eliminates uncertainty in finding the free parking lot and guarantees to find an optimal, in the means of time spent for the search, parking lot.

3 Related Works

The problem of finding a free parking lot receives more attention with the neverending growth of the cities and their population. However, most of the work is oriented on providing people with simple information on the occupancy status of a particular parking garage or parking lot. There is quite significant amount of work done in the field of detection of the parked vehicles using either a static monocular camera or a set of such cameras. In the works of Qi Wu (2006) the authors present an unsupervised system to monitor the occupied parking slots. They were using a stationary monocular camera to detect parked vehicles. In their work car detection is solely based on color, SVM is used to distinguish between occupied and free parking lots. The authors preprocess the ground truth images of the parking lots and search for color differences between actual measurements and the ground truth measured for an empty parking lot.
Another work True is showing a similar approach. Alike with the previous paper, an unsupervised system for parking lots detection is presented. The author is using 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. The histograms are then classified into free vs. occupied using either k-nearest neighbors or SVM. He also presents a variant of the work based on classifying color patches that are situated around corner detector's regions of interest, this however proves to be not sufficiently efficient.
The authors of the next paper R. Yusnita (2012) proposed a system to detect occupancu status of the parking lots of the 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 slot 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.
Another work that utilizes a static monocular camera Marc Tschentscher shows a comparison of several different features for occupancy analysis. As in the works presented above the authors of this paper have an overhead camera pointing to the parking lots, which are manually labeled in the images. They compare a couple of visual features such as color histograms, gradient histograms, difference of gaussian histograms and Haar features. They also explore different methods for training a classifier such as k-nearest neighbors, linear discriminant analysis and SVM. They achieved detection rate of 98%

while performing in real time.
In another work, Ching-Chun Huang (a), Ching-Chun Huang (b), using 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.
The authors of these papers K.Fintzel, F. Abad here modeling 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 points in the images seen from different angles. In these papers they used the ultrasonic sensor and 3D vision sensor that detects points of interest on the bodies of the parked cars and tracking them compare their occurancy 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. Empty parking lots are then modeled by empty spaces between the planes.
The authors of this paper H. Ichihashi have developed a system that uses a system of overhead cameras for occupancy detection in indoor and outdoor scenes. The focus is put on the clustering algorithms, showing that the performance of the detector based on the fuzzy c-means (FCM) clustering and the hyperparameter tuning by particle swarm optimization (PSO) significantly outperforms SVM both in speed and accuracy of detection.
The authors of this paper Vladimir Coric move focus from static overhead cameras on the developed parking lot to estimating parking lots for on-street parking. In this paper they use an ultra-sonic sensor mounted to the side of the car to estimate the parking possibilities along particular streets. Uses 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.
This paper Matthias R. Schmid and Wuensche (2012) 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 cells on curb and on other parked cars. This provides relatively precise estimation of the parking lot in 3D.
The authors of this paper Jae Kyu Suhr (2010) proposed a system that is able to detect parking lots from 3D data acquired from stereo-based multiview 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 were able to achive reliable results with the detection rate of 90%
The authors of this paper Jae Kyu Suhr (2013) proposed a system to a fully automatical

detection of parking slots markings. They are utilizing 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.

4 Background

In order to be able to solve the given problem we will have to rely on several well known approaches, which I will briefly introduce in this chapter.

4.1 Occupancy Grids

In this thesis we partly make use of the grid maps to model the environment with respect to modeling the probability of cars to be present in the particular part of the world. Grid maps partition the space into a grid of rectangular cells. Each grid contains information about the corresponding area in the environment. In this thesis we make use of one particular realization of grid maps — the occupancy grid maps. Occupancy grid maps assume that each grid cell is either occupied by an obstacle or free. In our case, each cell therefore 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. However, grid maps suffer from discretization errors.

4.2 Object Detection

In this thesis we made extensive use of object detection to be able to detect cars from visual information. Based of the work of N. Dalal we made use of the histograms of oriented gradients (HOG) features placed in a dense grid on the query images. The histograms from the training data, describing positive and negative sets are then divided into positive and negative groups via support vector machines (SVM). The HOG/SIFT 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 that the local spatial or orientation bin size

4.3 Markov Decision Processes

Markov decision processes are used to carry out complex decisions in a fully observable environment with a stochastic transition model. A specification of the outcome probabilities for each action in each possible state is called a transition model, we will use $T(s|s', a)$ to denote 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. We also need to define the reward function $R(s, a, s')$, that defines a reward for reaching state s' from state s carrying out action a. Overall, the MDP is defined by three components:

- Initial state: S_0
- Transition Model: $T(s|s', a)$
- Reward Function: $R(s, a, s')$

These components allow us to define the utility function, which is a measure of how good the state is in the long run:

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi, s_0 = s\right] \tag{4.1}$$

The optimal action is then chosen using the Maximum Utility Principle, that is, choose: the action that maximizes the expected utility of the subsequent state:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s|s', a)U(s') \tag{4.2}$$

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

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s|s', a)U(s') \tag{4.3}$$

In order to solve the MDP there are two common strategies to be used:

- Variable iteration
- Policy iteration

In this thesis we will only focus on the second, which is briefly introduced 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 MEU policy π_{i+l}, using one-step look-ahead based on U_i (as in Equation 4.2).

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

5 Our Approach

5.1 Overview

Unlike the works presented in the related works we were interested in integrating all stages of solving the problem such as perception, mapping model and action planning into one system. We further present an integrated framework for carrying out the task of detecting the parked cars, integrating occupancy information into a model that is suitable for further planning and planning for actions itself, when carrying out a decision of finding an unoccupied parking lot. We will further describe the different steps that we have taken on this road as well as the connections between them.

5.2 Perception

In order to detect the positions of the parked cars we first needed to find a way for visual recognition of cars. To achieve this goal we looked at a number of approaches for object detection and classification.

- Haar-cascade detection as presented by Paul Viola (2001)
- Local binary patterns as presented by Trefný and Matas (2010)
- HOG descriptor based classifier by Dalal and Triggs (2005)

However, whichever algorithm we use, we still need to make a transition from the image space to the 3D world. For each detected car we want to know precisely where is it situated with relation to the agent's current position. For this purpose we need to use depth information, which we may obtain from one of these sources:

- Sonar Based Depth
- Stereo Camera Based Depth
- Laser Range Finder Based Depth

We will further focus on the second two of them. As soon as we know a relative position of the detected car to the camera, we can move on to the next section –Model, where we stack many detections together in order to model the joint occupancy information. For now, let's look into some more details in each perception step:

5.2.1 HOG Detector

After analyzing the good and the bad sides of these methods we decided to go with the HOG-based method as it outperforms the Haar-based method while providing us with shorter training times when comparing to the local binary patterns based methods. There are still some differences between our approach and the one presented byDalal and Triggs (2005). These differences are due to the fact that we were interested in detecting cars, while the original paper is on detecting people. We were mainly focusing on detecting front/rear facing cars, but the approach is easily extended onto the side-view as-well. To be able to detect the front and read sides of the cars, we needed to switch the size of the hog descriptor window from 128×64 to 128×128. In the training faze we have considered around 1000 manually chosen 128×128 patches containing cars and around 8000 negative examples. Each HOG descriptor is then unfolded into a point in a multi-dimensional space. These points then have to be clustered into positive and negative ones. We describe the background for this decision in the next section.

5.2.2 SVM Classifier

In order to carry out the decision in the test data, we train a linear SVM classifier on top of all HOG descriptors. A more complex decision boundary could of course also be used, but linear SVM, despite its simplicity, and remembering that visual detection was not the key contribution of this work, provides reasonable results in reasonable time. The test data detection is carried out in a cascade fashion as presented by Paul Viola (2001) via the sliding window approach. For each sliding window content we create a HOG descriptor which is then tested against the pre-trained SVM classifier in order to find out on which side of the decision boundary it is. If the current HOG belongs to the area where cars are then the current sliding window is a region of interest and contains a car. Each detection is then stored as a rectangle in the coordinates of the image it belongs to. This ends the visual detection part and allows us to move on to the 3D world.

5.2.3 Depth Information

As we have pointed out before, we were mainly focusing on stereo cameras and laser range finders for depth acquisition. We did not consider using the sonar because it is by far not as reliable as a laser, while likewise needs a complicated setup. The stereo cameras, on the contrary, hardly need specific setup, but, on the down side, produce noisy information. However, for testing purposes, we decided to try using both — the stereo camera and the laser.

Stereo Camera

In order to find the depth from the stereo camera, we need to first calculate the disparity image from left and right images taken from the video stream. Knowing the internal camera parameters we can then reconstruct the relative position of each pixel of the disparity image. We first find 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. After Z is determined we can focus on finding X and Y coordinates from the ordinary projection equations:

$$X = \frac{uZ}{f}$$

$$Y = \frac{vZ}{f}$$

where u and v are the pixel location in the $2D$ image, X, Y, Z is the real $3D$ position. When we have the $3D$ positions for every pixel in the images, we can combine this information with the visual detection part. From the previous part we have a region of interest in the image coordinates, which allows us to accumulate the depth of all pixels that fall into it. We can then analyze all of these values to find the distance to the car, that is contained in this particular region of interest. We considered taking either the mean or the median of the chosen values. Given, that the depth that comes from the stereo camera is quite noisy and that the depth values of the pixels originate on the surface of the car and therefore contain quite a big amount of variance, we picked a median as an option as it is the most resistant statistic, having a breakdown point of 50%: so long as no more than half the data is contaminated, the median will not give an arbitrarily large result.

Laser Range Finder

Even though the stereo camera setup is a lot cheaper and easier to mount, it lacks robustness due to the noise that is present in disparity images and to some degree to the erroneous values that fall into the detected region of interest (such as around the car or in the glass parts of it). We therefore also tested a setup with a laser range finder. We used a EUROPA robot Obelix which has a 270° laser mounted approximately on the human knee level. This setup proves to be a lot more robust in terms of finding the exact $3D$ information about the position of the detected car. However, when using laser we need a different approach because there we only have images from a monocular camera and thus do not have a per-pixel association with the depth field the way we have it using the stereo camera. As the laser mounted on the robot provides us with 270° span it is able to cover approximately all the space related to the image from the camera. We then need to find the beams, that span through the area covered by the image. Given the fact, that the camera is mounted on the same Z axis as the laser this can be done in

a straightforward fashion. We are interested in the beams, the numbers of which follows this law:

$$\{beam_n | \forall n : \gamma_{start} < \alpha_{camera} + \alpha_0 + \alpha_n < \gamma_{end}\}$$

where α_{camera} is the angle of the camera with respect to the direction of the laser, γ_{start} and γ_{end} are the starting and ending angle of the detection bounding box in the image, α_n is the angle of the n-th beam in the laser frame. All the beam endpoints along with the direction of the beams provide us with consistent $3D$ information. In order to account for possible occlusions or mistakes we take the median of these values as a true position of the detected car. After finding the correct car position we may move on to the next part and aggregate occupancy information into a consistent spacio-temporal map.

5.3 Model

In the previous sections we have shown the steps of visual detection of the cars and finding their real $3D$ world position. Whichever path we take in the previous step, now we have a number of real world coordinates, that need to be integrated into a consistent map. In this section we will present two approaches that we have chosen to target this task.

5.3.1 Occupancy Grids

The first straightforward decision for mapping the occupancy is using the occupancy grid maps (see Background: Occupancy Grids). In this case we model each cell's occupancy as a static state Bayes filter. Let z_t be an occupancy probability estimate of a cell in the environment in time t. Considering the probabilistic nature of occupancy of every distinct cell we integrate multiple measurement, taken in different times into one model. We achieve that by using the static state binary Bayes filter that integrates occupancy probability for each cell i in M. Following the work of Moravec (1988), we can compute a recursive update formula for $P(a^i|z_{1:t})$

$$P(a^i|z_{1:t}) = \left[1 + \frac{1 - P(a^i|z_t)}{P(a^i|z_t)} \frac{1 - P(a^i|z_{1:t-1})}{P(a^i|z_{1:t-1})} \frac{P(a^i)}{1 - P(a^i)}\right]^{-1} \tag{5.1}$$

In order to gain efficiency, one can furthermore use the log-odds formulation of Moravec (1988), so that the operations in Eq. (5.1) are realized via addition and subtractions in the log-odds space. We can therefore model the probability of each cell to be occupied by a parked car at each time t. This information can be accumulated at any needed time as well as on different days. In order to fully define the equation given above we need to

define and observation model $P(a^i|z_t)$

$$P(a^i|z_t) = \begin{cases} 0.45, & \text{if before detection} \\ 0.9, & \text{if cell stores detection} \\ \text{prior}, & \text{if after detection} \end{cases} \tag{5.2}$$

Furthermore, the the observation model is defined along the rays, that span from the camera position to the defined $z_{\max}$ and along the defined for the camera field of view. Of course, we have an occupancy grid based world, so there has to be a way to compute the cells that fall into the field of view. In order to carry out this action, we first compute left and right further points. We find the cells that form the end frontier by using the Bresenham algorithm as defined by Bresenham (1965). Then, for each cell we carry out the same algorithm from camera cell to this 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 gives us the means of formulating full occupancy grid that can store occupancy information from different days. It is theoretically then possible to move on to planning.

5.3.2 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 may happen, that the detection is assigned to different cells of the occupancy grids. This leads us to the problems with data association and therefore difficulties in creating a meaningful accumulated map. This, in its turn, can lead to wrong decisions of occupancy and may ruin or at east make it harder for a planner to carry out a decision. For now we decided to use a pre-defined set of parking lots, that can be set from on-ground data or from aerial images (such as Google Maps). The rest of the theory stays untouched. We still make use of the static state binary Bayes filter as defined in 5.1. However, the observation model has changed to to the absence of the occupancy grid. We now have different distinct parking lots, that are represented by a point in the space with its coordinates. When we get an observation of occupancy during one full session of measurements, we pick the point to update by using the closest neighbor from the number of available parking lots. The chosen parking lot is updated in a similar way to the occupied case of 5.2. After the full session of observation is carried out, the untouched cells are updated as free. though this approach does not provide us with a consistent map of the environment we still get the occupancy information of the correctly situated in space parking lots, which allows us to form a graph that represents the environment and move on to the next part: 5.4.

5.4 Action Planning

Once we have defined the model for mapping, we are able to formally define the action planning framework that can lead to an optimal decision in the means of picking a path on our way to the parking lot. It is an open question what is an optimal decision here. Whenever we try to find a parking space it is usually the case, that there are some parts of the parking lot which are better suited for parking. Usually it means, that these parts are closer to the destination where all the people lean to. This also means that the "better" parts of the parking lots are more occupied than the other, which leads us to a trade-off between going straight to the closest, but likely occupied parking lot, which results in spending time searching for a new one or parking in a more distant area, most likely free, but it takes a while to walk from there to the goal. Given the constraints described above, the framework that we present focuses on finding the optimal solution in the means of joint time spent on finding a free parking spot and walking from the found one to the goal.

As stated in the work of Tipaldi and Arras (2011), MDPs can be used when there is a need to maximize the joint rewards instead of greedily going for the best possible goal. We follow similar approach, while defining rewards and transition probabilities based on the environment that we have in our problem.

In order to solve the given problem in an optimal way, we define rewards, which depict the time spent for each transition from state to state an agent can take. We then use the Markov Decision Processes as described in section 4.3 to find a path that maximizes the reward. The resulting path does not go for the best spot, but maximizes the probability of encountering a free parking lot, while trying keeping the length of the search path as short as possible.

As it is stated in section 4.3, we need to define rewards and actions in order to fully define an MDP problem. First, lets start, by defining the state space. The states are defined as the vertices of the graph V, each of which represents a point in space. The graph also has edges E a function $V \to V$, that correspond to the actions that can be carried out from one state to another. In the grid based situation, the states are the grids cells. In the case, where we explicitly map the parking lots — they, along with their spacial information form the vertices of the graph. In order to define rewards and probabilities of actions we need to define the set of all actions that can be carried out. We are assuming to be in a world, where only five actions are defined — "up": $\uparrow$, "down": $\downarrow$, "left": $\leftarrow$, "right": $\rightarrow$ and "park". Formally:

$$A = A_{\text{move}} \cup a_{\text{park}} \tag{5.3}$$

$$A_{\text{move}} = \{\uparrow, \downarrow, \leftarrow, \rightarrow\} \tag{5.4}$$

$$a_{\text{park}} = \{\text{“park”}\} \tag{5.5}$$

Now we can define the reward function $R(s, s', a)$ as well as transition function $P(s, s', a)$.

Let us first start with probabilities. For four actions that describe moving around the parking lot in a car we considered 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:

$$\forall (s, s') \in E, \forall a \in A_{\text{move}} : P(s|s', a) = 1 \tag{5.6}$$

$$\forall s \in V, (s, s_{\text{goal}}) \in E : P(s|s_{\text{goal}}, a_{\text{park}}) = P(\text{free}) \tag{5.7}$$

where $P(\text{free})$ is a probability of a parking lot to be free, observed through a period of time. To be consistent and to fully define the transaction model, we also need to set the remaining probabilities (the ones not set before) to the values such that $\forall s, s' \in V, (s, s') \in E : \sum_{s'} P(s|s', a) = 1$. This 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 has a 0.4 chance to stay in the same place.

Another part of defining the MDP is the definition of the reward function $R(s, s', a)$. We think that the motivation of the rewards should be based on the time it takes an agent to carry out an action. For now we assume, that it takes us a constant time for moving from one state s to another s'. If we cannot carry out the action and are forced to stay in the same position, we anyway lose the time. So, put bluntly, each action takes time. We therefore define a negative reward for carrying out any movement:

$$\forall (s, s') \in E, \forall a \in A_{\text{move}} : R(s, s', a) = R(s, s, a) = \text{-const} \tag{5.8}$$

We have defined the costs of moving, which "tells" our agent, that the best strategy would be to stay in one place. However, this is not the behavior we expect from an intelligent agent, so we need to define the rewards of reaching the goal state. The rewards should still be time based, but they have to form a decreasing linear function as the state closer to the destination should get a bigger reward than the one that is situated further. We decided to set the reward to $r_{\max} - r_s$, where $r_s = \sqrt{(s - s_{\text{goal}}) \cdot (s - s_{\text{goal}})^T}$ 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. Therefore:

$$\forall s \in V, (s, \text{goal}) \in E : R(s, s_{\text{goal}}, a_{\text{park}}) = r_{\max} - r_s \tag{5.9}$$

This guarantees, that the agent will try to find the way to maximize the reward he gets from parking in the described locations. Combining the defined transition model and the reward function allows us to fully define the MDP, which we are then able to solve with the policy iteration method as described in section Markov Decision Processes. The solution to the MDP guarantees to find an optimal solution to the given problem, which,

given the nature of our reward functions is also an optimal solution from the human point of view.

However, the MDP can fail on the given problem. The failure can be described as follows: the agent carries out an optimal strategy up to the point when he needs to carry out the parking decision. 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 need to plug in the observations of the world. Whenever we move past some parking lot we check if it is occupied and update the occupancy probability in this state to 1 or 0 respectively. We are then able to carry out a decision having a better background. To find what the next optimal action would be, we start the policy iteration on the newly defined graph for MDPs, providing the last found policy as initial. Under the assumption, that changing the occupancy probability of the observed vertices does not dramatically change the movement strategy of the agent, this yields optimal planning time because the policy iteration algorithm is carried out to the point when the policy does not change.

6 Experimental Results

Throughout the course of development of the full framework that we present in this thesis, we have made extensive use of such three components:

- visual car detection
- environment modeling
- planning

Let's focus on each of these as parts of an aggregated system.

6.1 Visual Car Detection

Following the work of Dalal and Triggs (2005), we created a training dataset that contains positive and negative examples that can be fully described with a HOG descriptor in order to cluster the outcome via SVM later. The cars don't look the same from different views, wherefore we have positive training data for different angles of the cars. We have considered to create training samples for eight different views of the cars. However, for our purposes there are some simplifications that can be taken into account. We do not need the diagonal views as the cars are usually observed from the side, front or back. The sides of the car are actually the same if we allow mirror transformation. Front and back of the car, though different to human point of view, seem to be quite similar in the HOG visualization. Therefore, the corresponding datasets can be combined into one. The positive dataset was filled from different sources, INRIA Car Dataset by Carbonetto and Dorkó, Motorway Car Dataset by Caraffi et al. (2012) and images found in a public domain via Google. In this work we were mainly focusing on the front/rear detection. The side detection can be handled in a similar way. The final positive training dataset contains 440 images, 128×128px each. Each of those images contains only a car and can be fully described with a HOG descriptor. The negative set is created in the way described in the work of Dalal and Triggs (2005). First, a number of 128×128px patches is sampled from the images, that contain no cars. This data is then used as the negative data for the classifier training process. The false detections of the outcome are also put to the same negative training set. This is performed several times. The final negative dataset contains 7972 images, each of which is 128×128px big. The system shows nice performance in detecting the cars it faces (we explicitly did not focus on detecting all

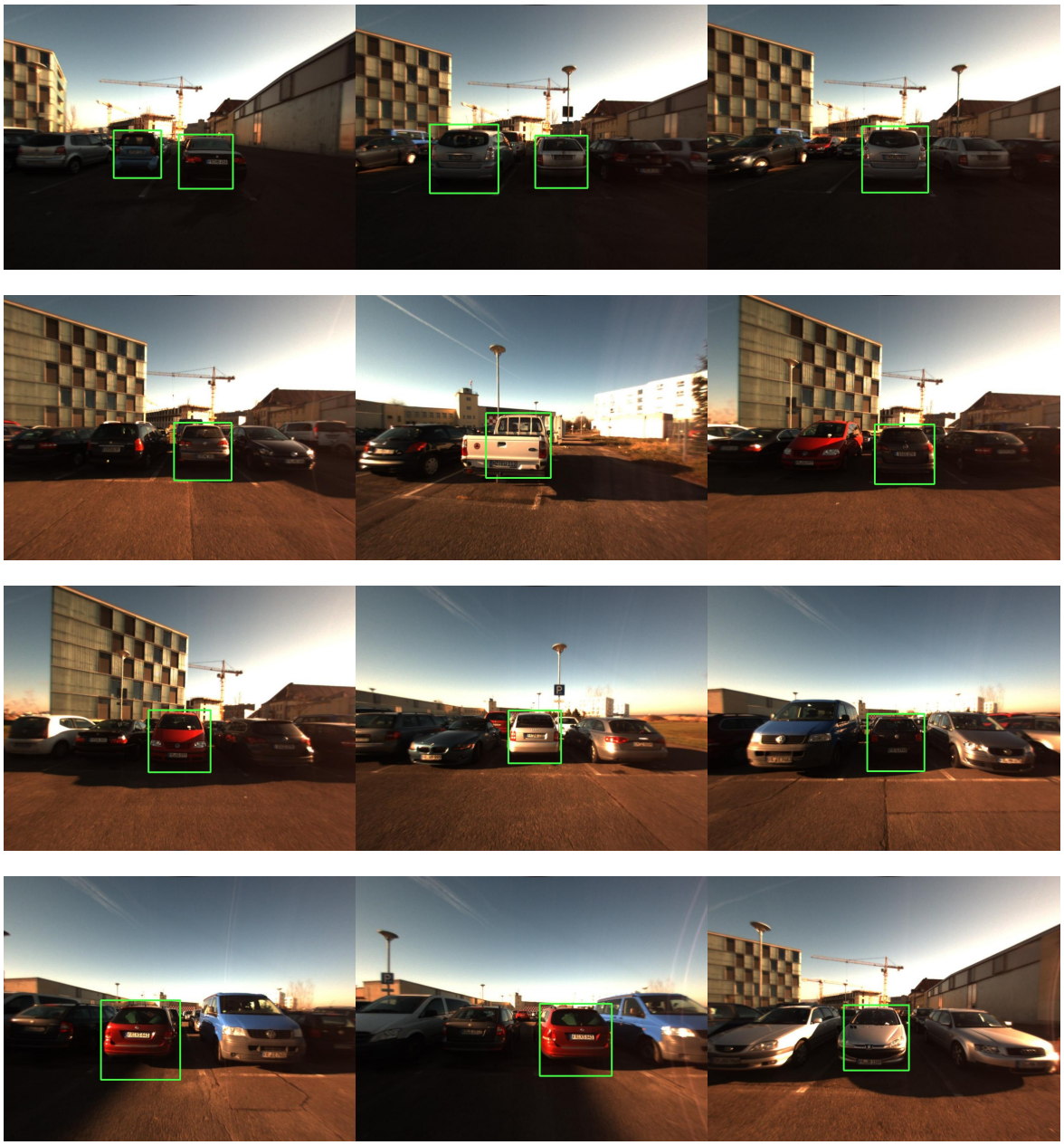

Figure 6.1: Examples of detections of different cars under different light conditions.

cars in the scene, such as those, that are found on the side of the image or parts of cars as it makes it harder to find the corresponding 2D position afterwards) and the detection rate of . Some of the detection examples are presented in Figure 6.1.

add detection rate for my approach

6.2 Occupancy Modeling

We here present the examples of occupancy data aggregated to the occupancy grids as well as into the pre-defined parking lot positions. The occupancy data is based on the 2D data obtained with lasers, mounted on the robot. 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 needed endpoints that represent the cars as described in chapter Our Approach, section Perception The precision of car position modeling can be seen in the figure.

do a photo of the robot with a camera

what to write more???

6.3 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.

Bibliography

J.E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, pages 25–30, 1965.

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.

Peter Carbonetto and Gyuri Dorkó. Inria car dataset.

Sheng-Jyh Wang Ching-Chun Huang. A hierarchical bayesian generation framework for vacant parking space detection. *IEEE Transactions on Circuits and Systems for Video Technology*, a.

Yao-Jen Chang Tsuhan Chen Ching-Chun Huang, Sheng-Jyh Wang. A bayesian hierarchical detection framework for parking space detection. b.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition*, 2005.

S. Wybo S. Bougnoux C. Vestri T. Kakinami F. Abad, R. Bendahan. Parking space detection.

M. Fujiyoshi A. Notsu K. Honda H. Ichihashi, T. Katada. Improvement in the performance of camera based vehicle detector for parking lot.

Ho Gi Jung Jae Kyu Suhr. Full-automatic recognition of various parking slot markings using a hierarchical tree structure. *Optical Engineering*, 2013.

Kwanghyuk Bae Jaihie Kim Jae Kyu Suhr, Ho Gi Jung. Automatic free parking space detection by using motion stereo-based 3d reconstruction. *Machine Vision and Applications*, 2010.

C.Vestri S.Bougnoux T.Kakinami K.Fintzel, R.Bendahan. 3d parking assistant system.

Marcel Neuhause Marc Tschentscher. Video-based parking space detection.

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

Hans P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, 1988.

Michael Jones Paul Viola. Rapid object detection using a boosted cascade of simple features. *Conference on Computer Vision and Pattern Recognition*, 2001.

Yi Zhang Qi Wu. Parking lots space detection. 2006.

Norazwinawati Basharuddin R. Yusnita, Fariza Norbaya. Intelligent parking space detection system based on image processing. *International Journal of Innovation, Management and Technology*, 3rd, 2012.

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`.

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

Nicholas True. Vacant parking space detection in static images.

Marco Gruteser Vladimir Coric. Crowdsensing maps of on-street parking spaces.