Free Space Detection Based On Occupancy Gridmaps

Freiraumbestimmung auf Basis von Occupancy-Gridmaps

Master-Thesis von Ruimin Zou aus Shanghai

April 2012

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department
Intelligent Autonomous Systems Group

Free Space Detection Based On Occupancy Gridmaps
Freiraumbestimmung auf Basis von Occupancy-Gridmaps

vorgelegte Master-Thesis von Ruimin Zou aus Shanghai

1. Gutachten: Prof. Jan Peters
2. Gutachten: M.Sc. Matthias Schreier

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 12th April 2012

(Ruimin Zou)

Abstract

The advanced driver assistance systems are auxiliary electronic systems in vehicles to assist the driver in certain driving situations. This paper focuses on the free space analysis, which allows the driver to recognize the antecedent obstacles directly in order to reduce the vehicle speed immediately to avoid the collsion.

The existing approaches to the free space computation in the automotive field are reviewed at the beginning of the work. After the revision of some basic knowledge, two algorithms for measuring the free space around the vehicle, which are based on an occupancy grid map with free evidence, is designed respectively in Matlab and C++. At last, the developed methods are examined in the framework wxWavE using the collected measurement data and conclusions are reached.

Keywords: Advanced Driver Assistance System, Free Space Computation, Stereo Vision, Occupancy Gridmap, Dynamic Programming

Zusammenfassung

Fahrerassistenzsysteme sind elektronische Zusatzeinrichtungen in Kraftfahrzeugen zur Unterstützung des Fahrers in bestimmten Fahrsituationen. Eine Aufgabe für solche Systeme ist beispielweise die Freiraumbestimmung. Damit erkennt der Autofahrer die vorausliegenden Hindernisse unmittelbar und verringert die Geschwindigkeit des Autos sofort zur Kollisionsvermeidung.

Im Rahmen dieser Arbeit werden am Anfang bestehende Ansätze zur Freiraumerkennung im automobilen Umfeld untersucht. Nach der Wiederholung von Grundlagen wird zwei Algorithmen zur Freiraumbestimmung basierend auf einer zur Verfügung gestellten Occupancy-Gridmap mit Frei-Evidenz in Matlab bzw. C++ entworfen. Zum Schluss werden die entwickelten Methoden mit den vorher gesammelten Messdaten in Framework wxWavE überprüft und eine Zusammenfassung wird ausgegeben.

Schlüsselwörter: Fahrerassistenzsystem, Freiraumbestimmung, Stereovision, Occupancy-Gridmap, Dynamische Programmierung

Notation

Throughout this work, the following notation will be used:

Z	depth
d	disparity in pixels
B	baseline in some metric unit
f	focal length in pixels
r	the range between radar sensor and object
$L_{i,j}$	likelihood function for the measurement of pixel
$D(i,j)$	intensity of an occupancy grid cell
p_l	vehicle coordinate in the local coordinate system
p_g	vehicle coordinate in the global coordinate system
$R(\theta)$	rotation matrix for angle θ
m	translation vector
$P = (X, Y, Z)^T$	world point coordinate
$p = (x, y)^T$	image point coordinate
$V(x)$	value function
$a(x)$	policy function
$T(x_t, a_t)$	state transition function
β	discount factor

Contents

1 Introduction

The first chapter introduces the significance of driver assistance systems and also indicates the intention and organisation of the thesis.

The first section is the motivation. Then the purpose of this work is described in the second section. Lastly, the outline of the thesis is displayed.

1.1 Motivation

Since the Industrial Revolution the transportation, especially steamboatings and railways, had rapidly developed. During the Second Industrial Revolution Karl Benz built the first successful gasoline powered automobile in Mannheim, Germany in 1885 [Ste67]. From then on, the motor vehicle was widely used in the whole world because of its convenience and comfort. Nowadays more and more people are dependent on cars for transportation. It is estimated that more than 70 million motor vehicles, including cars and commercial vehicles were produced worldwide in 2008 [OIC08]. However, the number of the traffic accident increases steeply at the same time. According to the *World Report On Road Traffic Injury Prevention* published by WHO [PS04], worldwide an estimated 1.2 million people were killed in road crashes each year and as many as 50 million were injured in 2004. The report also prognoses that these figures will increase by about 65% over the next 20 years unless there is new commitment to prevention. For that reason people are always trying every means to avoid automobile accidents.

Driving a car is not an easy task. It requires total concentration on road conditions during the journey. Even a perfect driver sometimes makes mistakes which may lead to accidents. Thus, scientists and engineers have been focusing on the improvement of vehicle design, construction, and equipment to minimize the occurrence and consequences of automobile accidents since last century. The development of the mechanical control technology and later the computer science has formed today's Advanced Driver Assistance Systems (or ADAS). The following Table 1.1 shows a timeline of selected safety features and related work.

Time	**Safety Measures**
1902	The speedometer was invented in Germany. [Sie]
1910	The standard electric car horn was invented in England. [Wikc]
1934	The first barrier crash test was performed by GM. [Cra]
1938	The modern turn signal was patented. [Tur]
1958	The seat belt becomes standard. [Sea]
1974	The airbag is made available in production vehicles by GM. [Wikb]
1978	The anti-lock braking system (ABS) was first brought to market by Bosch. [Bos]
1980	The bordcomputer is applied in production vehicles by BMW. [BMW]
1994	The integrated navigation system was available. [BMW]

Table 1.1.: The Timeline of Selected Safety features and Related Work

With the development of technology and the increase of safety awareness, many vehicles are equipped with the driver assistance system. In 2003, the driver assistance systems in each sold vehicle costed about 900 € on average in Germany. Most focused on ABS, ESP, brake assist, tire pressure monitoring, ACC, adaptive light [Wika]. These modern systems have been making an increasing contribution to safety. According to the official report about the trend in the number of people killed in road accidents from

1953 to 2010 published by the Federal Statistical Office of Germany (German: Statistisches Bundesamt, shortly Destatis) in 2011 [FSO], a total of 3,648 people were killed in road traffic accidents in Germany in 2010, which was a decline of 504 persons (or 12%) in 2009. Except for very few years, the number of deaths has continued decreasing since 1972. The chart is shown in Figure 1.1.

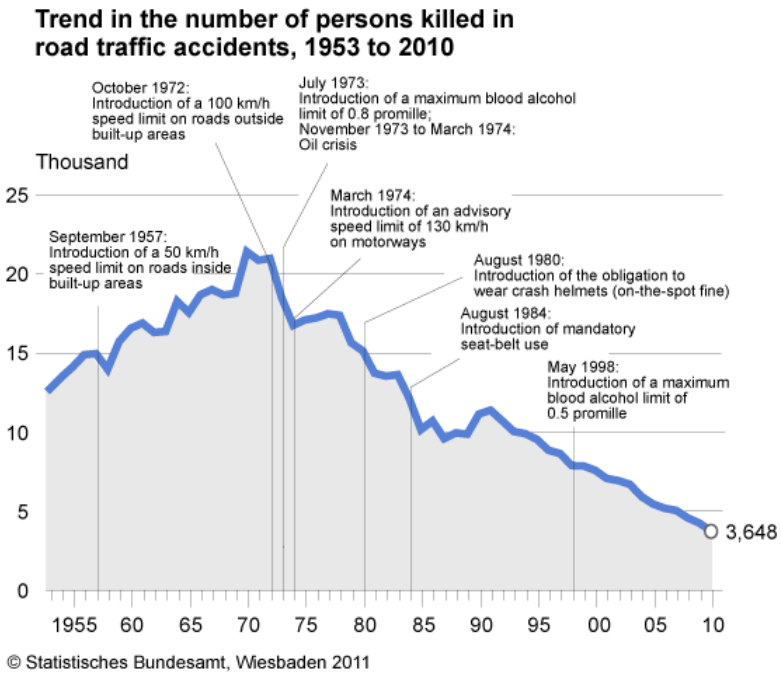


Figure 1.1.: Trend in the number of persons killed in road traffic accidents in 1953 - 2010 [FSO]

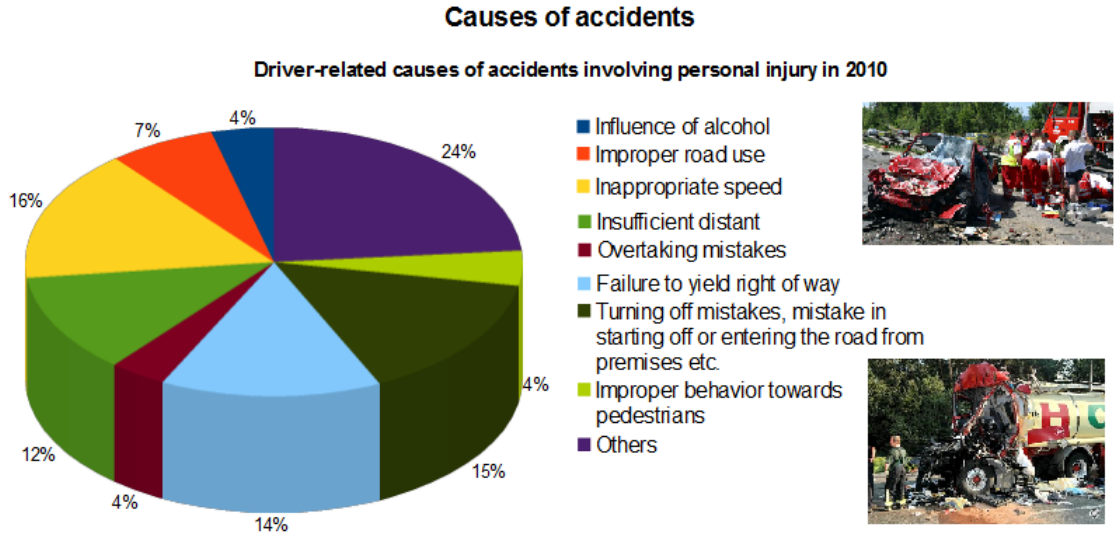


Figure 1.2.: Driver-related causes of accidents involving personal injury in 2010. The data is from the Federal Statistical Office of Germany [FSO].

The application of driver assistance systems may influence the number of death in traffic accidents. Nevertheless, the factors causing accidents shall be never ignored. The diagram *Driver-related causes of accidents involving personal injury in 2010* from the Federal Statistical Office of Germany, which is

illustrated in Figure 1.2, indicates that the main cause recorded for accidents was inappropriate speed with 16%; the second most common cause was mistakes in turning off, turning, reversing, entering and starting with 15%.

This diagram also reveals that 12% of car crashes are caused by the insufficient distance. In most traffic accidents a combination of insufficient headway and unforessen greater stopping distance leads to the driver's failure to stop the car before the collision. The human factors make drivers tend to exhibit an unusual delayed cognition or reaction, which may result in an maintained shorter headway [Mat96]. The intelligent system can overcome the human factors. It continually monitors traffic in realtime, measuring relative speeds and distances between vehicles. In this thesis, a method for analysing the free space in front of the vehicle is presented.

1.2 Intention

Before work the following questions should be taken into consideration:

- Which methods can measure the free space around the vehicle?
- Is there a possibility to analyse the free space optimally?
- What does the system architecture look like for free space computation?

For the first question, there already exist a number of methods for the free space detection: occupancy grids [BFMM07], contrast restoration [HTA09], optical flow [YMOI08], time correlation [CG05] and Gabor Filters [SWTS03]. In this article the free space is based on occupancy gridmaps.

The dynamic programming is a method for solving complex problems by recursively breaking them down into simpler subproblems, which makes the solution optimal. The segmentation between free space and occupancy is also very complex. So this paper tries to find out an algorithm based on the dynamic programming to simplify the segmentation problem, which answers the second question.

The last question suggests, except for developing the algorithm for free space computation, the algorithm design should properly cooperate with other modules in the system, such as interface, input and output data, channel.

1.3 Outline

The whole thesis is divided into the following seven chapters:

- **Chapter 2: Literature Review**
  The literatures about free space computation using occupancy grid are reveiwed. The literatures using other methods are also introduced.
- **Chapter 3: Foundation**
  Some basic theories about stereo camera and radar sensor are reviewed and the occupancy grid for environment modeling cooperates with evidence theory.
- **Chapter 4: Implementation**
  Two algorithms for free space measurement are built.
- **Chapter 5: Experiment Platform**
  The hardware and software of the experiment platform are introduced.
- **Chapter 6: Results**
  The experiment results achieved in the work are summarized.
- **Chapter 7: Discussion**
  A conclusion is reached and future work is presented.

2 Literature Review

The intelligent autonomous vehicle is a challenging research topic. Each year many papers about this topic are presented. Until now, numerous designs or inventions in the field of driver assistance have been equipped in vehicles so as to make our lives safer. The ambitious driver assistance for complex urban scenarios especially demands a complete awareness of the situation, including all moving and stationary objects which limit the free space. In order to avoid collisions, one of the solutions is the free space computation. Its goal is regardes as objects separation from background. Mapping based approaches are frequently adopted in the published papers.

In the following chapter, literatures about occupancy grids are reviewed. At first, occupancy grids are applied in the field of mobile robots planning and navigation. Then their application turns into automotive industry, such as vehicle navigation system. In the chapter, literatures about the application of occupancy grids in free space computation are enumerated. Other methods for free space analysis are also introduced.

2.1 Occupancy Grids in Mobile Robot Domains

Occupancy grids have been used in many different fields. They were first introduced in Literature [Elf87]. In that paper a sonar-based mapping and navigation system is developed for robots in unknown and unstructured environments. A sonar occupancy grid map is generated by storing integrated range measurements from multiple points of view. The measurements saved in each cell in the map are interpreted using probability profiles to determine empty and occupied areas.

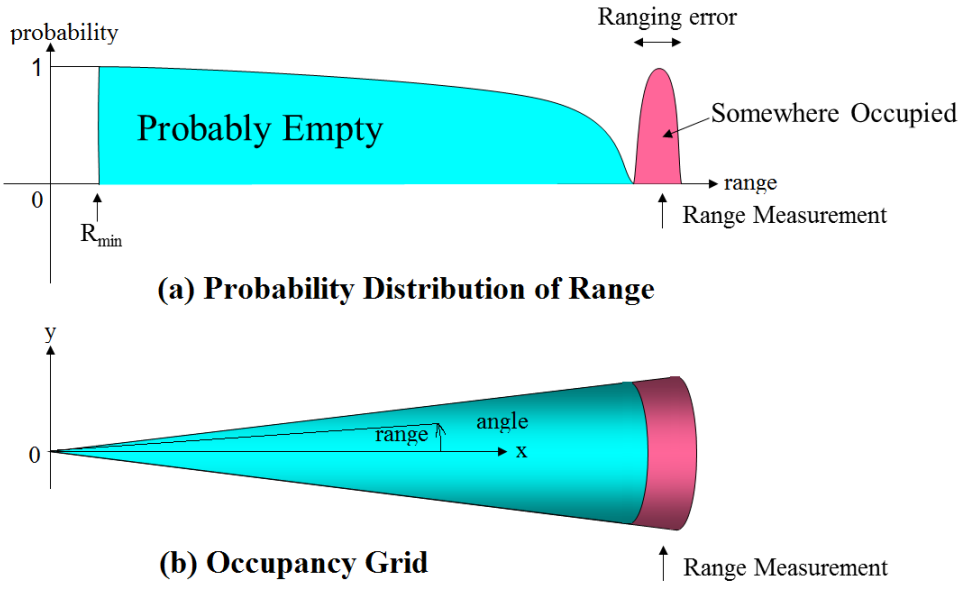


Figure 2.1.: Probability profiles corresponding to Probably Empty and Somewhere Occupied regions in sonar beam. Profiles correspond to horizontal cross section of beam. [Elf87]

According to that paper, the mapping process is explained in detail. The sonar data shall be preprocessed at first. This includes rejecting data below the lower range threshold R_{min}. Then a range reading is used to assert two volumes in 3D space: 'probably empty' and 'somewhere occupied'. The sonar beam

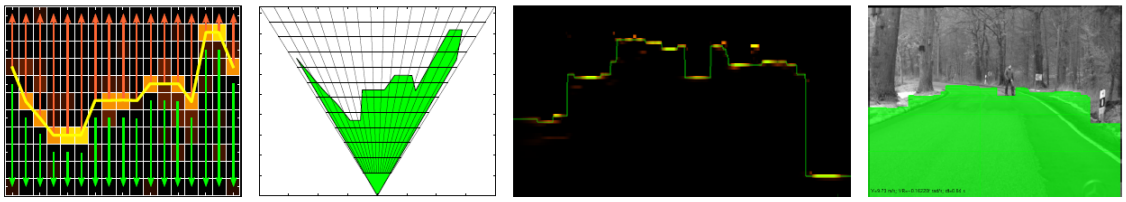
is modeled by two probability density functions, f_E and f_O which are defined over these volumes. Figure 2.1a. shows the empty (in color blue) and occupied (in color pink) probability distributions for the range r. Figure 2.1b. explains that these probability density functions are evaluated for each reading and projected on a horizontal two-dimensional grid. The state of each cell is presented in three ways: unknown 0, empty $[-1, 0)$ or occupied $(0, 1]$. Furthermore, sensor fusion shall be also conducted.

Other literatures, such as Literature [TBF01], present some methods for applying occupancy grid in the field of robot navigation and path planning.

2.2 Occupancy Grids for Free Space Computation

Since many theories and technologies in the field of robot navigation and path planning are similar to those in the automotive industry, a lot of researches try to apply occupancy grid map into vehicle navigation. An example is free space analysis.

Most researches on free space analysis have a common task: constructing a stochastic occupancy grid map in a two-dimensional array, which models occupancy evidence of the environment around the vehicle. For example, occupied cells in the map suggest that these cells could be held by obstacles.



(a) Polar Occupancy Grid. (b) Corresponding free space in world coordinates. (c) Segmentation result. (d) Freespace.

Figure 2.2.: Free Space Search in Polar Representation [BFMM07]

Occupancy grids are applied to analyse free space in real time by Literature [BFMM07], where a novel method for the free space computation based on dynamic programming on a polar occupancy grid is presented. Figure 2.2 shows the free space analysis in a polar representation and the corresponding free space area in the camera image.

Literature [HMMT] explains the process in detail how an occupancy grid is finally obtained. In the beginning the disparity map is produced by the stereo camera. After initialization, this previous disparity map is transformed into the new one according to the depth of the scene and the ego-motion of the camera. That means, there are two disparity maps ideally depicting the scene viewed from the same angle. Then a Kalman filter is used to integrate these two disparity maps into one. In the end a occupancy grid is created based on the filtered disparity maps.

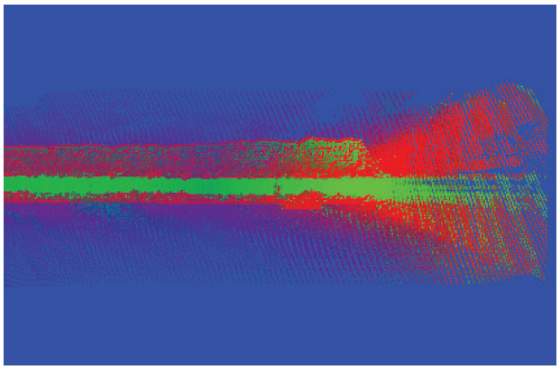
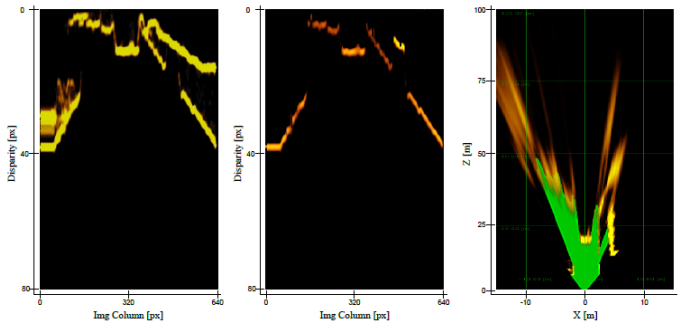


Figure 2.3.: Free Space Analysis using Dempster-Shafer Theory [Eff09]

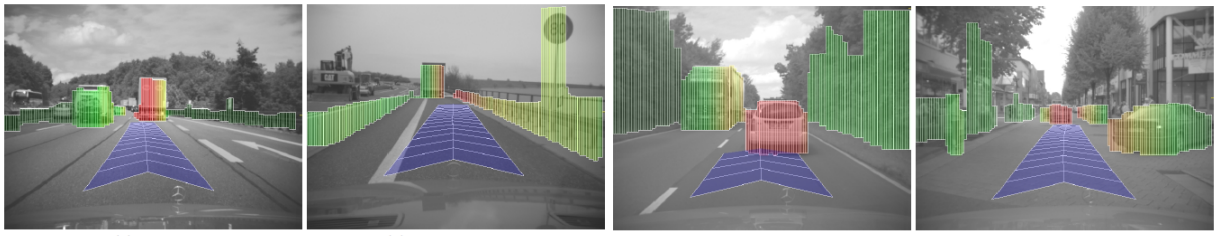
For every grid cell the occupancy probability is estimated by using a binary Bayes filter with static state. With accumulated occupancy likelihood over time, the moving object can be filtered out. But Bayes filter has its own limitation: it can not do with the uncertain measure data. Literature [Eff09] solves this problem by using Dempster-Shafer theory. The Unknownness can also be expressed together with free and occupied cells in the grid map. An example of grid map using evidence theory for free space analysis is shown in Figure 2.3. In the image red spots represent occupancy, green spots represent free space and the blue represents unknownness. Moreover, in other literatures the moving object can be filtered out by being extracted as conflict, since a grid cell is detected as free at one time and as occupied at other time.



(a) Polar occ. grid. (b) Background subtraction. (c) Obtained free space.

Figure 2.4.: Occupancy grids: Figure (a) shows a polar occupancy grid; Figure (b) shows the resulting image when background subtraction is applied to Figure (a); The free space obtained from dynamic programming is shown in Figure (c) in green. [BFP09]

One typical failure case for free space computation using mapping techniques is that static objects are always considered as being within the background and the segmentation path is on the background, which leads to an important issue for driving safety application. A simple way to tackle this problem is the background subtraction, which is explained in Literature [BFP09]. As a preprocessing, this method is carried out before applying dynamic programming. All occupied cells which are above a given threshold are marked as free. The selected threshold shall be quite larger than the occupancy grid noise expected in the grid. An example of background subtraction is shown in Figure 2.4.



(a) Highway (b) Construction site (c) Rural road (d) Urban traffic

Figure 2.5.: Results of Stixels in Different Road Scenarios [BFP09]

Besides, different from Literature [BFMM07], Literature [BFP09] also takes into account that the free space in front of vehicles is limited by objects with almost vertical surfaces, which are approximated by adjacent rectangular sticks of a certain width and height. After free space analysis, a second pass of dynamic programming is applied in order to obtain the upper boundary of the obstacle. Once the free space and the height for every column have been computed, the extraction of the stixel is straightforward. Figure 2.5 displays the results in different road scenarios, such as highway, construction site, rural road and urban environments.

2.3 Other Methods for Free Space Computation

Furthermore, there are several methods without using occupancy grid maps or dynamic programming.

Literature [YMOI08] develops an algorithm detecting the dominant plane as free space from a sequence of omnidirectional images captured by the camera mounted on the autonomous robot. The dominant plane detected from the optical-flow field is the largest planar area in the image. The motion separation property is adopted for the detection of the dominant plane from the optical-flow field.

Another example is the free space computation based on time correlation and inverse perspective mapping (IPM), which is introduced by Literature [CG05]. A new algorithm is built to measure the free space in front of vehicles driving on the highway.

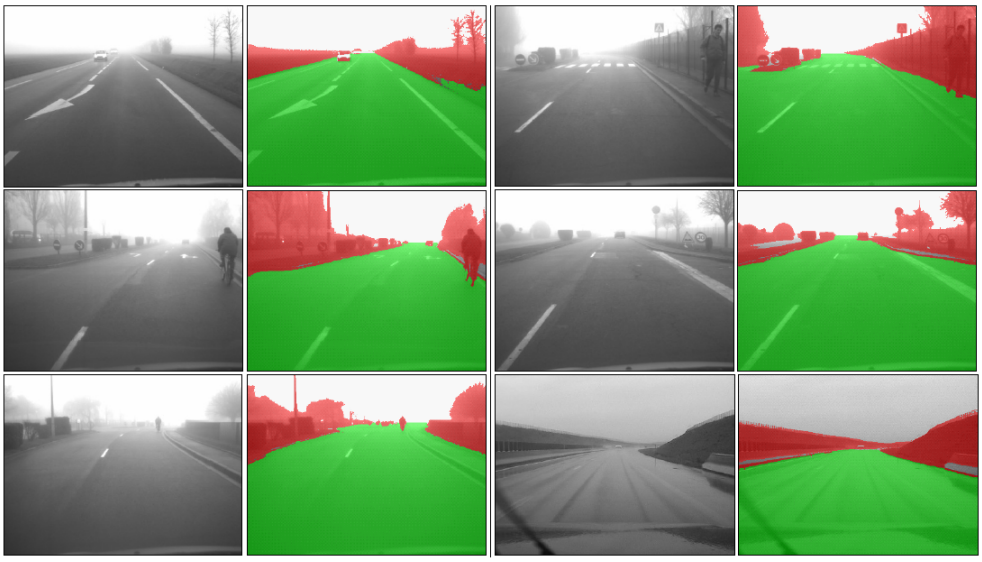


Figure 2.6.: Results of Free Space Detection using Contrast Restoration in Different Road Scenarios [HTA09]

Last but not least, the free space analysis can also be accomplished by using a contrast restoration according to Literature [HTA09]. Since the classical approaches have difficulties in adverse weather conditions, such as fog, a new method based on fog density estimation is built to restore the contrast of the images assuming a flat world. In doing so, the intensity of objects that do not respect this assumption becomes zero, which causes a very efficient segmentation of the free space. The result of this method is shown in Figure 2.6.

3 Foundation

The goal of this work is to develop a suitable algorithm for free space analysis. Before implementing the algorithm, the vehicle shall know about the outside world. With stereo camera, radar or other sensors, the vehicle can perceive the environment. For instance, the distance to an obstacle can be measured by radar sensor. Then these information are utilized to create an environment model which will be regarded as one of the inputed parameters of the algorithm for free space measurement.

In this chapter some basic knowledge is reviewed. First Section 3.1 explains how to get the depth by stereo camera and how to compute the distance between radar sensor and object. Then Section 3.2 presents a widely used environment model for navigation and path planning, i.e. occupancy grid.

3.1 Environment Perception

To perceives the outside environment, a sensor model is necessary for the intelligent system. During the whole work, two sensors are chosen: a stereo camera and a radar sensor. Hence, in this section some theroies of these two sensors are reviewed.

3.1.1 Depth Inference

In order to create an occupancy grid map, it is necessary to determine the world point coordinates. After the disparity and depth information are derived from the images obtained by the stereo camera, the world point coordinate can be acquired through the relation to the image point coordinate.

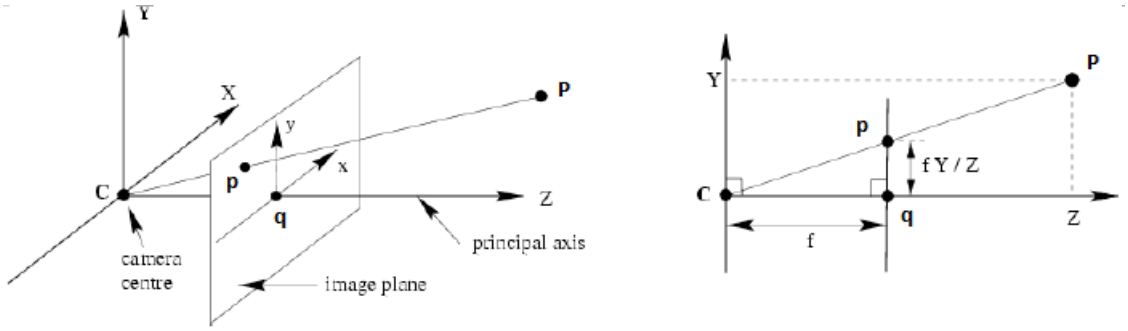


Figure 3.1.: Perspective projection on the x-axis. The y-axis is similar. [Rot11]

Assuming a perfect pinhole camera, world points undergo a perspective projection during image formation. In Figure 3.1 a world point is defined as $P = (X, Y, Z)^T$ and a point on the image plane is defined as $p = (x, y)^T$. According to the geometry, the equations of perspective projections are:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z} \tag{3.1}$$

It should be noted that the image point coordinate depends not only on the world point coordinate $(X, Y)^T$, but also on the depth Z.

To map the pixel coordinate to the world, the depth Z must be known. It is possible to use disparity to infer the depth of the world point. Disparity is the measure of difference in position of a world point in one image compared to another image. Usually, disparity is the displacement in the x direction.

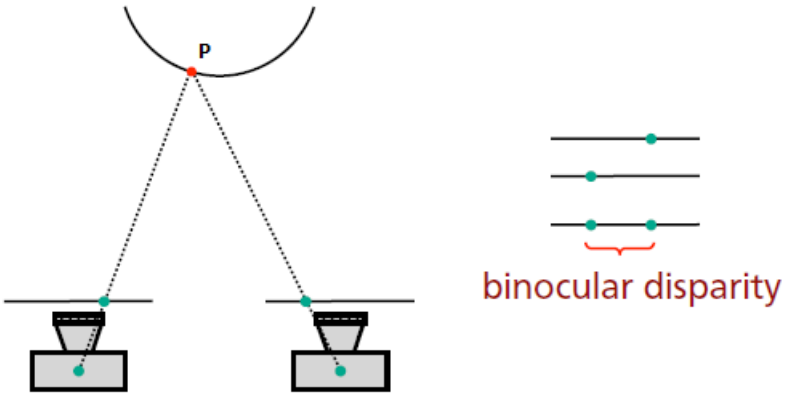


Figure 3.2.: Binocular disparity [Rot11]

In Figure 3.2, the world point P is visible in both left and right images with coordinates $x_{P,left}$ and $x_{P,right}$ respectively. The disparity d_P then becomes

$$d_P = x_{P,left} - x_{P,right} \tag{3.2}$$

There exist some methods for disparity estimation, which depend on the type of camera setup.

After the disparity d is acquired, the depth Z can be estimated from the disparity. Then the world point coordinate can be accomplished using triangulation.

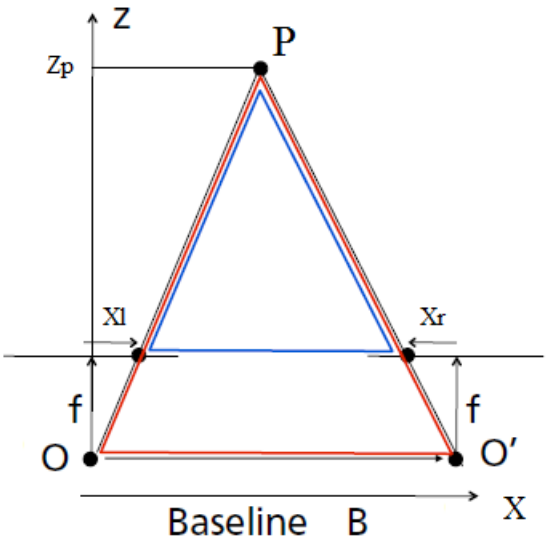


Figure 3.3.: Relation between world points and stereo images. [Rot11]

In Figure 3.3 two cameras with their own coordinate system are placed in Point O and Point O'. The line between OO' is the baseline B and parallel to the x-axis. Here the length of the baseline is also denoted by B. Both cameras have the same focal length f. The world point P is projected to both image planes. But the x coordinates differs between two image planes. x_l is to the right, while x_r is to the

left. The disparity is the difference between these two coordinates, i.e. $d = x_l - x_r$. According to the geometry, the depth Z_P can be acquired by the following equations:

$$\begin{aligned} \frac{B - x_l + x_r}{B} &= \frac{Z_P - f}{Z_P} \\ \frac{B - d}{B} &= \frac{Z_P - f}{Z_P} \\ Z_P &= f\frac{B}{d} \end{aligned} \tag{3.3}$$

A pixel $p = (x, y, d)^T$, where x,y are the image coordinates and d is the disparity, is now mapped into a world point $P = (X, Y, Z)^T$:

$$Z = f\frac{b}{d} \qquad X = Z\frac{x}{f} \qquad Y = Z\frac{y}{f} \tag{3.4}$$

where

b : Baseline in some metric unit, e.g. centimeter
d : Disparity in pixels
f : Focal length in pixels

3.1.2 Distance and Relative Speed Measurement

Radar sensors usually measures distance and relative speed by using Doppler effect. It is helpful to identify dynamic objects in the static map. They transmit electronmagnetic radiation through the antenna. When a signal propagates, objects can reflect, refract or even absorb it. The radiation that returns to the receiving antenna turns to a vector of values representing the received signal amplitude.

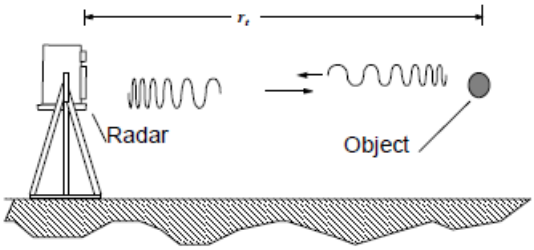
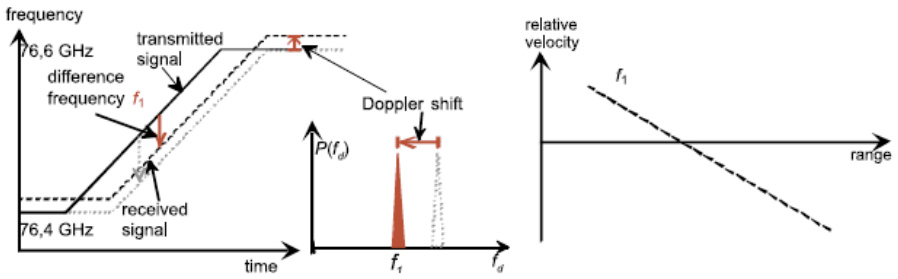


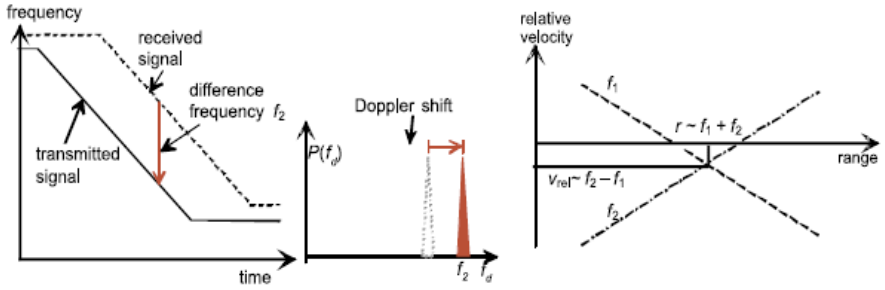
Figure 3.4.: FMCW Radar Applied to Measure Distance and Relative Speed [Foe00]

There exist many kinds of radar sensors. Figure 3.4 presents the frequency modulated continuous wave (FMCW) radar. It measures the range to objects by comparing the frequency of transmitted and received signals. The transceiver sends a signal of increasing frequency. The signal propagates from the antenna to the object and then back. The difference frequency between the received signal and the transmitted signal is proportional to the propagation range.

An analysis of the relation between distance or relative speed measurement and FMCW with a positive or a negative ramp has been already made by Literature [WHW09]. No matter whether the ramp slope is positive or negative, the greater the difference frequency is, the greater the range will be. The analysis is shown in Figure 3.5.



(a) FMCW with Positive Ramp at Approaching Object



(b) FMCW with Negative Ramp at Approaching Object

Figure 3.5.: Analysis of Relation between Distance or Relative Speed Measurement and FMCW with Positive or Negative Ramp [WHW09]

Literature [WHW09] also gives the formulas of distance and relative speed.

$$r = \frac{c}{2} \cdot \frac{\omega_{obj,1} - \omega_{obj,2}}{m_{\omega,1} - m_{\omega,2}} \tag{3.5}$$

$$\dot{r} = \frac{c}{2\omega_0} \cdot \frac{m_{\omega,1}\omega_{obj,2} - m_{\omega,2}\omega_{obj,1}}{m_{\omega,1} - m_{\omega,2}} \tag{3.6}$$

3.2 Occupancy Grid

After being collected by sensors, the measurements are used to set up an environment model. The occupancy grid is usually chosen as the environment model.

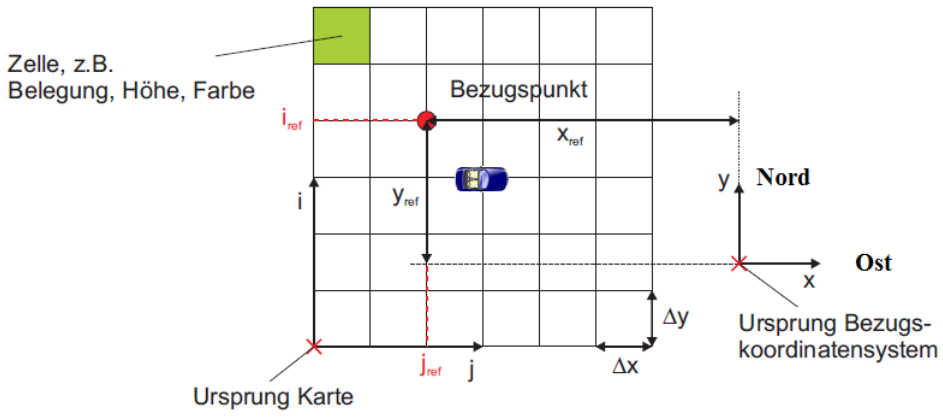


Figure 3.6.: Free Space Search in Polar Representation [Eff09]

An occupancy grid is a two-dimensional array which models occupancy evidence of the environment. So at first all the 3D points shall be projected onto a 2D plane which is parallel to the road. Then the 2D plane is discretized into several cells with the same size. Figure 3.6 shows the definition of an occupancy grid in 2D.

The occupancy grid applied for the environment modeling has the following properties:

- Each cell is independent from the other. The intersection area of any two cells is always empty.
- The cells of the grid are addressed using world coordinates.
- The vehicle is always located in the middle of the grid. Even when the vehicle runs, the grid always moves with it. But the grid moves only in the horizontal or vertical direction.
- For the grid, two coodinate systems are used. One is a continous coordinate system, which uses X as easting and Y as northing. The other is a discrete coordinate system, whose axes are oriented in the same way as the continous coordinate system using i for the X-axis and j for the Y-axis. Both coordinate systems can convert to each other.
- Every cell of the grid maintains an occupancy likelihood of the represented world area. Because of different definitions of the occupancy probability, there exists at least two kinds of occupancy grids applied in the automotive industry. One is Bayesian grid and the other is Dempster-Shafer cell. They are explained in this section.

3.2.1 Bayesian Occupancy Grid

In Bayesian occupancy gird, the environment state in each grid cell (i, j) is modeled by using two states: free and occupied.

$$\begin{aligned} z(i,j) = 0 \quad &: \quad free \\ z(i,j) = 1 \quad &: \quad occupied \end{aligned} \tag{3.7}$$

Since each cell is independent from its neighbors, the probability of free space and occupancy in each cell (i, j) can be deduced.

$$\begin{aligned} P[z(i,j) = free] &= P[\overline{z}(i,j)] \\ P[z(i,j) = occupied] &= P[z(i,j)] \end{aligned} \tag{3.8}$$

The a-posteriori probability distribution of the whole map under the consideration of all measurement data Y at time k is

$$P[Z|Y_k] = P[z(1,1),...,z(M,N)|Y_k] = \prod_{i,j=1}^{M,N} P[z(i,j)|Y_k] \tag{3.9}$$

According to Literature [Eff09], the binary Bayes filter is implemented by using the common log odds ratio l for cell (i, j).

$$l[z_k] = log \frac{P[z_k]}{1 - P[z_k]} \tag{3.10}$$

Then a recursive filter algorithm for updating each cell based on 3.10 is deduced in Literature [Eff09]. It consists of three components: actual measurement reading, a-priori knowledge and the last measurement.

$$l[z_{k+1}] = log \frac{P[z_{k+1}|y_{k+1}]}{1 - P[z_{k+1}|y_{k+1}]} + log \frac{P[\overline{z}_{k+1}]}{1 - P[\overline{z}_{k+1}]} + l[z_k] \tag{3.11}$$

3.2.2 Evidence Theory Based Occupancy Grid

The measurement area of sensors is limited. And some area around vehicles can not be measured by sensors due to technical reason, such as blind spot. In this situation, the Bayesian occupancy grid can not be used any more, because such conflictive, uncertain and incomplete measurements can not be modeled by the Bayesian grid. If the probabilities of these cells are set to 0, then the unknown area including obstacles would be improperly recognized as the free space.

The Dempster-Shafer theory based occupancy grid can solve that problem. In Dempster-Shafer evidential reasoning, a belief mass is assigned to each element of the power set X. For each sensor S_i, the function

$$m_i \quad : \quad 2^X \rightarrow [0,1] \tag{3.12}$$

is called a basic belief assignment (BBA). It has following properties.

$$\forall A \in 2^A, 0 \leq m_i[A] \leq 1 \tag{3.13}$$

$$m_i(\emptyset) = 0 \tag{3.14}$$

$$\sum_{A \in 2^X} m_i[A] = 1 \tag{3.15}$$

Any probability mass that is not assigned to a proper subset of X is included in $m_i(X)$ and represents the residual uncertainty of S_i that is distributed in some unknown manner among its focal elements.

Dempster's rule of combination is used to fuse the propositions A_1 and A_2 from the two sensors S_1 and S_2, which is shown in Equation 3.17. K in Equation 3.16 is a measure of the amount of conflict between the two mass sets.

$$K = \sum_{A_1 \cap A_2 = \emptyset} m_1[A_1] \cdot m_2[A_2] \tag{3.16}$$

$$m_{1,2}[A] = m_1[A] \oplus m_2[A] = (1-K)^{-1} \sum_{A_1 \cap A_2 = A} m_1[A_1] \cdot m_2[A_2] \tag{3.17}$$

The occupancy grid map based evidence theory is applied in the work. Each cell in the map has two values.

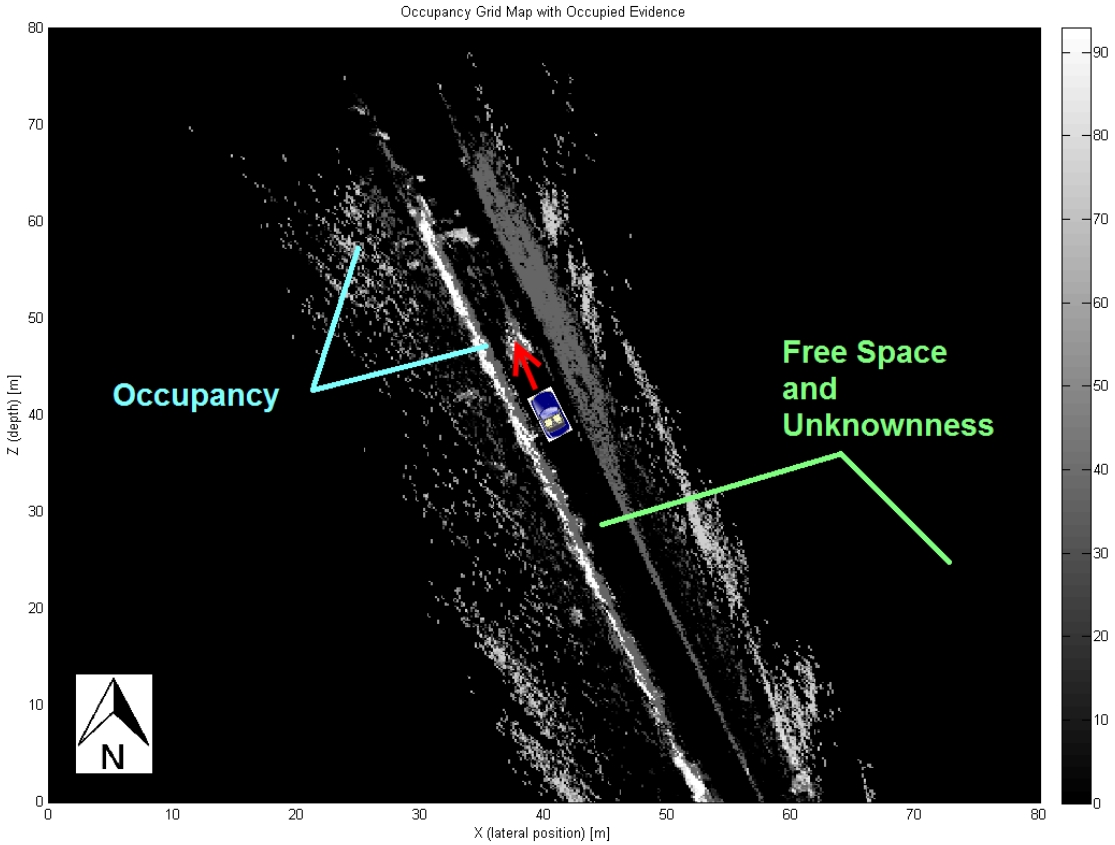
- $m_i[B]$ as a degree of being occupied
- $m_i[F]$ as a degree of being free

The measure value of Unknownness is deduced based Equation 3.15.

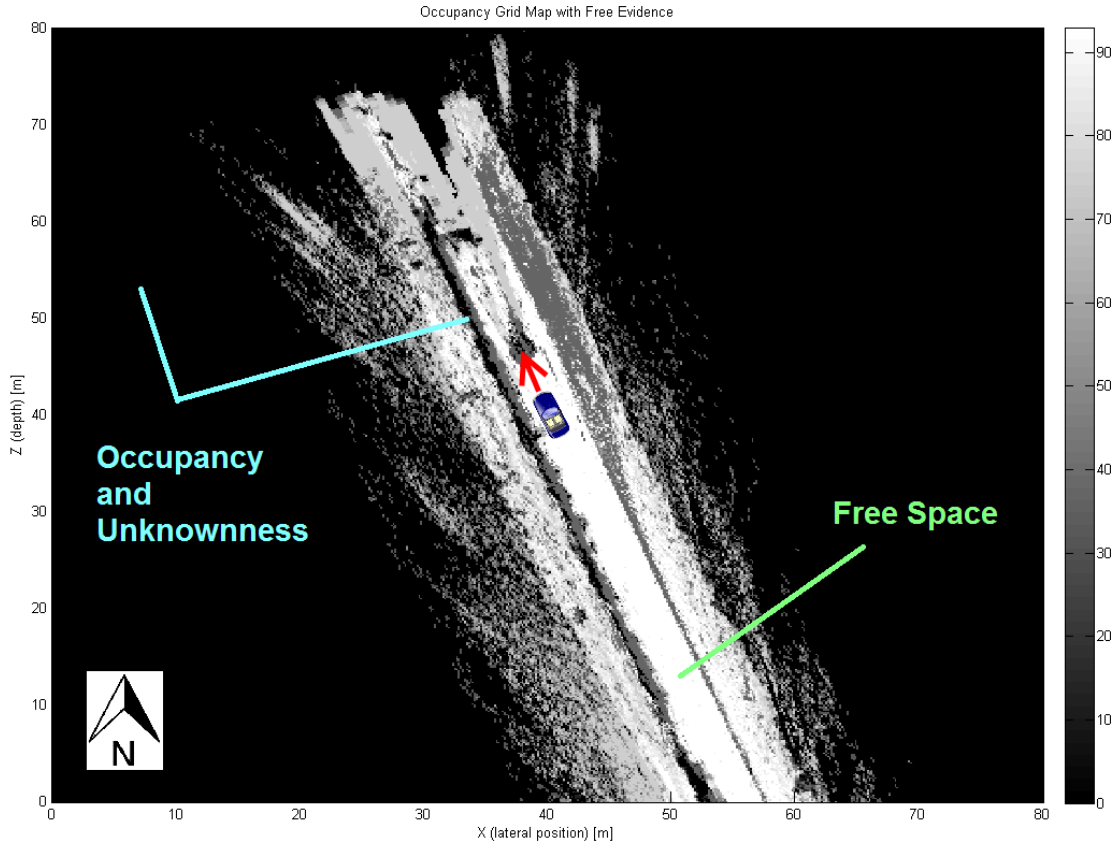
$$m_i[U] = m_i[B \cup F] = 1 - m_i[B] - m_i[F] \tag{3.18}$$

According to Equation 3.16 and 3.17, the mass function m_{k+1} under the consideration of all measurement data Y at time $k+1$ is deduced in Literature [Eff09].

$$\begin{aligned}
m_{k+1}[B] &= \frac{m_k[B] \cdot m_{y_{k+1}}[B] + m_k[U] \cdot m_{y_{k+1}}[B] + m_k[B] \cdot m_{y_{k+1}}[U]}{1 - m_k[B] \cdot m_{y_{k+1}}[F] - m_k[F] \cdot m_{y_{k+1}}[B]} \\
m_{k+1}[F] &= \frac{m_k[F] \cdot m_{y_{k+1}}[F] + m_k[U] \cdot m_{y_{k+1}}[F] + m_k[F] \cdot m_{y_{k+1}}[U]}{1 - m_k[B] \cdot m_{y_{k+1}}[F] - m_k[F] \cdot m_{y_{k+1}}[B]} \\
m_{k+1}[U] &= 1 - m_{k+1}[B] - m_{k+1}[F]
\end{aligned} \tag{3.19}$$



(a) Occupancy Grid Map with Occupied Evidence



(b) Occupancy Grid Map with Free Evidence

Figure 3.7.: Occupancy Grid Map with Dempster-Shafer Evidence

The occupancy grid map with occupied and free evidence are respectively illustrated in Figure 3.7a and 3.7b. In both maps two axes are parallel to the world coordinate system. The horizontal axis points to the east and the vertical axis points to the north. Each side of the map is 80 meter long. The vehicle is fixed in the middle. The color bars which represent the probability in different color stand on the right side of each map.

The occupancy grid map with occupied evidence shows the probability distribution of occupied cells. The higher the probability is, the more certain it is that the cell is occupied. The lower probability indicates that the cell can be free or unknown.

On the contrary, in the occupancy grid map with free evidence, the higher the likelihood of the cell is, the more likely the cell is to be free. The lower likelihood represents that the cell can be occupied or unknown.

Since the map with free evidence can separate free space from obstacles and unknown environment, this kind of occupancy grid is used for free space analysis during the whole work.

3.2.3 Representation Types

After a disparity map of the scene is produced by the stereo camera, a large number of the world points are available. An occupancy grid is a binning of each world point to a corresponding position in the occupancy grid. The intensity of each cell is regarded as the number of world points falling into the bin.

For each cell $D(i, j)$ in the occupancy grid, the intensity is

$$D(i,j) = \sum_{k=1}^{m} L_{i,j}(P_k) \tag{3.20}$$

where m is the number of data points available, p is a pixel with associated disparity and $L_{i,j}$ is the likelihood function mapping the data point to the cell (i,j).

There are three representation types for occupancy grids: cartesian occupancy grid, column/disparity map and polar occupancy grid.

In a cartesian occupancy grid, the axes are world coordinates. The columns represent the lateral position and the rows represent the depth.

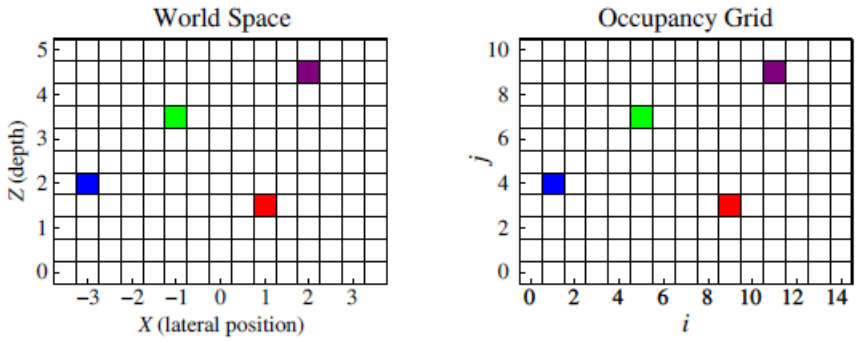


Figure 3.8.: Scene in world space and resulting Cartesian occupancy grid [HMMT] [BFMM07]

Figure 3.9 shows the column/disparity map. The columns correspond to the columns of the disparity map, and the rows represent disparity. Hence, it preserves the perspective projection in an image.

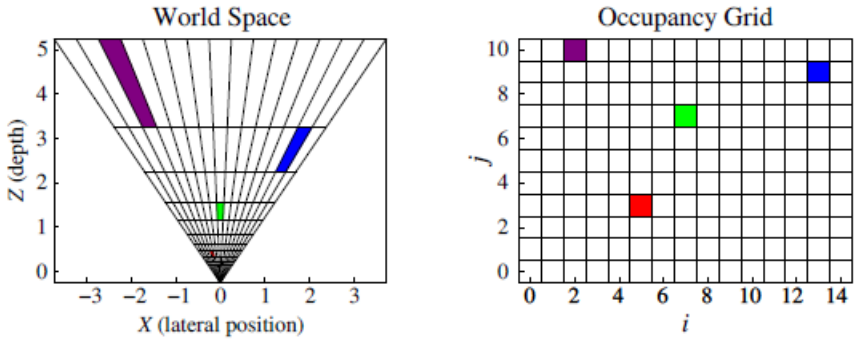


Figure 3.9.: Scene in world space as seen by the camera, and resulting column/disparity occupancy grid [HMMT] [BFMM07]

The third occupancy grid is the polar occupancy grid. As with the column/disparity version, the columns correspond to the columns of the disparity map, but instead of disparity, the rows now represent depth. This results in an even distribution of depth along the rows, consequently yielding lower resolution at close proximity but finer resolution at large depths. shows an example.

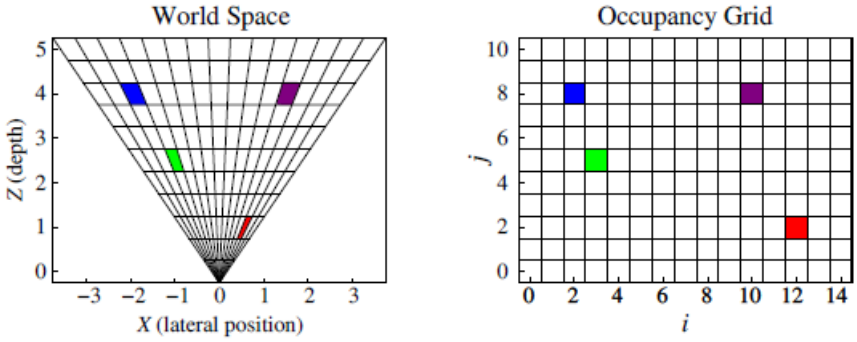


Figure 3.10.: Scene in world space as seen by the camera, and resulting polar occupancy grid [HMMT] [BFMM07]

3.3 Coordinate Systems and Transformation

Coordinate systems are used to uniquely determine the position of a point or other geometric elements, after the coordinate systems and their relation have been defined. All coordinate systems must observe the right-hand rule. In the coordinate system the vehicle is often modelled as a mass point.

There are two kinds of coordinate systems, i.e. global and local coordinate systems. The global coordinate system is fixed and is used to watch the vehicle motion. The vehicle has its own local coordinate system. It is combined with the vehicle. Figure 3.11 provides an overview:

- The blue circle represents the moving vehicle and M indicates the center of the vehicle.
- xOy is the global world coordinate system. It is fixed relative to the vehicle coordinate system.
- $x'My'$ is the local vehicle coordinate system. It is fixed at the vehicle and thus moves relative to the world coordinate system when the vehicle moves.

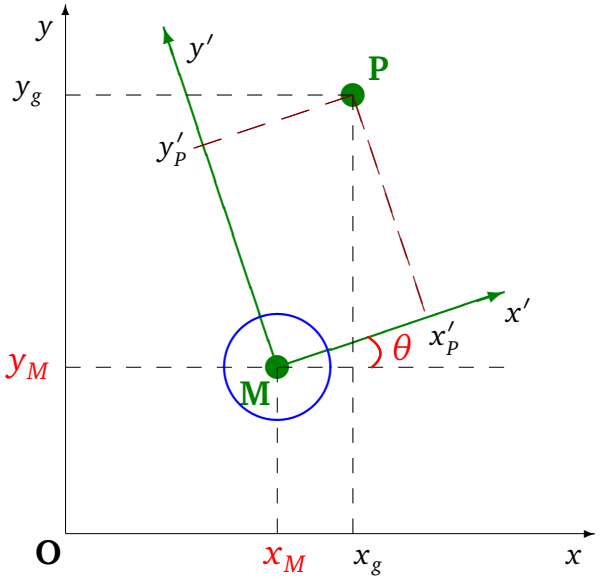


Figure 3.11.: Overview of the different coordinate systems

The vehicle position p in Figure 3.11 is determined by the coordinate system M in the global coordinate system and the angle θ between the local x'-axis and the global x-axis.

$$p = (x_M, y_M, \theta)^T \tag{3.21}$$

The point P is measured in the vehicle coordinate system as vector p_l

$$p_l = (x_l, y_l)^T \tag{3.22}$$

and its coordinate in the world coordinate system is p_g

$$p_g = (x_g, y_g)^T \tag{3.23}$$

The transformation from p_l to p_g is:

$$\begin{aligned} p_g &= R(\theta)p_l + m \\ p_g &= \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} p_l + \begin{bmatrix} x_M \\ y_M \end{bmatrix} \\ \begin{bmatrix} x_g \\ y_g \end{bmatrix} &= \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x_l \\ y_l \end{bmatrix} + \begin{bmatrix} x_M \\ y_M \end{bmatrix} \\ \begin{bmatrix} x_g \\ y_g \end{bmatrix} &= \begin{bmatrix} x_l cos(\theta) - y_l sin(\theta) + x_M \\ x_l sin(\theta) + y_l cos(\theta) + y_M \end{bmatrix} \end{aligned} \tag{3.24}$$

where

$R(\theta) = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$: Rotation matrix

$m = \begin{pmatrix} x_M \\ y_M \end{pmatrix}$: Translation vector

Furthermore, transformations are generally written as homogeneous transformations for clarity.

4 Implementation

In this chapter two algorithms for free space analysis are developed. One is the thresholding operation, the other is dynamic programming. The process for these two algorithms is: transformation, polar representation and segmentation.

4.1 Transformation

In Chapter 2, it is mentioned that the grid map moves with the vehicle only in the geographical direction north-south or west-east. The axis X is used as easting in the world coordinate system, while the axis Y is as northing. The grid map can not be rotated, but the vehicle can run in any direction at any time. The likelihood of obstacles or unknownness in the driving direction shall be collected so that the free space can be analyzed according to those information. Thus, in the beginning of the program, the world coordinate system shall be transformed into the vehicle coordinate system.

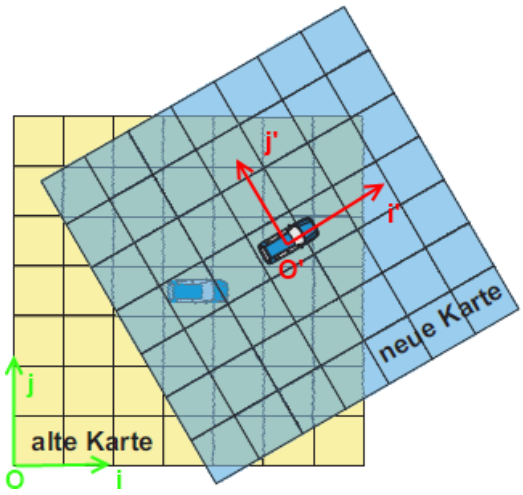


Figure 4.1.: Transformation from World to Vehicle Coordinate System [Eff09]

Here all occupancy grids in the map are rotated around the vehicle by an angle Ψ. The new coordinate of occupancy grids can be acquired through the following equation:

$$p_{new} = R(\Psi)p_{alt} \tag{4.1}$$

, where $R(\Psi)$ is the rotation matrix

$$R(\Psi) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

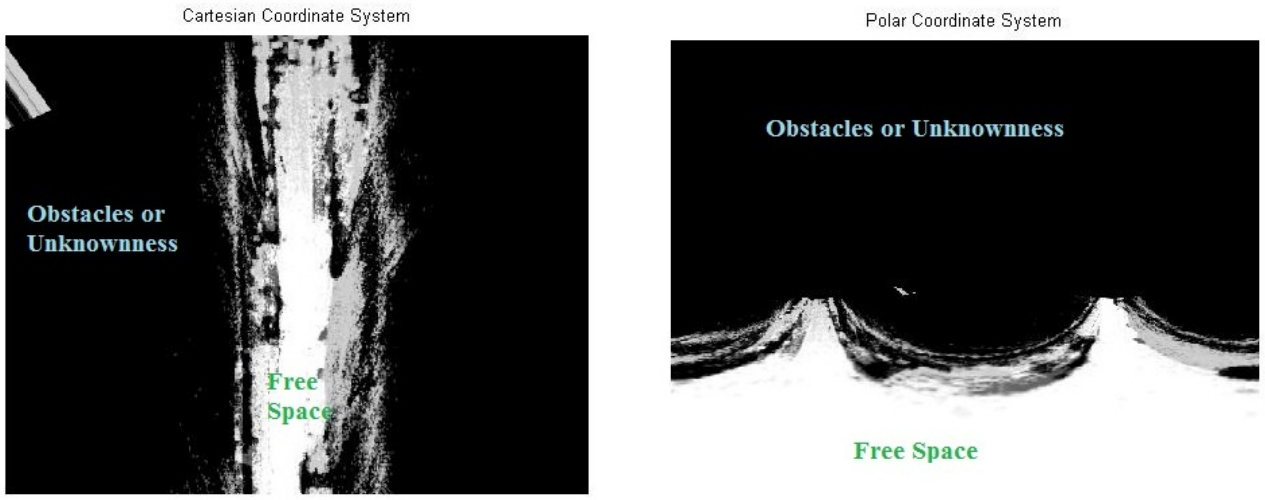
4.2 Polar Representation

In order to measure the free space around the vehicle, it had better applicate polar representation. The polar coordinate system takes its inspiration from the human visual system. The eye is a fixed point lying

in the center and the rays come from the eye in any direction. The observed things are located on the circles with different radii around the eye. Their positions are described with an angular coordinate and a radius.

Here the vehicle takes the place of the eye, and each occupancy grid represents discretized values of coordinates (u, z), where u corresponds to the angle relative to the horizontal in the vehicle coordinate system with the range from 0 to 2π, and z the depth.

The main advantage of polar girds is that the free space computation can become very efficiently. Before free space computation, the vehicle coordinate system shall be transformed into the polar coordinate system. Figure 4.2 compares a cartesian to a polar coordinate system. In both images the white color represents free space, while the black color obstacles or unknownness. On the left side, Figure 4.2a shows a cartesian coordinate system. Both horizontal and vertical axes indicate the distance in two directions. The free space is dispersed in the middle of the map with obstacles and unknownness, and its form is very complex, which makes the free space computation very difficult. On the right side, Figure 4.2b displays the polar coordinate system. The horizontal axis offers the angle range from 0° to 360° and the vertical corresponds the depth. The free space concentrates in the low part of the map and distinguishes itself from obstacles and unknownness. The computation turns to the segmentation between free space and obstacles or unknownness. The polar space is more appropriate than the cartesian space, because the set of rays leave the camera (or the vehicle) and span the whole grid. Furthermore, the search for the visible relevant obstacle must be conducted in the direction of rays. Every grid column in the polar representation is already in the direction of rays. Therefore, searching for obstacles is straightforward.



(a) Cartesian Coordinate System **(b)** Polar Coordinate System

Figure 4.2.: Comparison between Cartesian and Polar Coordinate System

Moreover, during the polar transformation some data could be lost. The solution to avoid this problem is to use interpolation methods. The OpenCV function cvLogPolar remaps an image to log-polar space, and it also provides a flag CV_WARP_FILL_OUTLIERS for the interpolation method to fill all of the destination image pixels against data loss.

4.3 Segmentation

The position of the nearest obstacles shall be measured so that the free space around the vehicle can be analysed. In the polar representation, the task is to find a suitable occupied cell in the positive direction of depth. But each occupied cell in the occupancy grid map owns a value which indicates the probability of that cell being occupied. Some cells with a high likelihood actually represent obstacles, such as other

vehicles, pedestrians or even buildings, others with a low likelihood are just noise. Now the question is how to choose a 'suitable' occupied grid according to the likelihood.

There exist many different ways of segmentation. In this section two simple methods thresholding and dynamic programming are implemented.

4.3.1 Segmentation by Thresholding

According to Literature [HMMT], with thresholding, a inputed threshold determines the segmentation between obstacles and free space. Each column of the occupancy grid is considered individually and the search in different columns is independent from each other. The search traverses from near to far in the positive direction of depth and stops when the likelihood of a cell reaches the threshold. The depth of that cells is the maximum depth free of obstacles.

Figure 4.3.: Segmentation Method - Thresholding

Figure 4.3 interpretes the process of the thresholding operation. In this example the threshold is set to 11. The search begins from the bottom cell in the first column of the left side and runs in the vertical direction. If the cell with the likelihood larger than 11 is found, this cell belongs to the free space. Otherwise, the search in this column is finished and the new search starts in the next right column right now.

The algorithm of the thresholding operation is shown in Algorithm 1.

Algorithm 1 Thresholding $(value)$

```
1: THRESHOLD ← value
2: for col = 1 to Number of Columns do
3:    for row = 1 to Number of Rows do
4:       if Gridmap(row, col) ≥ THRESHOLD then
5:          Freespace(row, col) ← 1
6:       else
7:          break
8:       end if
9:    end for
10: end for
```

The thresholding operation is easy to design and implements very fast. But the segmentation in each column is considered independent from others. It leads to the solution without spatial or temporal smoothness and non-global optimization. Besides, the segmentation result highly depends on the inputed threshold.

4.3.2 Segmentation by Dynamic Programming

The thresholding method can not make sure that the segmentation result is an optimal solution. If there is a method which considers the segmentation in different columns together, an optimal solution may be produced and this method can be applicated widely.

Literature [BFMM07] introduces a new method, i.e. dynamic programming (DP), which can find the optimal segmentation. As proposed in [BFMM07], it has the following properties.

- **Global optimization**: every row is not considered independently, but as a part of a global optimization problem that is optimally solved.
- **Spatial and temporal smoothness of the solution**: the spatial smoothness is imposed by the use of a cost that penalizes jumps in depth while temporal smoothness is imposed by a cost that penalizes the deviation of the current solution from a prediction. The prediction is obtained from the segmentation result of the previous cycle.
- **Preservation of spatial and temporal discontinuities**: the truncation of the spatial and temporal costs allows the preservation of discontinuities.

The dynamic programming operation can be described by using the mathematic expression. According to [BFMM07], the objective is to find the minimal path. The cost of the edge connecting two occupied cells (i, j) and (k, l) is expressed by

$$c_{i,j,k,l} = E_d(i, j) + E_s(i, j, k, l) \tag{4.3}$$

where

$$E_d(i, j) = \frac{1}{D(i, j)} \tag{4.4}$$

is the data term defined by the inverse likelihood of the cell (i, j), and

$$E_s(i, j, k, l) = S(j, l) + T(i, j) \tag{4.5}$$

is a smoothness term composed of a spatial and a temporal part. The spatial term penalizes jumps in depth and is defined as:

$$\begin{array}{lll} S(j,l) = C_s d(j,l) & ; & \text{if } d(j,l) < T_s \\ S(j,l) = C_s T_s & ; & \text{if } d(j,l) > T_s \end{array} \tag{4.6}$$

The function $d(j, l)$ returns the distance in meters between cells in rows j and l. The constant C_s is a cost parameter penalizing jumps in depth, and the threshold T_s saturates the cost function, allowing the preservation of depth discontinuities. The temporal term $T(i, j)$ has the same form.

$$\begin{array}{lll} T(i,j) = C_t d(j,j') & ; & \text{if } d(j,j') < T_t \\ T(i,j) = C_t T_t & ; & \text{if } d(j,j') > T_t \end{array} \tag{4.7}$$

where C_t is the cost parameter, T_t is the maximal distance for the saturation, and j' is the prediction obtained by the previous cycle.

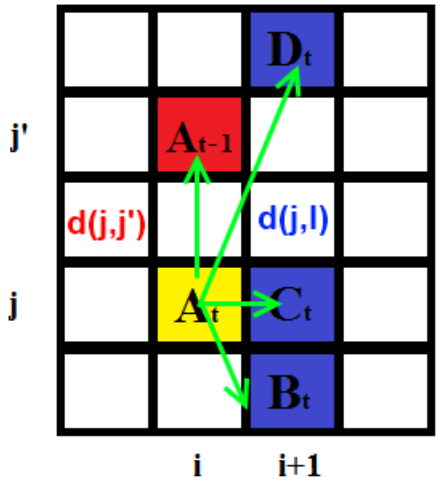


Figure 4.4.: Spatial and Temporal Distances in Dynamic Programming (DP)

For a better understanding of the spatial and temporal distances in the dynamic programming operation is therefore, Figure 4.4 is displayed above. In this example the yellow cell A (at the i-th column and the j-th row) is the start point at the t time. There exist three possibilities for the connection with A in the next right column $i+1$, which are filled in color blue: B, C and D. Here it is obviously that the spatial distance $d(j,l)$ between cell C and cell A in the positive direction of depth equals to zero, because both cells are in the same rows. Besides, the red cell (at the i-th column and the j'-th row) shall be also considered, which is the result of A in the previous cycle at $t-1$ time. The temporal distance $d(j,j')$ returns the deviation between row j and row j'.

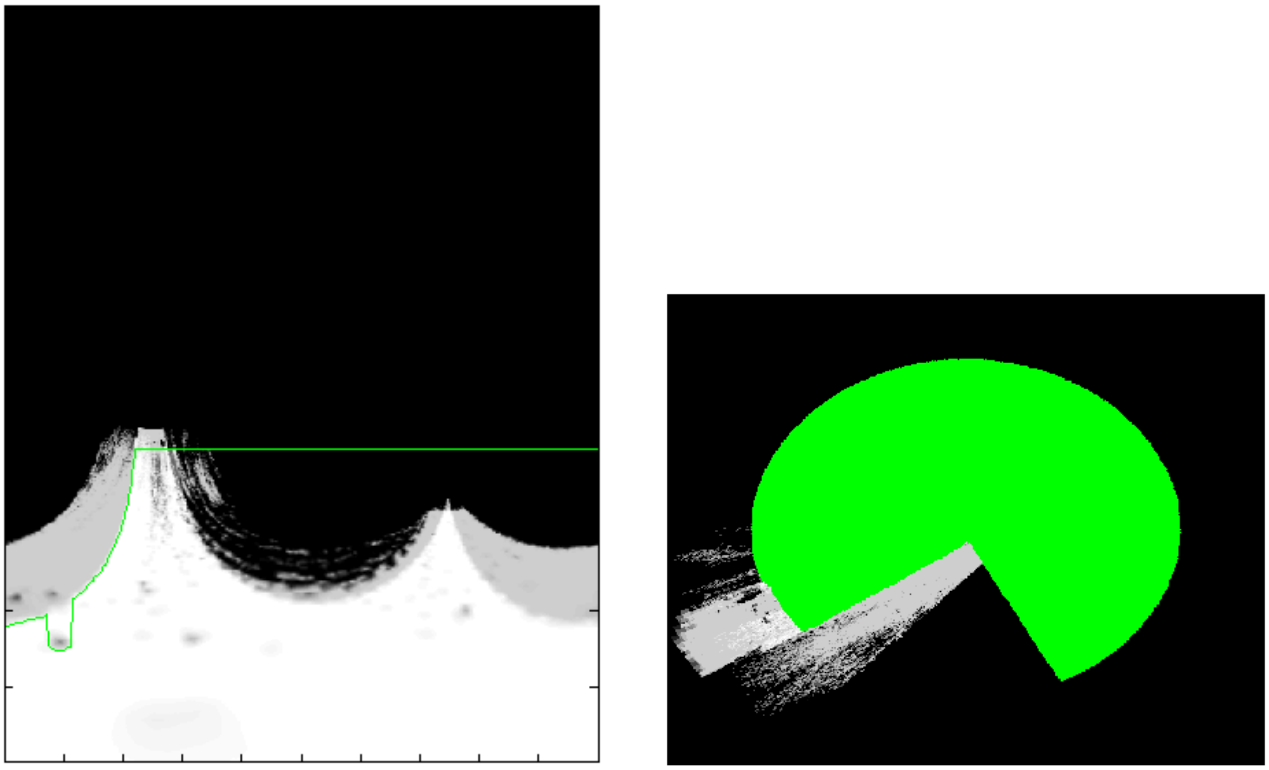
The algorithm of the dynamic programming operation is shown in Algorithm 2.

Algorithm 2 Dynamic Programming

Cell (i,j) is the start point for searching occupied cells in column k. (Here $k = i+1$)
The n possible connection points are in column k: $(k,l_1),(k,l_2)...(k,l_n)$
$E_d(i,j) \leftarrow \frac{1}{D(i,j)}$
if $d(i,j) < T_s$ **then**
    $S(j,l_p) \leftarrow C_s d(j,l_p) \qquad \forall p = 1,2,...,n$
else
    $S(j,l_p) \leftarrow C_s T_s \qquad \forall p = 1,2,...,n$
end if
if $d(j,j') < T_t$ **then**
    $T(i,j) \leftarrow C_t d(j,j')$
else
    $T(i,j) \leftarrow C_t T_t$
end if
$E_s(i,j,k,l_p) = S(j,l_p) + T(i,j) \qquad \forall p = 1,2,...,n$
$c_{i,j,k,l_p} = E_d(i,j) + E_s(i,j,k,l_p) \qquad \forall p = 1,2,...,n$
Select the minimum cost $c_{i,j,k,l}$ from $c_{i,j,k,l_p} \forall p = 1,2,...,n$
Cell (k,l) turns to the next start point for searching occupied cells in column $k+1$.

In most cases, a column in occupancy grid maps may contain likelihoods of background and forground objects, or even unknownness (when Dempster-Shafer cells are used). With dynamic programming, the segmentation path shall go through foreground objects. Due to the property of dynamic programming: spatial and temporal smoothness of the solution, the optimal boundary is likely to be found not on the forground object, but on the background object. This leads to a segmentation failure, and some

area occupied by obstacles would be regarded as free space. Figure 4.5a shows the optimal path (i.e. the green line) cuts through the background object (i.e. the black part), and the wrong segmentation produces the wrong free space analysis which is displayed in Figure 4.5b.



(a) Segmentation Failure: Optimal Boundary on Background Object

(b) Wrong Free Space Analysis

Figure 4.5.: Dynamic Programming without Preprocessing

To cope with the problem above, Literature [BFP09] gives a suggustion: a background subtraction is carried out before dynamic programming. All occupied cells with likelihoods above a given threshold are marked as free. The threshold must be correctedly selected so that the rest occupied cells only represent foreground objects.

The algorithm of the thresholding operation is shown in Algorithm 3.

Algorithm 3 Preprocessing $(value)$

```
1:  THRESHOLD ← value
2:  for col = 1 to Number of Columns do
3:    if Gridmap(1, col) = 0 then
4:      for row = 1 to Number of Rows do
5:        if Gridmap(row, col) > THRESHOLD then
6:          Gridmap(row, col) ← 0
7:        end if
8:      end for
9:    end if
10: end for
```

4.4 Summary

In this chapter the whole free space computation process has been presented. The available occupancy grid map is transformed from the world coordinate system into the vehicle coordinate system. Then it turns to the polar representation in order for the free space analysis. There are many ways for free space measurement. Two methods - thresholding and dynamic programming are illustrated in this paper. At last, the free space is colorized in the original world cartesian coordinate system.

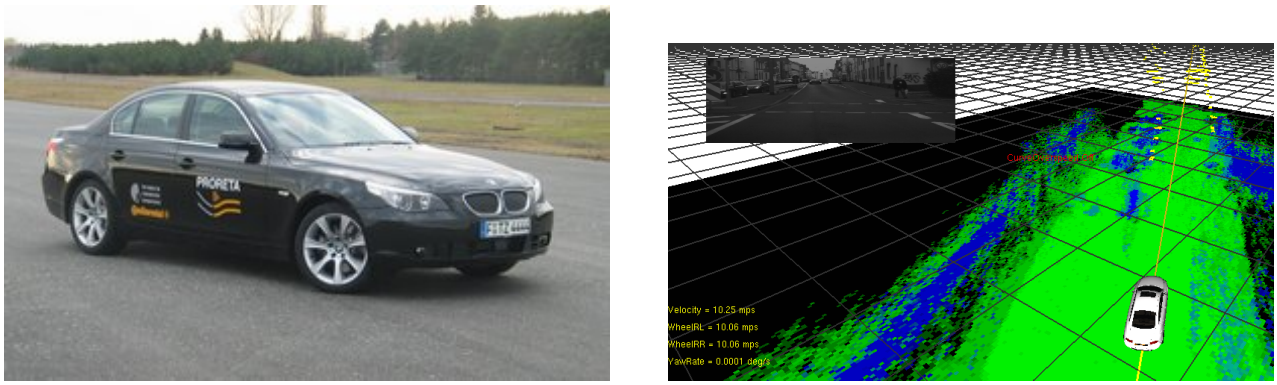
The differences between thresholding and dynamic programming operations are concluded in the table below.

Table 4.1.: Differences between Thresholding and Dynamic Programming

	Thresholding	Dynamic Programming (DP)
Algorithm Design	Each column of the occupancy grid is considered individually and the segmentation in different columns is independent from each other.	Every row is not considered independently and a global optimal segmentation is found.
Segmentation Line	rough	spatially and temporal smooth
Implementation	very fast	slower
Application	The application result is highly dependent on the threshold.	DP can be applied in a wide range of scenes.
Preprocessing	needless	necessary

5 Experiment Platform

The vehicle used for testing and evaluation of this work is displayed in Figure 5.1a. This car is equipped with a stereo camera and a radar sensor. They are used to collect the measurements during the test route. After that these measurements together with the developed algorithms in Chapter 4 are adopted to visualize the free space in the framework wxWavE, which OpenSplice supplies with the real-time data distribution service. Figure 5.1b is a screenshot of wxWavE.



(a) Hardware: Test Vehicle [tes] **(b)** Software: Framework wxWavE

Figure 5.1.: Hardware and Software of Experiment Platform

In the beginning of the chapter, some technique information about two applied sensors, stereo camera CSF 200 and radar sensors ARS 300 is presented in Section 5.1. Then the framework wxWavE and its data distribution service provider OpenSplice are introduced in Section 5.2. In the end of the chapter, the module structure of the developed algorithms for free space computation in Chapter 4 is illustrated in Section 5.3.

5.1 Sensors

The test vehicle is equipped with stereo camera CSF 200 and radar sensor ARS 300, which are displayed in Figure 5.2.

(a) Stereo Camera CSF 200 [csf] **(b)** Radar ARS 300 [ars]

Figure 5.2.: Sensors Equipped on Test Vehicle

From Figure 5.3, we can see that the stereo camera (in red color) is set up on the top of the test vehicle, while the radar sensor (in green color) is installed on the bottom.

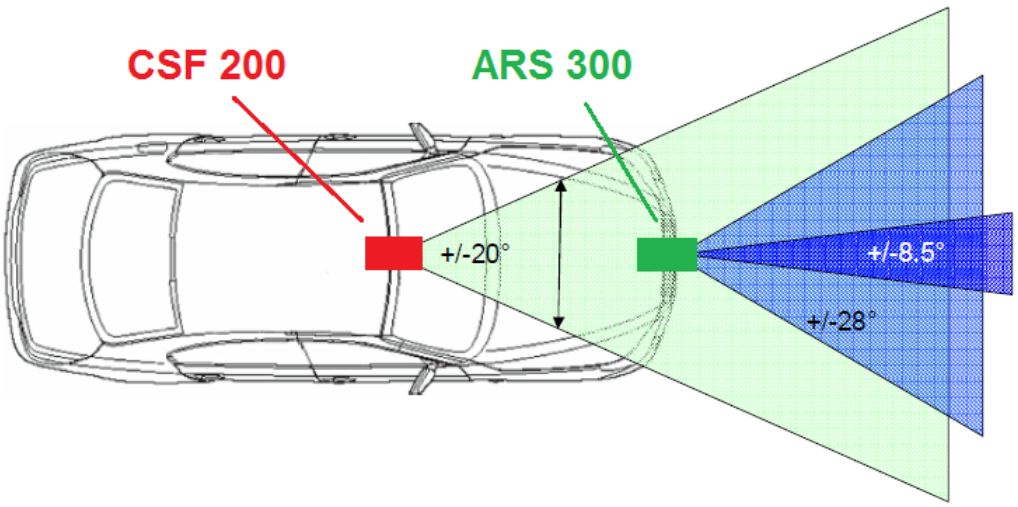


Figure 5.3.: Measurement Area of Stereo Camera CSF 200 and Radar Sensor ARS 300 [Rie10]

CSF 200 (in Figure 5.2a) is a mono CMOS camera. It has a field of view (FOV) of up to 50 meters in the low light or daylight, with a lateral FOV of 20°. A RGB-image of $752x480$ pixels is generated for advanced processing. CSF 200 has been applied in Continental's lane-departure warning (LDW) and lane-keeping system (LKS). The image-processing software calculates the position of the vehicle within the lane markings. The camera sees ahead and issues a warning before the vehicle starts to cross the line.

ARS 300 (in Figure 5.2b) is a radar sensor produced by Continental AG. It applies Scanning principle in the angle measurement and Doppler's principle in the distance and velocity measurement without reflector in one measuring cycle due basis of Frequency Modulated Continuous Wave (FMCW). A special feature of the device is the simultaneously measurement of great distances up to 200 meters, relative velocity and the angle relation between two objects. Moreover, ARS 300 has many advantages: its design is kept robust and small; it dispels with the apparent contradiction between excellent great measuring performance and a high degree of operational safety; it is fail-safe and able to recognize troubles of the sensor and sensor environment, and display it automatically.

The configuration of both sensors is concluded in Figure 5.4.

Sensor	ARS300 77 GHz Radar 	CSF200 Mono CMOS camera (color) 
Data base for fusion algorithm	Object list	RGB 752 x 480 pixel
Information / quality Distance Relative speed Lateral position Classification	 high high medium medium	 low low high high
Measurement area	± 28° up to 60m ± 8,5° up to 200m	± 20° up to 50m

Figure 5.4.: Fusion of Beam (Radar) and Image (Camera) [Rie10]

5.2 Framework

Before implementing visualization in wxWavE, a specified data distribution environment shall be set up. In this environment, a large amount of data is transferred in real time and the software module must immediately make reaction, which fits the requirment of in-car softwares. Here, OpenSplice offers the real-time data distribution service.

5.2.1 OpenSplice

OpenSplice is an open source implementation of the object management group's (OMG) data distribution service for real-time systems (DDS). OpenSplice DDS leaps forward in publish/subscribe middleware and is carefully specified to allow very high performance, scalable, predictable and high-availabilitiy implementations. The DDS is the only technology that spans across the board. It guarantees exceptional real-time behavior, while providing unparalleled level of throughput.

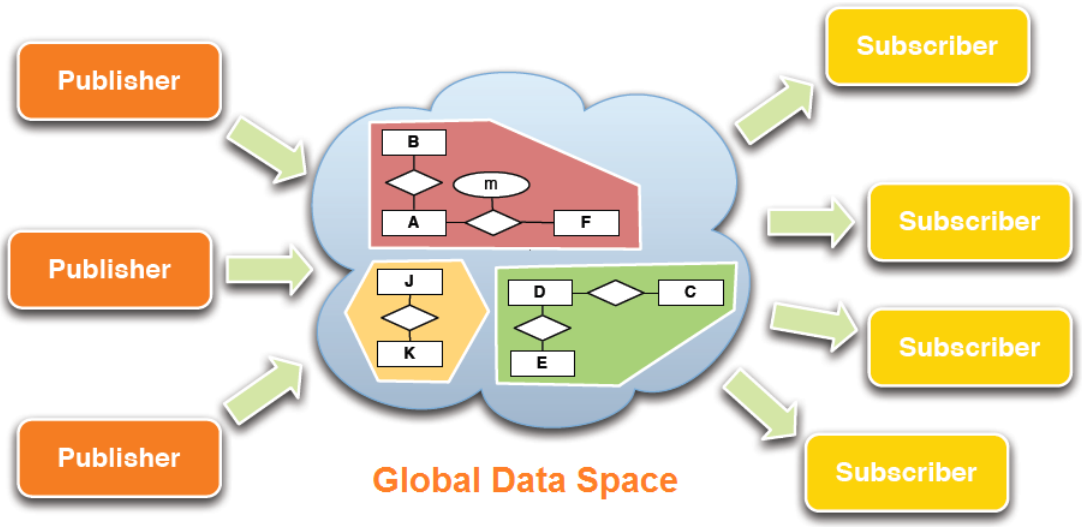


Figure 5.5.: Data-Centric Publish Subscribe (DCPS) Layer [Cor08]

OpenSplice DDS Community Edition provides a fully compliant implementation of the data-centric publish subscribe (DCPS) layer, which is illustrated in Figure 5.5. The DCPS layer relies on a rational information model to specify the information which belongs to the global data space. Subscriptions can be specified by means of topic and their associated types as well as content often presented by SQL expressions.

5.2.2 wxWavE

wxWavE is a visualization and simulation framework. With wxWavE the software modules written in C, C++, Python and Simulink can be visualized and tested. It not only provides the offline visualization for measurement data, but also supports the online visualization of realtime systems such as CAN, FlexRay and TCP/IP. wxWavE offers a good human machine interface (HMI) for in-car applications.

wxWavE combines different open and closed source frameworks. The programming language either C++ or python script can be used. Both 2D GUI framework wxWidgets and 3D OpenGL framework OpenSceneGraph can be implemented in wxWavE. The Vector Informatik products are supported by CAN, FlexRay. The measured data can be imported from or exported to different file types, such as CSV, MATLAB, Canalyzer, Diadem and HDF5.

Figure 5.6 shows a design overview of wxWavE. The green stack represents the kernel of wxWavE, which dealt with events, signals, GUI and software modules. The data are inputed from CAN, FlexRay, measurement and custom plugins to the system, and then outputed from the system back to CAN, measurement and custom plugins.

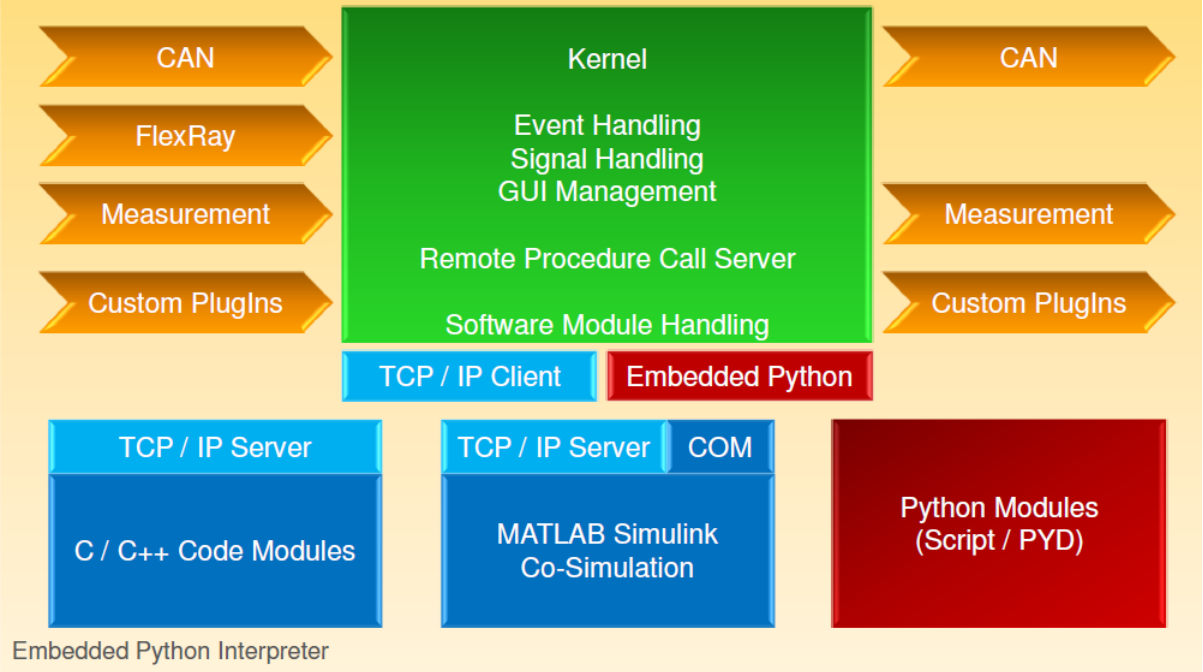


Figure 5.6.: wxWavE Design Overview [Sch10]

wxWavE supplies a flexible HMI. A large number of predefined widget types are available, such as plotwindows, bar graphs and ledš. The user operates the interface only by dragging and dropping the mouse without programming knowledge. The widget properties can be modified even at run time. The layouts can be customized by using own widget templates. They can be exchanged between users partly or completely. The layout is expressed in XML format. Moreover, the HMI automation is realized via python script or custom C++ callback dll's. On the one side, all HMI events can be passed to python script. On the other side, HMI can be controlled from the outside via python or C++ TCP/IP interface. The wizard is based on the generation of python script skeletons and C++ dll frames.

The defined software module interface is lean and strong. The modules are developed in Microsoft Visual Studio with versions 2005, 2008 and 2010. One single solution can generate binaries for TCP/IP module or S-Function-Dll and Python-DLL. Fast prototyping in python script must be conducted before C/C++ implementation. To the identical C/C++ algorithms, they can be tested in different environments.

The communication between HMI application and software modules is implemented via TCP/IP, which is shown in Figure 5.6 in the blue stacks. The software modules are used to describe the application process. If a software module is being debugged, wxWavE host application will not be freezed. If a module code is being crashed, wxWavE host application will not be affected. Furthermore, the host application and software modules can be distributed over the network.

5.3 Module Structure

In this work, the framework wxWave is used to visualize the free space based on the developed algorithms and the collected measurements.

Figure 5.7 shows the structure of the whole platform. The color green cuboid describes the channel and the blue square represents a DLL file. The developed algorithm for free space computation lies in the module 'FreeSpace Computation' in color khaki.

From Figure 5.8, it is clearly that the created module has four input parameters. One is the occupancy grid with free evidence, which is generated by 'MapFusion Task' and transmitted through the channel 'DstMap'. Another is the vehicle position information including the position coordinate, the vehicle angle and timestamp, which is provided by the channel 'VehiclePose'. The rest are the segmentation method and its corresponding threshold. The segmentation method shall be chosen between thresholding operation and dynamic programming.

After free space analysis, a new occupancy grid for the detected free space is outputted and then transmitted through the channel 'DstMap2' to the framework wxWavE.

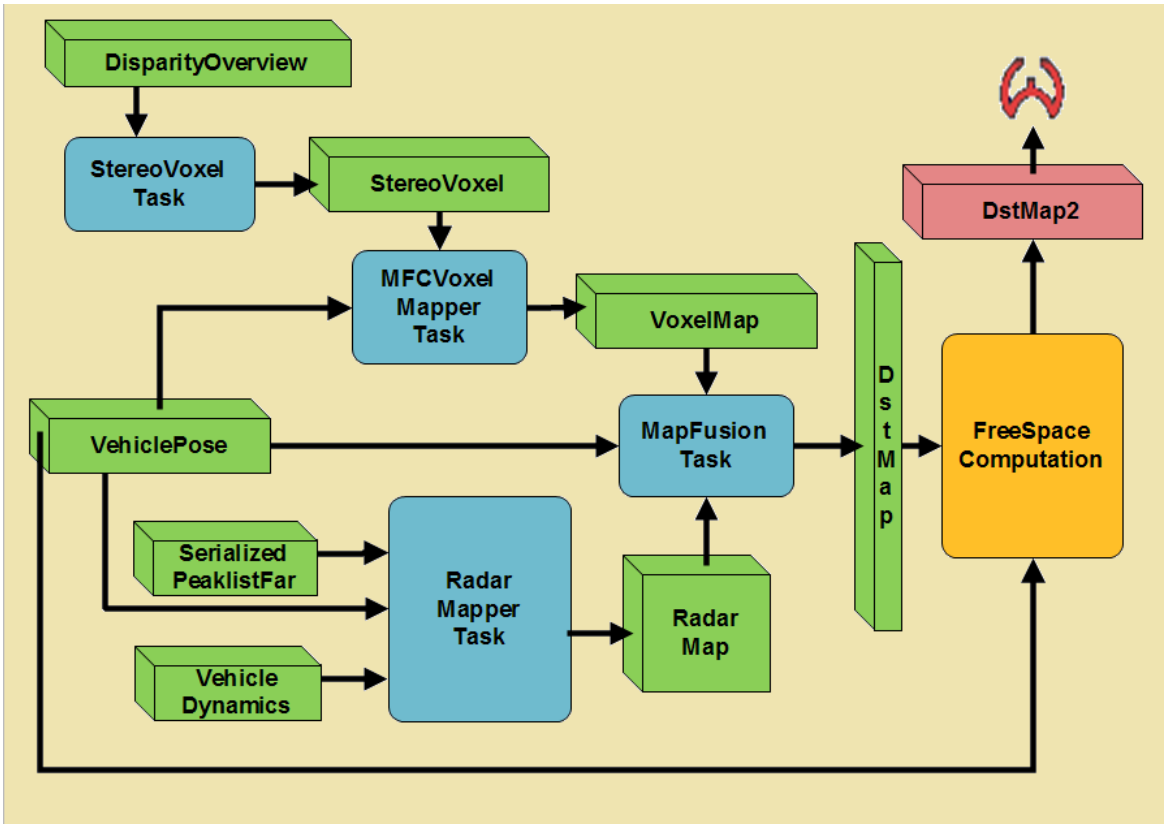


Figure 5.7.: wxWavE Design Overview [Sch10]

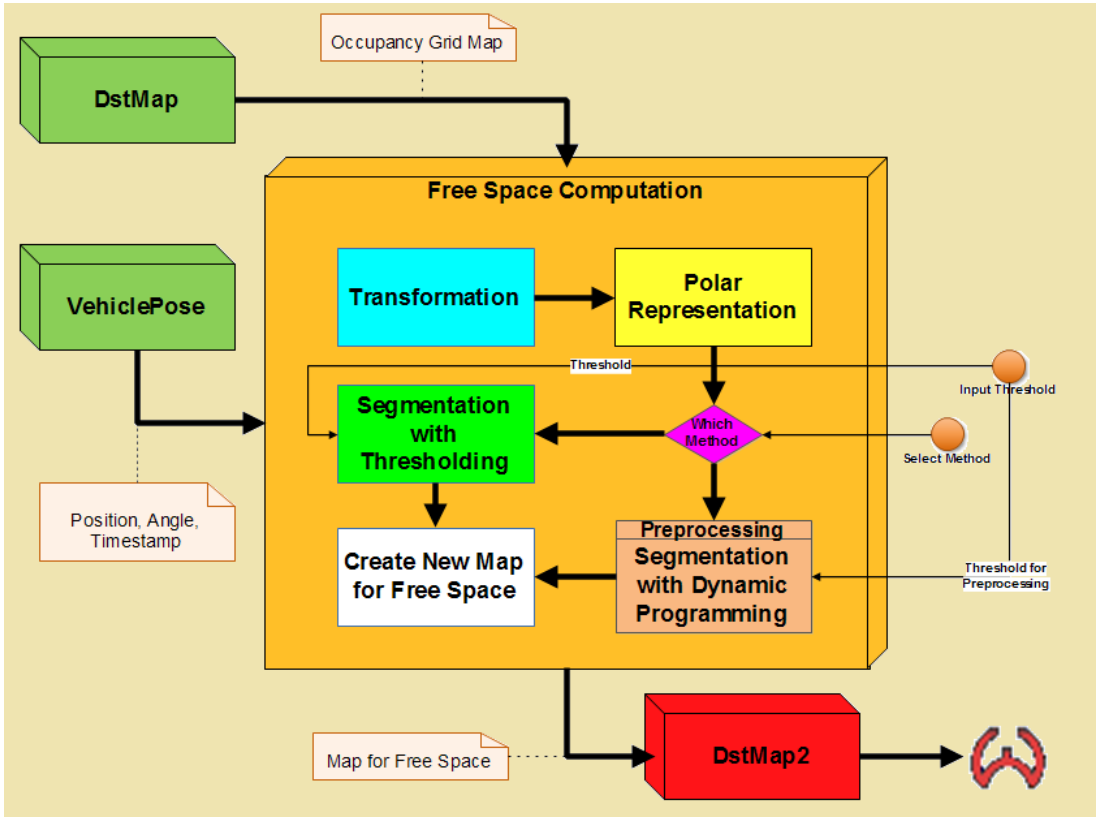


Figure 5.8.: wxWavE Design Overview [Sch10]

6 Results

The algorithm developed in this work shall undergo extensive tests to ensure a reliable and secure functionality of free space computation. This chapter evaluates and discusses the performance of those two methods presented in Chapter 4. Before the evaluation, a number of data sets have been already collected by the stereo camera CSF200 and the radar sensor ARS300. These two sensors have been introduced in Chapter 5. The measurement collection was conducted on 11. November 2011 and all the data were saved in *.dat files.

Because of the complexity and diversity of the urban environment the test shall be conducted in different scenarios, such as freeway, highway or downtown. In this work the measurements are collected along the test route which is illustrated in Figure 6.1. The blue line drawn in the image is the test route. The test begins at Point A, i.e. northwest cross Frankfurt. Then the test vehicle drives along Lorscher Strasse to Frankfurt Rödelheim downtown, where Point B, Point C and Point D are located. After turning left to Ludwig-Landmann-Strasse, our vehicle runs gerade until the final station of underground railway Line 6, Heerstrasse which Point E stands for.

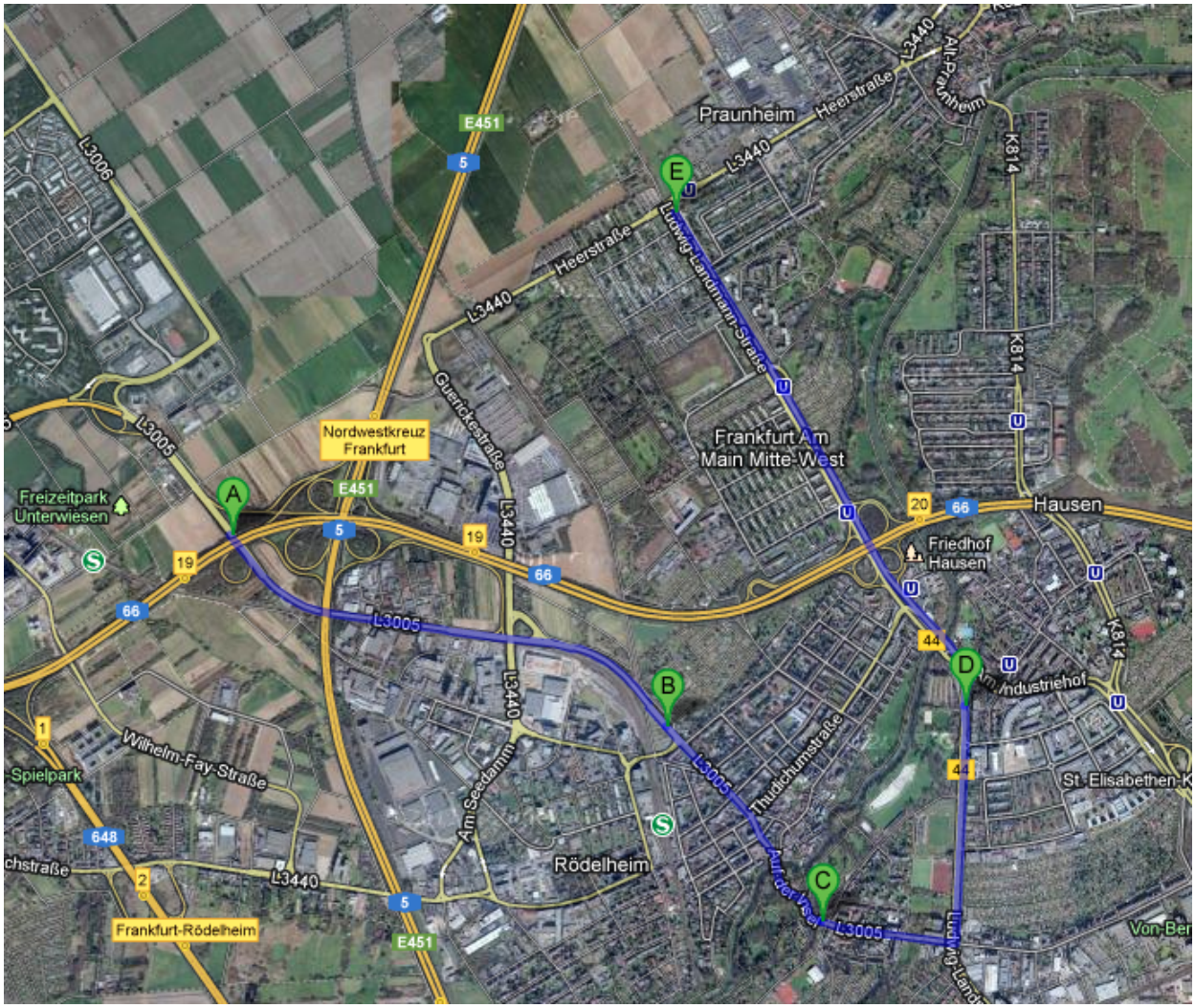


Figure 6.1.: Test Route Shown in Google Map

The whole test route consists of different scenarios including highway and downtown. The situation in the freeway is quite simple. Only cars run on the wide road and there is no pedestrian across the road. But in Downtown Rödelheim, the test environment becomes complex and diverse. The street is narrow, many cars are parked on both sides of the street and pedestrians sometimes suddenly cross the street. Hence, this test route is suitable for evaluating the functionality of free space analysis.

In the following sections some results of free space analysis in different scenarios are displayed. Section 6.1 tests the developed algorithm at the exit of freeway. Section 6.2 evaluate the test results in the urban environment. Section 6.3 analyses the segmented free space by the underground railway station.

During the evaluation in each scenario, results of the segmented occupancy grid in the corresponding polar representation and the free space in the corresponding cartesian coordinate system are analysed and compared with photos made by stereo camera.

6.1 Scenario 1: Exit of Freeway

Figure 6.2a shows the test route of Scenario 1 in Google Map. According to Google Map, the departure place is by northwest cross Frankfurt. Then test vehicle drives along Lorscher Strasse in the direction to Frankfurt Rödelheim downtown. Figure 6.2b shows the scene at the exit of Freeway A66. From Figure 6.2b we can see the road situation in Lorscher Strasse looks quite simple. Although there is only one lane in each direction, the road is still relatively wide. Only vehicles runs on the road and there is no pedestrian across the road.

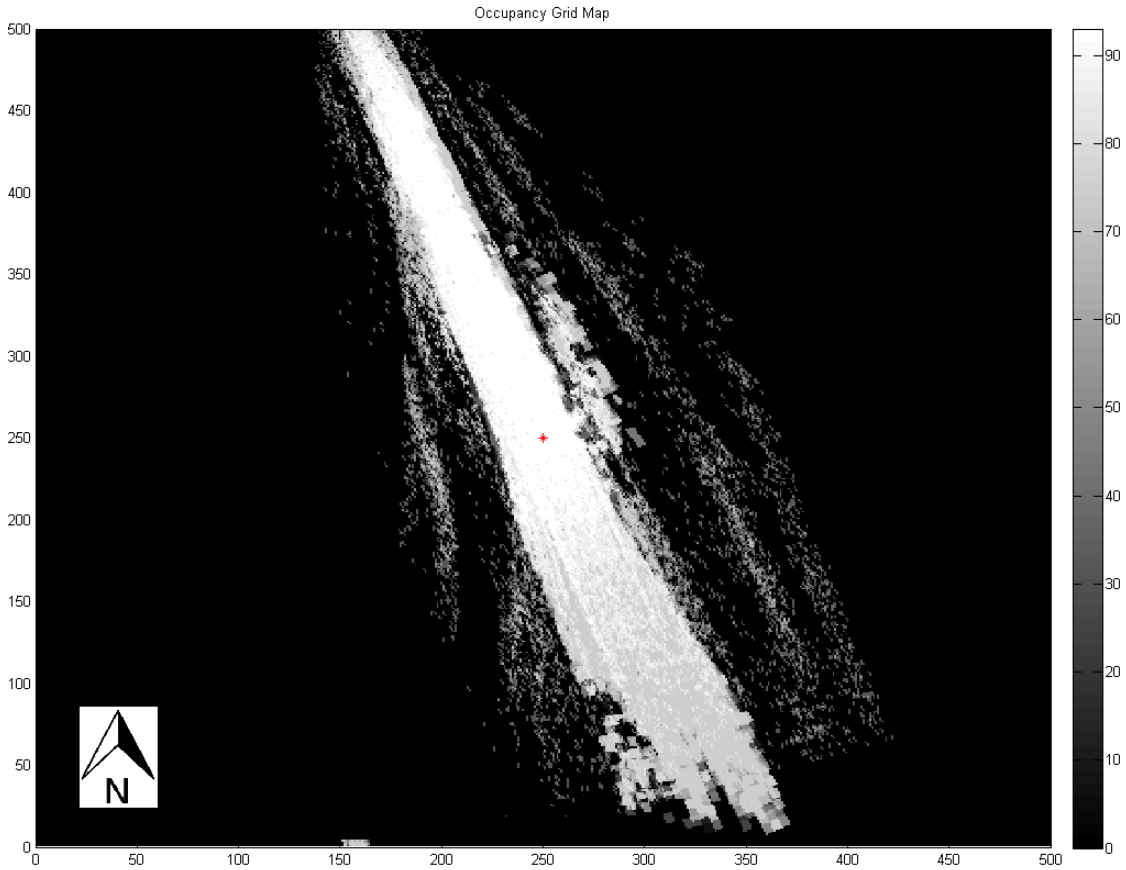
(a) Test Route of Scenario 1 Shown in Google Map

(b) Scene at Exit of Freeway A66 Shown from Google Map

Figure 6.2.: Scenario 1 Shown in Google Map

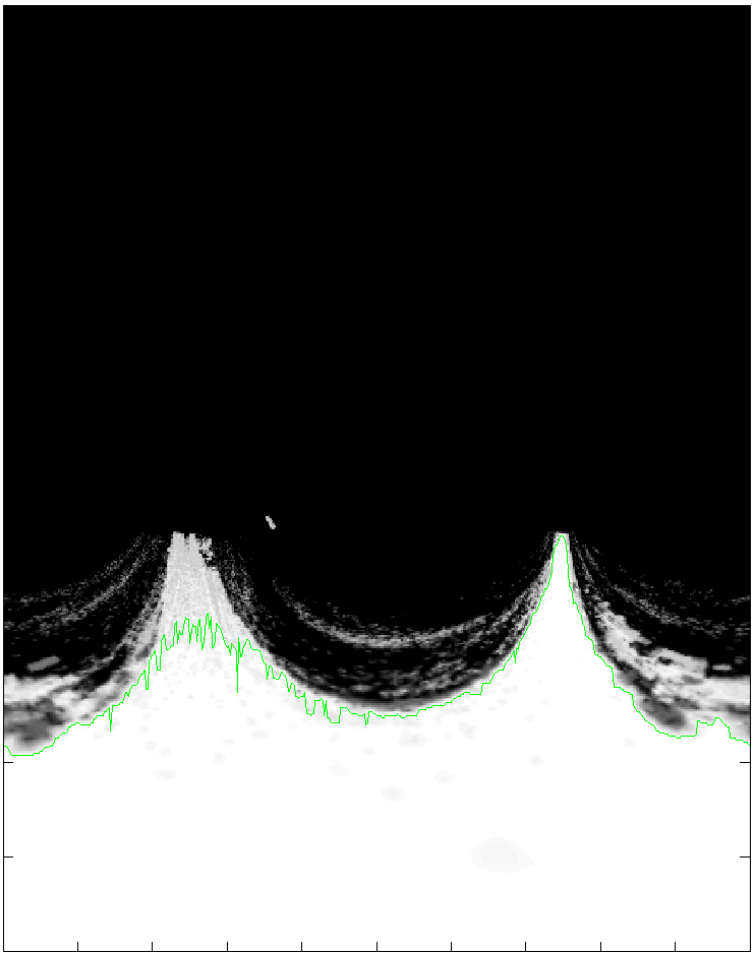
The exampler is chosen at timestamp 1312.230sec. At that moment the scene in front of the test vehicle is captured by the stereo camera, which is illustrated in Figure 6.3a. The generated occupancy grid map corresponding to that timestamp is also illustrated in Figure 6.3b. In the occupancy grid map, both axes correspond the world coordinate system. The horizontal axis points to the east, while the vertical axis points to the north. Each edge of the map is composed of 500 cells. The free cells are displayed in color white and the occupied cells or unknownness are displayed in color black. On the right side there is a color bar. Different colors are used to represent different likelihoods of the free space. The darker the cell's color displays, the less likelihood the free space exists with. Our vehicle is located in the middle of the map at the point $(250, 250)$, which is marked by the red cross. And it is running from the northwest to the southeast.

(a) Scene at Timestamp=1312.230sec Captured by Stereo Camera

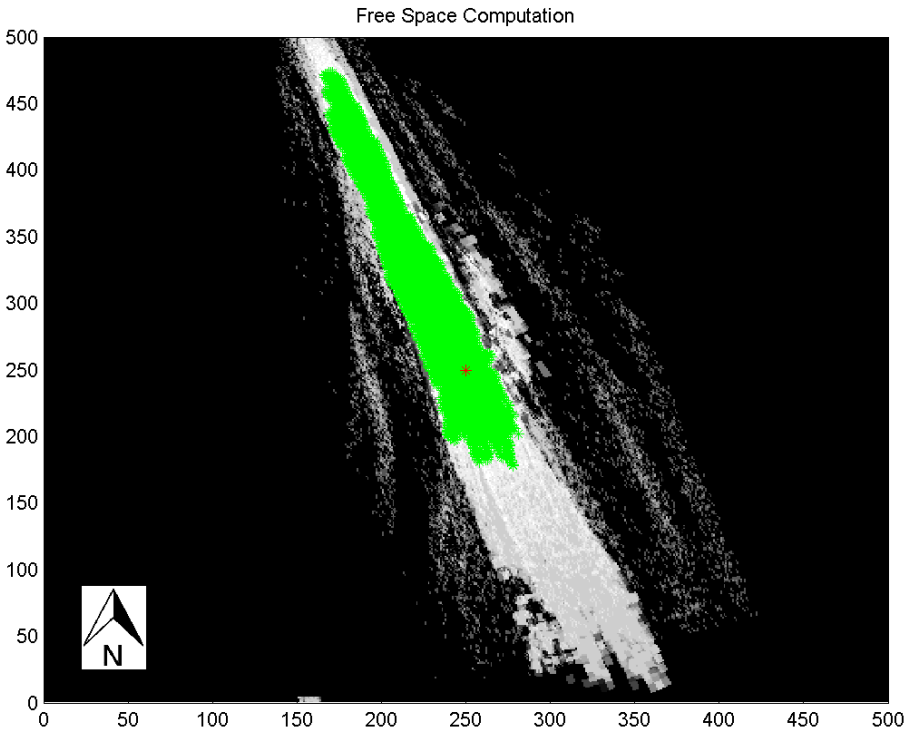


(b) Generated Occupancy Grid Map at Timestamp=1312.230sec

Figure 6.3.: Input Occupancy Grid Map with Corresponding Photo in Scenario 1

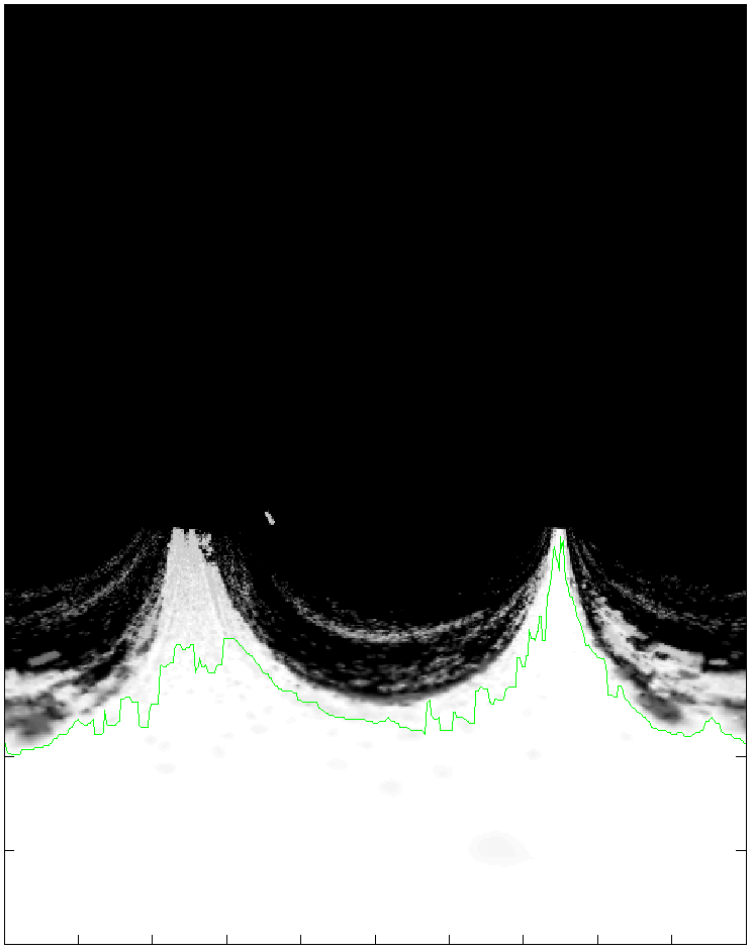


(a) Segmented Occupancy Grid in Corresponding Polar Representation

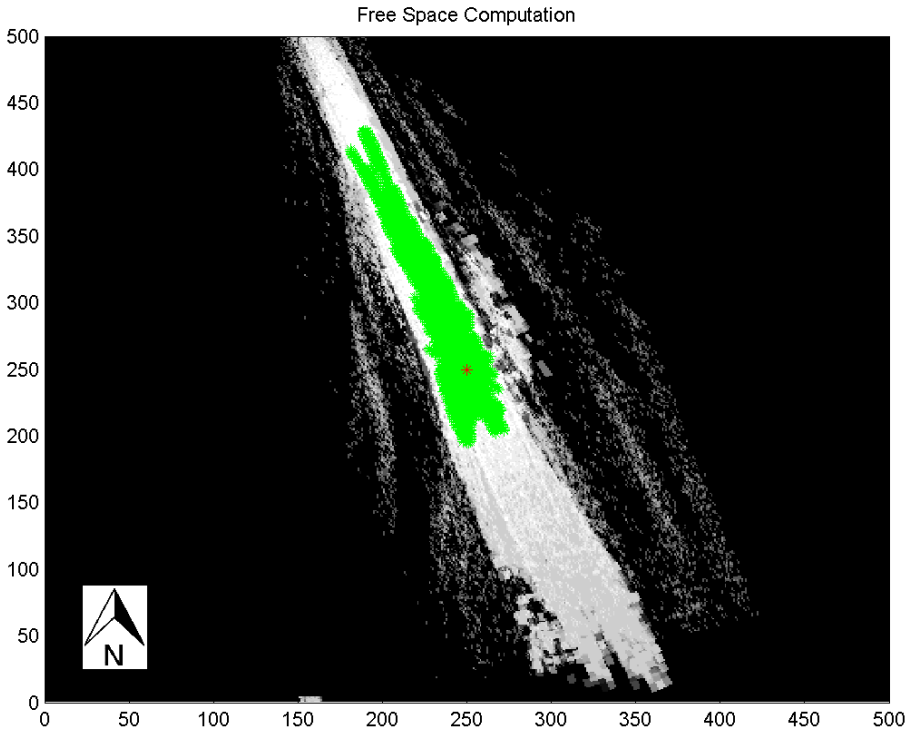


(b) Free Space in Corresponding Cartesian Coordinate System

Figure 6.4.: Free Space Computation by Thresholding with Threshold=85 in Scenario 1



(a) Segmented Occupancy Grid in Corresponding Polar Representation



(b) Free Space in Corresponding Cartesian Coordinate System (Preprocessing with Threshold=15)

Figure 6.5.: Free Space Computation by Dynamic Programming in Scenario 1

Based on the occupancy grid map in Figure 6.3b, the free space around the vehicle is computed by thresholding and dynamic programming methods respectively. The obtained free space by thresholding operation is illustrated in Figure 6.4b and the result by dynamic programming is illustrated in Figure 6.5b. The segmented occupancy grids in the corresponding polar representation generated by both methods are displayed in Figure 6.4a and 6.5a respectively.

For the whole free space analysis, the inputed threshold is set to 85 in the thresholding operation and the threshold in the preprocessing before conducting dynamic programming is set to 15.

Comparing these two results, the following conclusion can be reached. Both results are good, because they can clearly distinguish the free space around the vehicle from obstacles and unknownness. The only difference is, the segmentation path is rougher and the analysed free space around the vehicle is relatively bigger in thresholding operation than in dynamic programming.

6.2 Scenario 2 and 3: Downtown Rödelheim

The second and the third scenario take place in the downtown. At that moment our test vehicle is driving across Rödelheim. The test route in Google Map is shown in Figure 6.6a and the street scene of Rödelheim from Google Map is shown in Figure 6.6b. The street environment becomes complex and diverse. From Figure 6.6b, we can see that the downtown street is much narrower than the highway; on the both sides of the street many cars are parked; and even worse, pedestrians can suddenly cross the street.

(a) Test Route of Scenario 2 and 3 Shown in Google Map

(b) Scene in Downtown Rödelheim Shown from Google Map

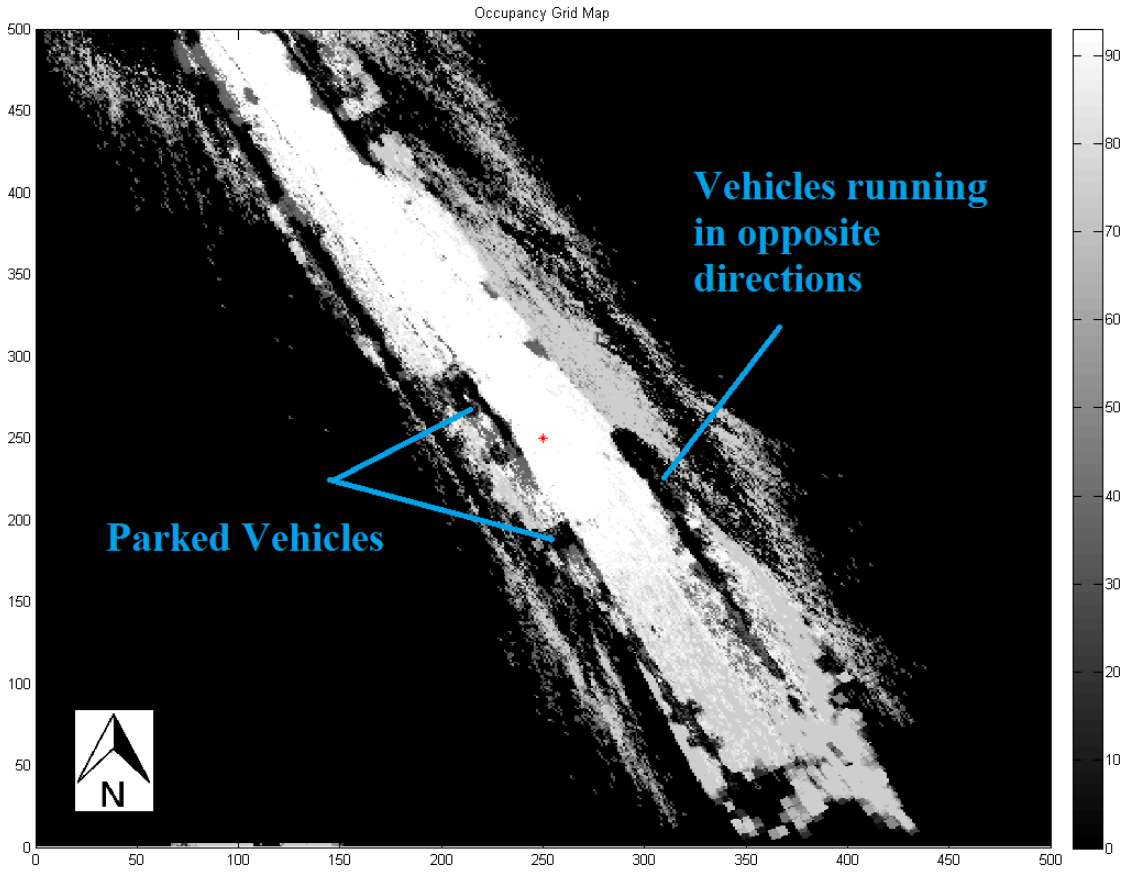
Figure 6.6.: Scenario 2 and 3 Shown in Google Map

Thus, results of free space computation in different conditions shall be observed. In Subsection 6.2.1 the obtained free space in the narrow downtown street is checked and in Subsection the result is inspected whether a pedestrian is distinguished from the free space.

6.2.1 Scenario 2: Narrow Downtown Street

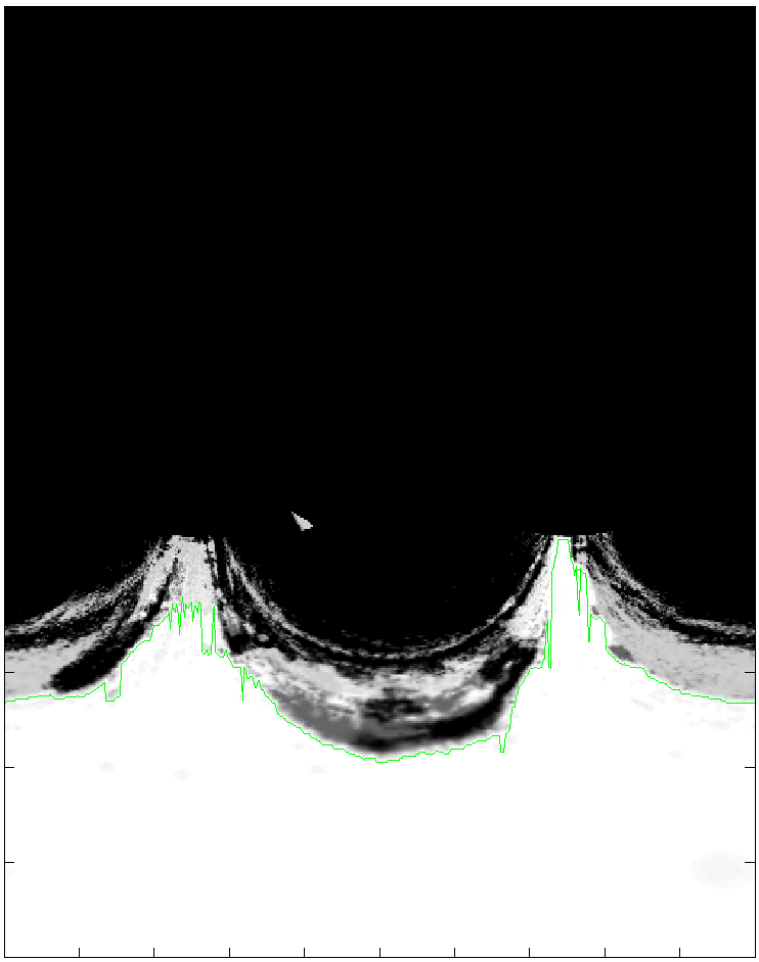
The exampler is chosen at timestamp 1445.232sec. At that moment the scene in front of the test vehicle is captured by the stereo camera, which is illustrated in Figure 6.7a. The generated occupancy grid map corresponding to that timestamp is also illustrated in Figure 6.7b. The scenario takes place in the crossroad between Lorscher Strasse and Arnoldshainer Strasse. From Figure 6.7a we can see many cars parked on both sides of the street and several vehices are just running in opposite directions of our test vehicle, which are also marked in the occupancy grid map.

(a) Scene at Timestamp=1445.232sec Captured by Stereo Camera

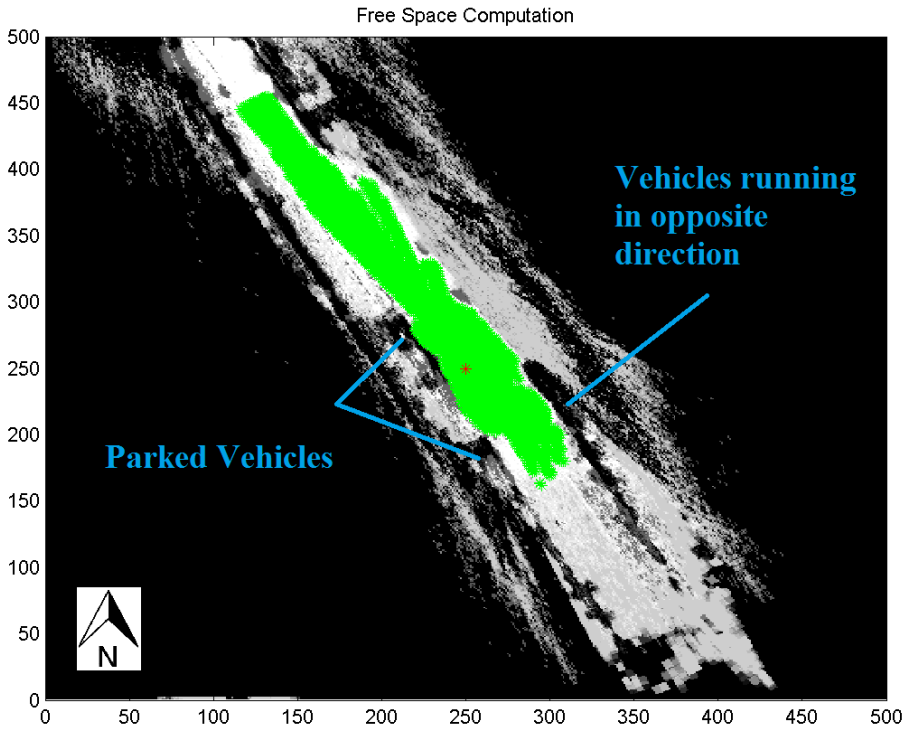


(b) Generated Occupancy Grid Map at Timestamp=1312.230sec

Figure 6.7.: Input Occupancy Grid Map with Corresponding Photo in Scenario 2

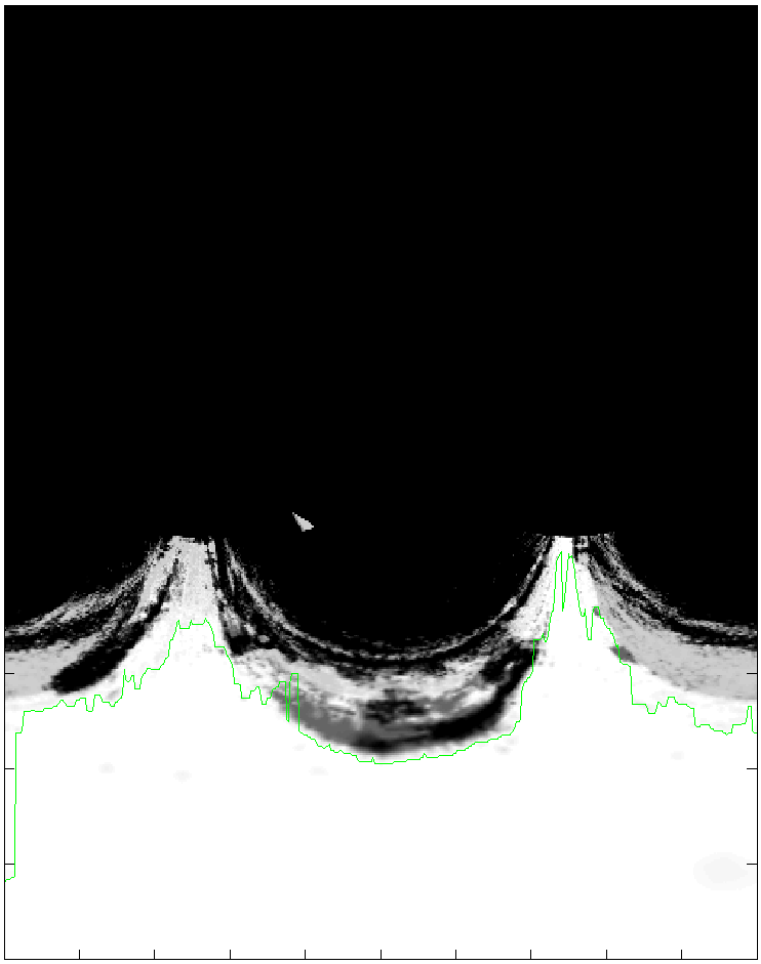


(a) Segmented Occupancy Grid in Corresponding Polar Representation

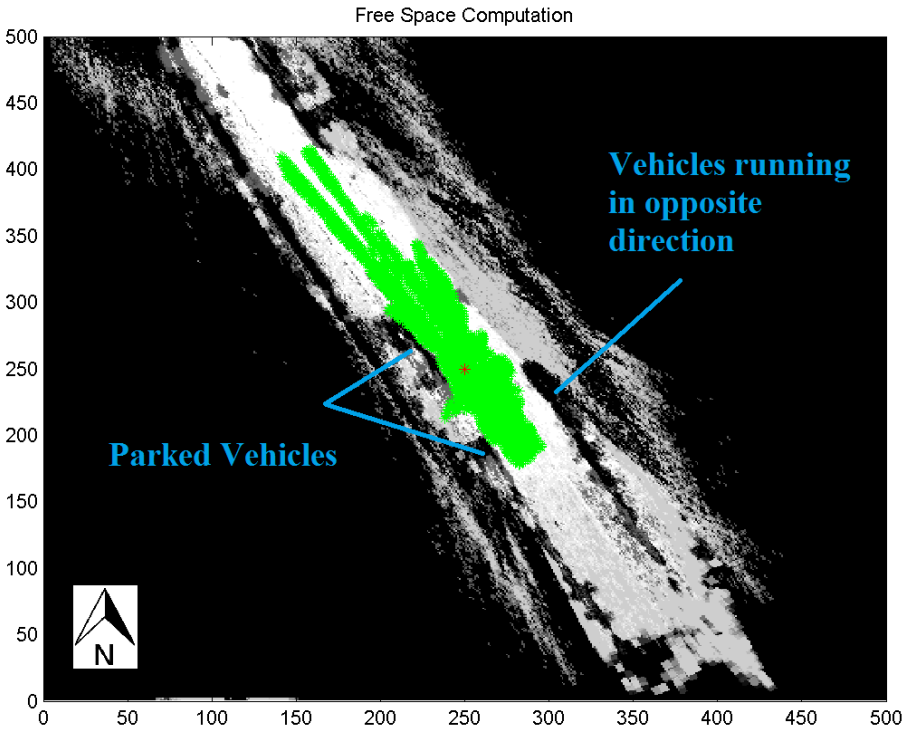


(b) Free Space in Corresponding Cartesian Coordinate System

Figure 6.8.: Free Space Computation by Thresholding with Threshold=85 in Scenario 2



(a) Segmented Occupancy Grid in Corresponding Polar Representation



(b) Free Space in Corresponding Cartesian Coordinate System (Preprocessing with Threshold=15)

Figure 6.9.: Free Space Computation by Dynamic Programming in Scenario 2

Based on the occupancy grid map in Figure 6.7b, the free space around the vehicle is computed by thresholding and dynamic programming methods respectively. The obtained free space by thresholding operation is illustrated in Figure 6.8b and the result by dynamic programming is illustrated in Figure 6.9b. The segmented occupancy grids in the corresponding polar representation generated by both methods are displayed in Figure 6.8a and 6.9a respectively.

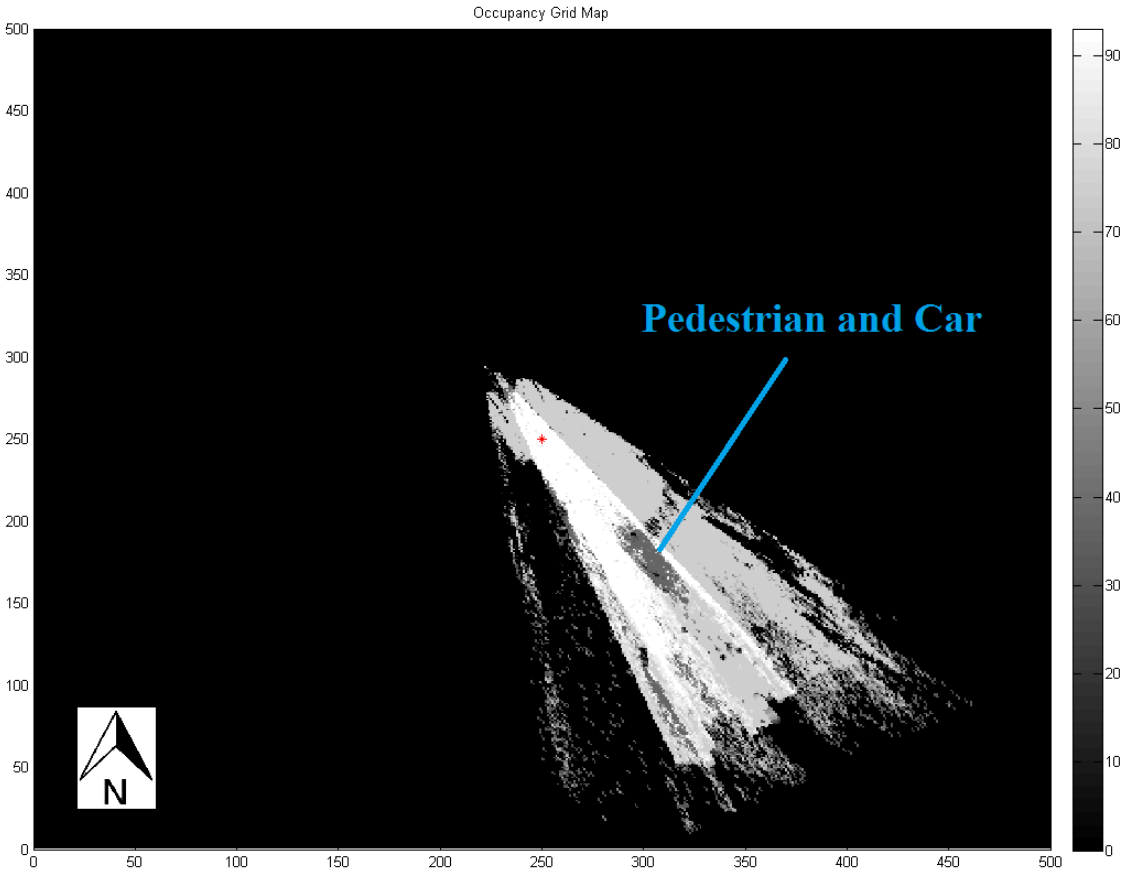
For the whole free space analysis, the inputed threshold is set to 85 in the thresholding operation and the threshold in the preprocessing before conducting dynamic programming is set to 15.

Comparing these two results, the following conclusion can be reached. Both results are good, because they can clearly seperate the free space around our vehicle from other vehicles running in the opposite direction and parked cars on both sides of the street. The only difference is, the segmentation path is rougher and the analysed free space around the vehicle is relatively bigger in thresholding operation than in dynamic programming.

6.2.2 Scenario 3: Pedestrian across Street

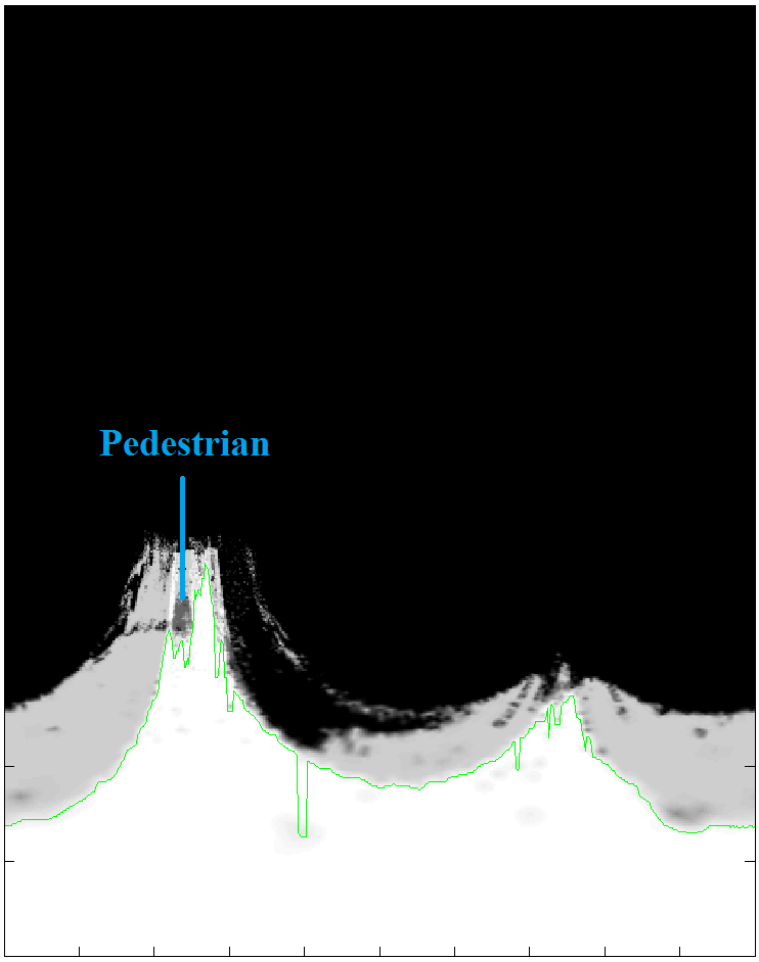
The exampler is chosen at timestamp 1493.366sec. At that moment the scene in front of the test vehicle is captured by the stereo camera, which is illustrated in Figure 6.10a. The generated occupancy grid map corresponding to that timestamp is also illustrated in Figure 6.10b. In the occupancy grid map, both axes correspond the world coordinate system. The horizontal axis points to the east, while the vertical axis points to the north. Each edge of the map is composed of 500 cells. The free cells are displayed in color white and the occupied cells or unknownness are displayed in color black. On the right side there is a color bar. Different colors are used to represent different likelihoods of the free space. The darker the cell's color displays, the less likelihood the free space exists with. Our vehicle is located in the middle of the map at the point $(250, 250)$, which is marked by the red cross. And it is running from the northwest to the southeast. The scenario takes place in the crossroad between Lorscher Strasse and Thudichumstrasse. From Figure 6.10a we can see a pedestrian suddenly crossing the street when traffic light turns green. The pedestrian is also marked in the occupancy grid map.

(a) Scene at Timestamp=1493.366sec Captured by Stereo Camera

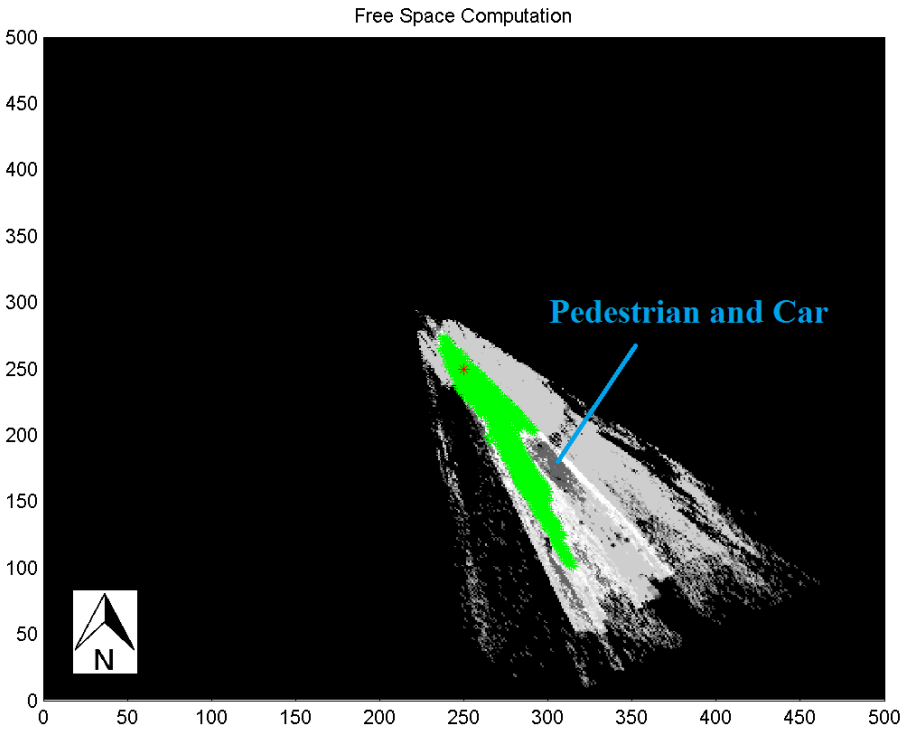


(b) Generated Occupancy Grid Map at Timestamp=1493.366sec

Figure 6.10.: Input Occupancy Grid Map with Corresponding Photo in Scenario 3

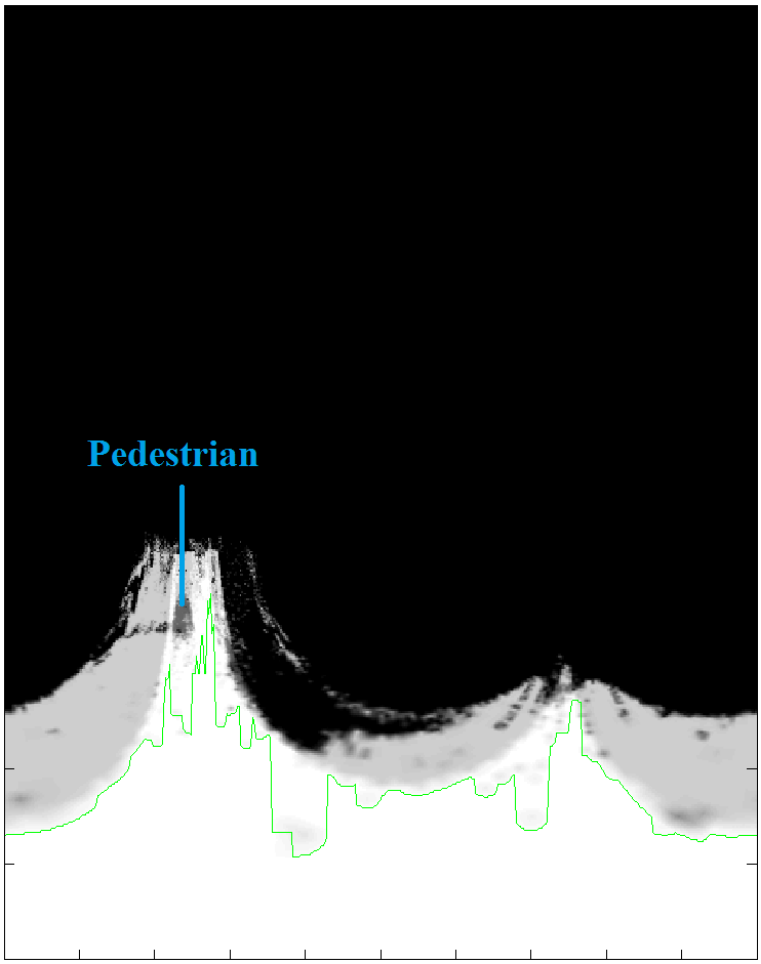


(a) Segmented Occupancy Grid in Corresponding Polar Representation

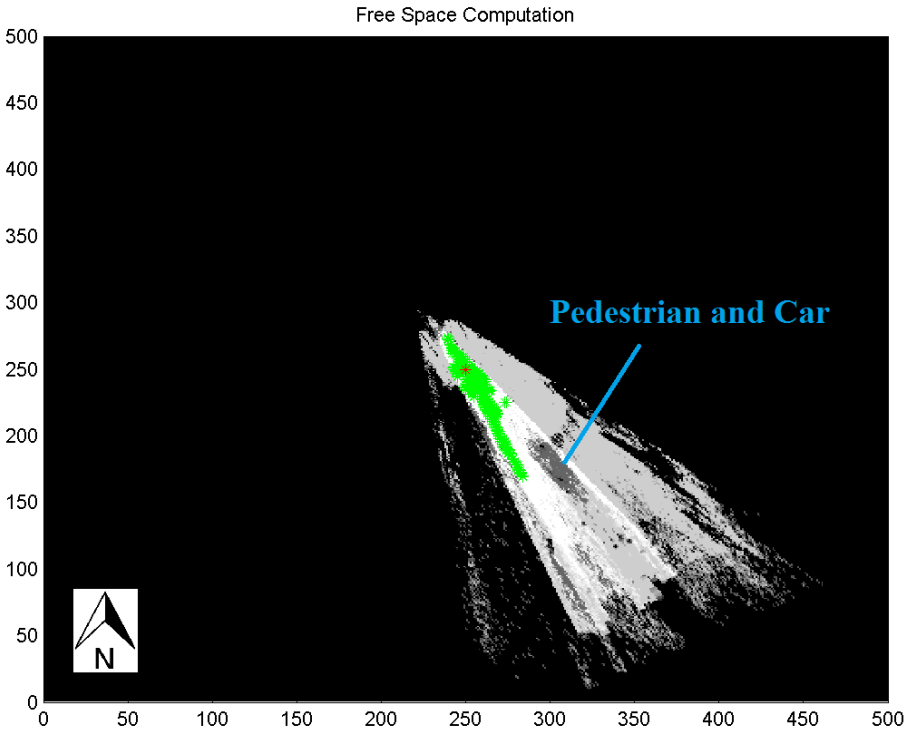


(b) Free Space in Corresponding Cartesian Coordinate System

Figure 6.11.: Free Space Computation by Thresholding with Threshold=85 in Scenario 3



(a) Segmented Occupancy Grid in Corresponding Polar Representation



(b) Free Space in Corresponding Cartesian Coordinate System (Preprocessing with Threshold=15)

Figure 6.12.: Free Space Computation by Dynamic Programming in Scenario 3

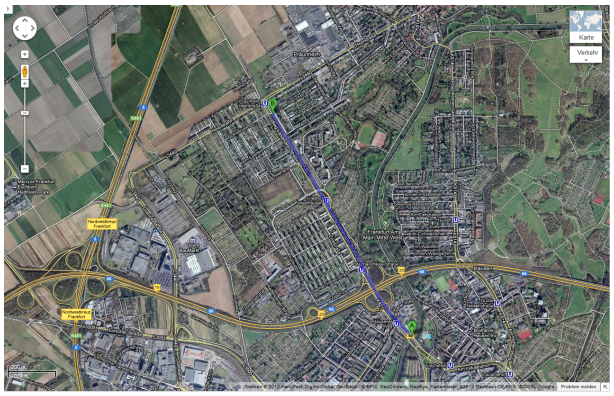
Based on the occupancy grid map in Figure 6.10b, the free space around the vehicle is computed by thresholding and dynamic programming methods respectively. The obtained free space by thresholding operation is illustrated in Figure 6.11b and the result by dynamic programming is illustrated in Figure 6.12b. The segmented occupancy grids in the corresponding polar representation generated by both methods are displayed in Figure 6.11a and 6.12a respectively.

For the whole free space analysis, the inputed threshold is set to 85 in the thresholding operation and the threshold in the preprocessing before conducting dynamic programming is set to 15.

Comparing these two results, the following conclusion can be reached. Both results are good, because they can clearly isolate the free space around our vehicle from the pedestrian and the car ahead. The only difference is, the segmentation path is rougher and the analysed free space around the vehicle is relatively bigger in thresholding operation than in dynamic programming.

6.3 Scenario 4 : Underground Railway Station

Figure 6.13a shows the test route of Scenario 4 in Google Map. According to Google Map, our test vehicle drives along Ludwig-Landmann-Strasse until underground railway station Heerstrasse. The test is finished at the crossroad between Ludwig-Landmann-Strasse and Heerstrasse. Figure 6.13b shows the scene of the underground railway station Heerstrasse. From Figure 6.13b we can see the road situation in Ludwig-Landmann-Strasse looks quite simple. The underground railway divides the road into two parts. Each part has two lanes in the same direction. Only vehicles runs on this wide road and there is no pedestrian across the road.



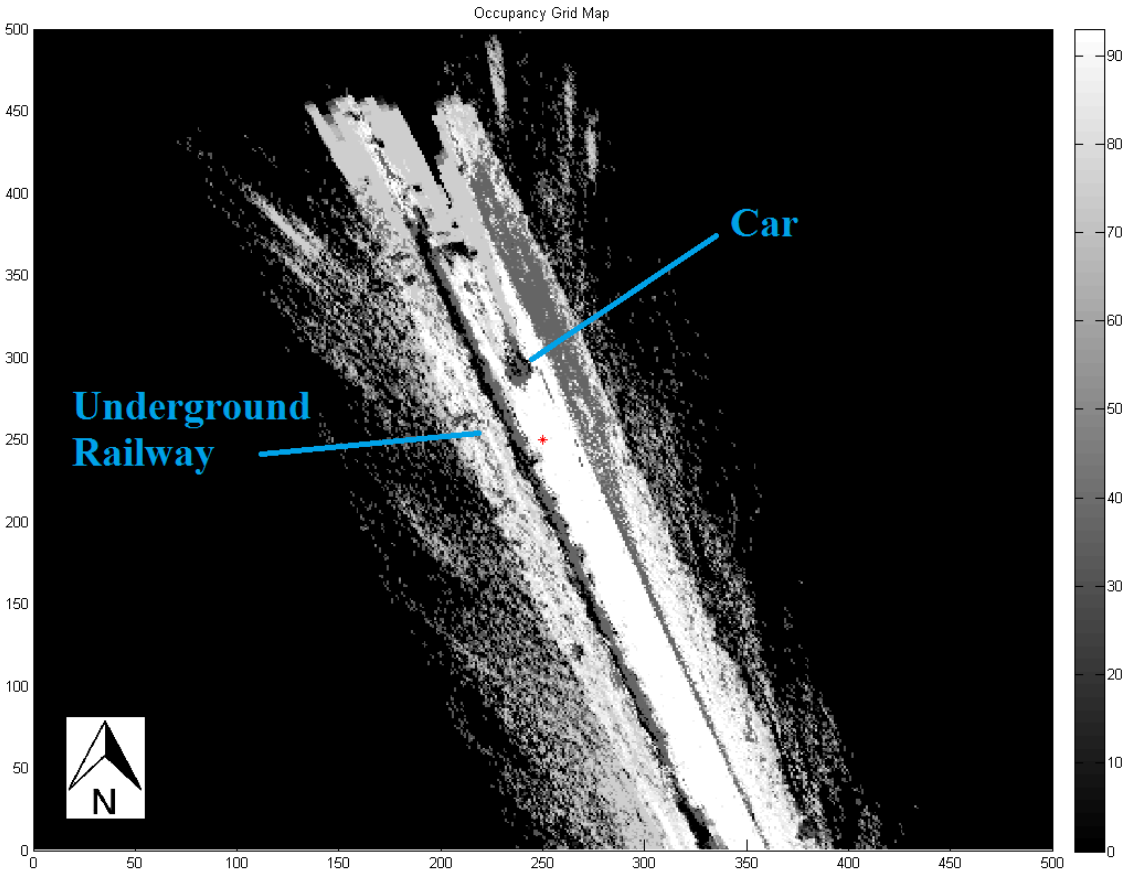
(a) Test Route of Scenario 4 Shown in Google Map

(b) Scene at Exit of Freeway A66 Shown from Google Map

Figure 6.13.: Scenario 4 Shown in Google Map

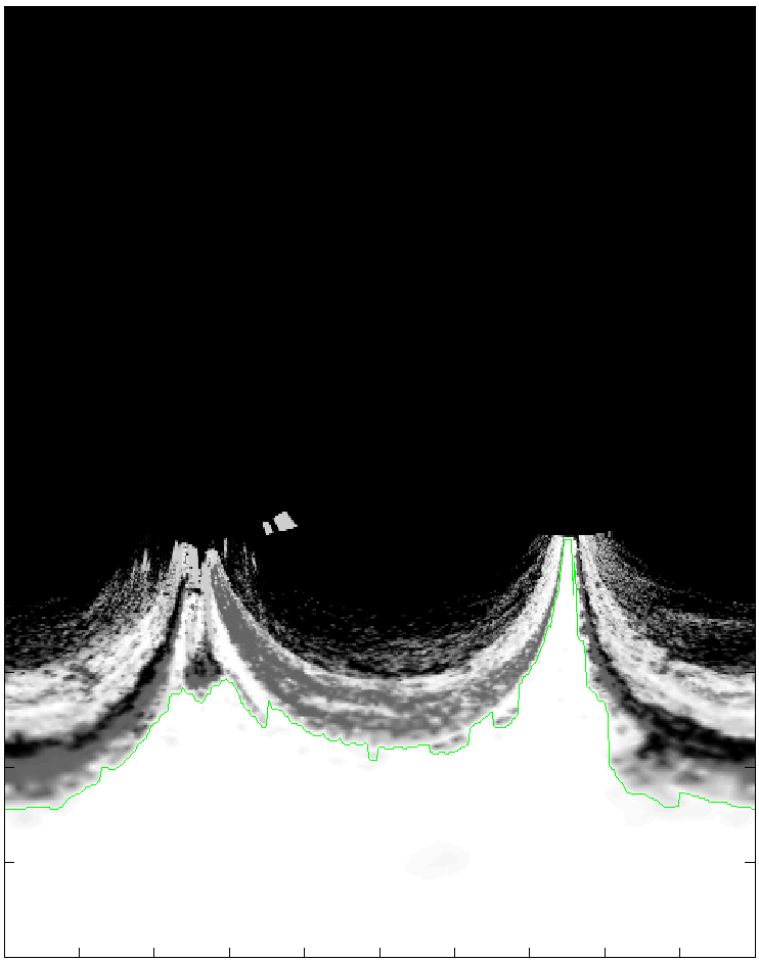
The exampler is chosen at timestamp 1837.241sec. At that moment the scene in front of the test vehicle is captured by the stereo camera, which is illustrated in Figure 6.14a. The generated occupancy grid map corresponding to that timestamp is also illustrated in Figure 6.14b. The scenario takes place in the crossroad between Ludwig-Landmann-Strasse and Heerstrasse. From Figure 6.14a we can see many cars parked on both sides of the street and several vehices are just running in opposite directions of our test vehicle, which are also marked in the occupancy grid map.

(a) Scene at Timestamp=1837.241sec Captured by Stereo Camera

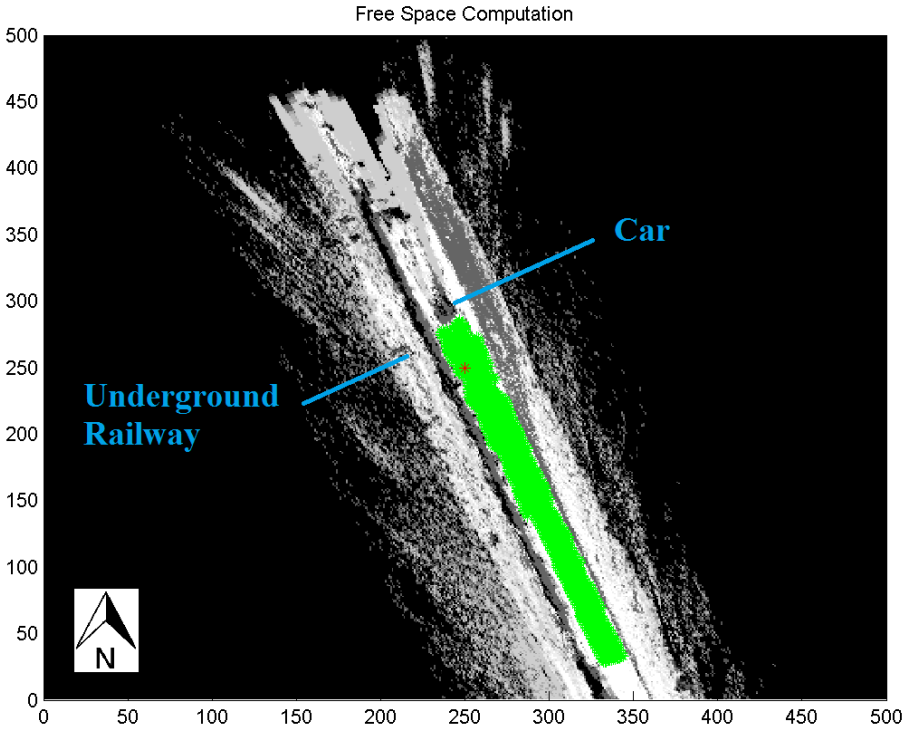


(b) Generated Occupancy Grid Map at Timestamp=1837.241sec

Figure 6.14.: Input Occupancy Grid Map with Corresponding Photo in Scenario 4

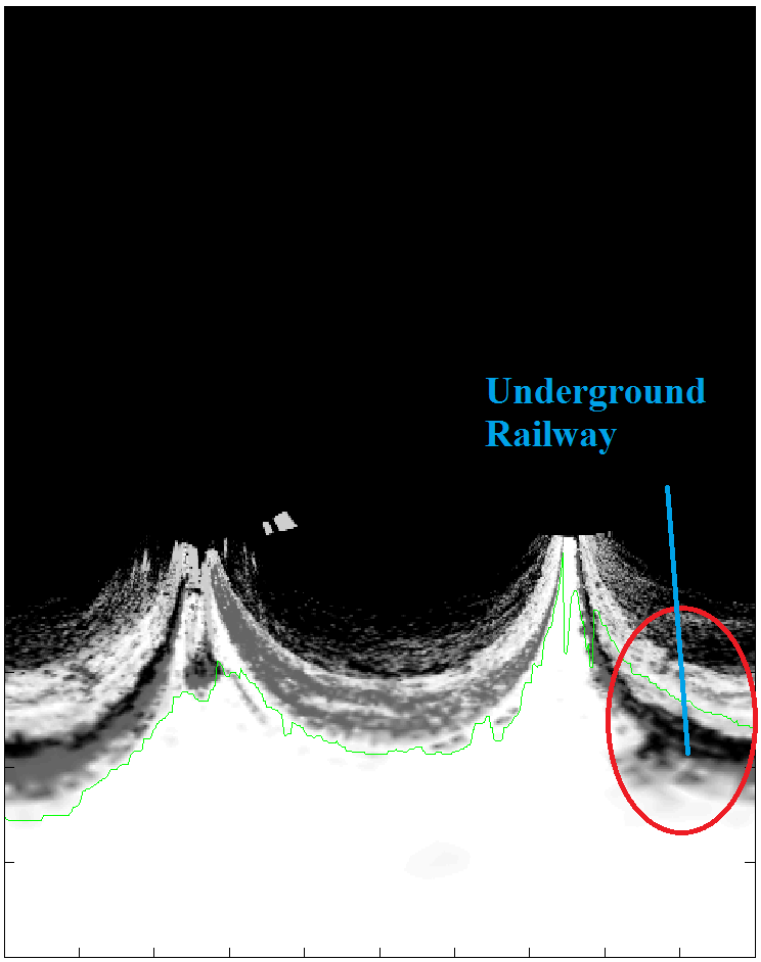


(a) Segmented Occupancy Grid in Corresponding Polar Representation

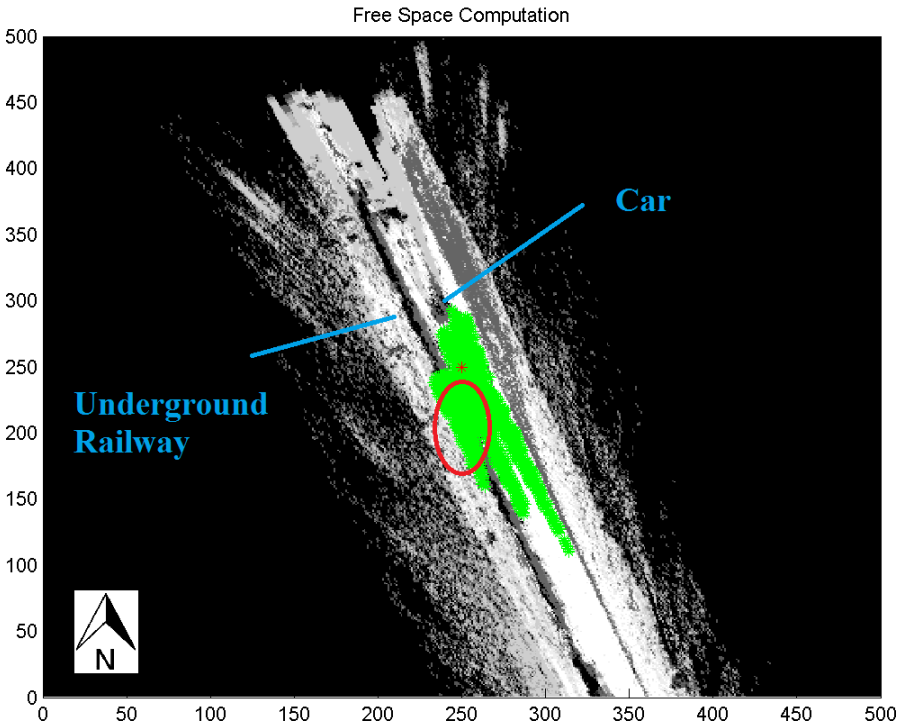


(b) Free Space in Corresponding Cartesian Coordinate System

Figure 6.15.: Free Space Computation by Thresholding with Threshold=85 in Scenario 4



(a) Segmented Occupancy Grid in Corresponding Polar Representation



(b) Free Space in Corresponding Cartesian Coordinate System

Figure 6.16.: Free Space Computation by Dynamic Programming in Scenario 4

Based on the occupancy grid map in Figure 6.14b, the free space around the vehicle is computed by thresholding and dynamic programming methods respectively. The obtained free space by thresholding operation is illustrated in Figure 6.15b and the result by dynamic programming is illustrated in Figure 6.16b. The segmented occupancy grids in the corresponding polar representation generated by both methods are displayed in Figure 6.15a and 6.16a respectively.

For the whole free space analysis, the inputed threshold is set to 85 in the thresholding operation and the threshold in the preprocessing before conducting dynamic programming is set to 15.

Comparing these two results, the following conclusion can be reached. The result of thresholding operation is much better than that of dynamic programming, because some underground railway area shall be regarded as occupied area, but it is recognized as free space by dynamic programming. A solution to that problem is to change the threshold in the preprocessing.

7 Discussion

This chapter is the last part of the paper. A conclusion is reached and futur directions of the work is presented.

7.1 Conclusion

With the motivation of safty drive, the purpose of this project is to find out a possible way to analyse the free space around the vehicle.

Through the review of the literature about occupancy grid, it is clear that this method was first introduced in the field of mobile robot navigation and path planning. Nowadays, a lot of researches measure the free space with the occupancy grid. Besides, there exist other ways for free space analysis that do not apply occupancy grid, such as the optical flow.

During this work, two algorithms for free space computation is built. One is segmentation by thresholding, the other is segmentation by dynamic programming. Although threholding operation can implement faster than dynamic programming, the obtained result of free space is not optimal.

In this work the developed program has been tested with the real measure data in the framework wxWavE. All the measurements have been already collected by a stereo camera CSF 200 and a radar sensor ARS 300. The test route lies in the downtown and highway. Since the street environment in the downtown is much more complexer and diverser than that in the highway, the test route can completely examine the functionality of the designed module.

7.2 Future Work

Most test results are good, but there exist some bad results when dynamic programming is used. The problem is that it is difficult to set the threshold for the preprocessing of dynamic programming.

Moreover, during the work, there is no test on real vehicle conducted. That can be realized in the futur.

A Dynamic Programming

Complex problems are often broken down into several simple subproblems by the dynamic programming method. If the overlapping subproblems are slightly smaller and optimal substructure, this method will be applicable and take far less time. The Bellman equation is used as the tool of dynamic programming.

A.1 Overview

In the 1940s Richard Bellman put forward a new theory to deal with the mathematical optimization method known as dynamic programming. In honor of Bellman's dedication, the discovery is named after his name. The word *dynamic* and *programming* chosen by Bellman have their own meaning: the former is used to capture the time-varying aspect of the problems; the latter is referred to the use of the method to find an optimal program.

Dynamic programming is both a mathematical optimization method and a computer programming method. It deals with complex problems in a recursive manner: first solving different parts of the problem and then combining the solutions to subproblems. If some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively, which corresponds to the "Principle of Optimality". Likewise, in computer science, a problem that can be broken down recursively is thought to have optimal substructure.

A.2 Dynamic Programming in Computer Programming

If the dynamic programming method is applicable, the optimization problem must have the following two important attributes: optimal substructure and overlapping subproblems.

Optimal substructure indicates that the solution to a given optimization problem can be obtained by the combination of optimal solutions to its subproblems. Consequently, the first step towards devising a dynamic programming solution is to check whether the problem exhibits such optimal substructure. Such optimal substructures are usually described by means of recursion.

Overlapping subproblems show that the space of subproblems must be small. That means, any recursive algorithm solving the problem should solve the same subproblems over and over rather than generating new subproblems. Dynamic programming takes account of this fact and solves each subproblem only once.

Dynamic programming can be achieved in either of two ways: top-down approach and bottom-up approach. Top-down dynamic programming breaks each large problem down into several small problems recursively and stores the solutions to these subproblems which will be later used again. On the contrary, bottom-up dynamic programming solves all the subproblems first and use their solutions to build-on and arrive at the solutions to bigger subproblems.

Some programming languages can automatically memorize the result of a function call with a particular set of arguments, in order to speed up call-by-name evaluation. Some languages make it possible portably (e.g. Scheme, Common Lisp or Perl), some need special extensions (e.g. C++). Some languages have automatic memoization built in, such as tabled Prolog and J.

A.3 Bellman Equation

If subproblems can be nested recursively inside larger problems, dynamic programming methods will be applicable, because there exists a relation between the value of the larger problem and the values of the subproblems. This relationship is called the **Bellman equation**.

The Bellman equation is applicable, only if a dynamic optimization problem is in discrete time. A continuous-time optimization problem can be handled by the Hamilton-Jacobi-Bellman equation.

To understand the Bellman equation, the following concepts should be understood.

- **objective function**: the mathematical function that describes the objective: max or min
- **state variable**: the variable of the information about the current situation that is needed to make a correct decision
- **control variable**: the action that changes into the new state
- **policy function**: the function of the states that determines the actions
- **value function**: the function of the state that describes the best possible value of the objective by the optimal decision rule

The state at time t is written as x_t. The decision begins at time 0 with the initial state x_0. The set of possible actions depends on the current state $a_t \in \Gamma(x_t)$ where the action a_t represents one or more control variables. The state transforms from x_t to a new state x_{t+1} when the action a_t is taken, which is described by a state transition function $T(x_t, a_t)$. The cost from taking the action a_t in the state x_t is expressed as a function $F(x_t, a_t)$. Finally, the impatience is assumed by a discount factor $0 < \beta < 1$.

Under these assumptions, a decision problem takes the following form:

$$\begin{aligned} &V(x_0) = \min_{\{a_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t F(x_t, a_t) \\ &a_t \in \Gamma(x_t), x_{t+1} = T(x_t, a_t), \forall t = 0, 1, 2, ... \end{aligned} \tag{A.1}$$

The notation $V(x_0)$ represents the optimal value obtained by minimizing this objective function subject to the assumed constraints. This function is the value function. It is a function of the initial state variable x_0, since the best value obtainable depends on the initial situation.

The dynamic programming method breaks the decision problem into smaller subproblems. Richard Bellman's **Principle of Optimality** describes how to do this:

> Principle of Optimality: *An optimal policy has the property that, whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.* (See Bellman, 1957, Chap.III.3.)

According to the suggestion of the Principle of Optimality, the first decision can be separated from all future decisions and then taken into consideration. The rest optimization problem begins from time 1 with the new state x_1. Collecting the future decisions in brackets on the right, the decision problem with Equation A.1 can be rewritten as:

$$\begin{aligned} &V(x_0) = \min_{a_0} \left\{ F(x_0, a_0) + \beta \left[\min_{\{a_t\}_{t=1}^{\infty}} \sum_{t=1}^{\infty} \beta^{t-1} F(x_t, a_t) s.t. a_t \in \Gamma(x_t), x_{t+1} = T(x_t, a_t), \forall t = 1, 2, ... \right] \right\} \\ &a_0 \in \Gamma(x_0), x_1 = T(x_0, a_0) \end{aligned} \tag{A.2}$$

Here the action is chosen as a_0, which will cause the new state $x_1 = T(x_0, a_0)$ at the time 1. The whole future decision problem from time 1 on inside the square brackets on the right will be affected by this new state.

In Equation A.2 what is inside the square bracket on the right means the value of time 1 decision problem, beginning with the state $x_1 = T(x_0, a_0)$. It can be simplified by the notation $V(x_1)$:

$$\begin{aligned} V(x_0) &= \min_{a_0} \{F(x_0, a_0) + \beta V(x_1)\} \\ a_0 &\in \Gamma(x_0), x_1 = T(x_0, a_0) \end{aligned} \tag{A.3}$$

Or even further

$$V(x_0) = \min_{a_0} \{F(x_0, a_0) + \beta V(T(x_0, a_0))\} \tag{A.4}$$

Equation A.3 and A.4 are the Bellman equations. They try to solve the optimization problem by finding the unknown function $V(x)$. $V(x)$ is the value function of the state x and describes the best possible value of the objective. In each state the function $a(x)$ can be found out. $a(x)$ is the policy function and describes the optimal action of a state.

Bibliography

[ars] *ARS 300 Long Range Radar Sensor 77 GHz*. http://www.conti-online.com

[BFMM07] BADINO, Hernán ; FRANKE, Uwe ; MESTER, Rudolf ; MAIN, Frankfurt A.: Free Space Computation Using Stochastic Occupancy Grids and Dynamic Programming. In: *In Dynamic Vision Workshop for ICCV*, 2007

[BFP09] BADINO, H. ; FRANKE, U. ; PFEIFFER, D.: The Stixel World - A Compact Medium Level Representation of the 3D-World. In: *Lecture Notes in Computer Science* 5748 (2009), S. 51. http://dx.doi.org/10.1007/978-3-642-03798-6_6. – DOI 10.1007/978–3–642–03798–6_6

[BMW] BMW: *BMW Group Fahrerassistenzsysteme. Mehr Komfort, mehr Souverï£initï£it, mehr Sicherheit*. http://www.thinkingcars.com/images/BMW_German_Document.pdf

[Bos] BOSCH: *ABS - A Success Story*. http://www.bosch.com/assets/en/company/innovation/theme03.htm

[CG05] CERRI, P. ; GRISLERI, P.: Free Space Detection on Highways using Time Correlation between Stabilized Sub-pixel precision IPM Images. In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, S. 2223 – 2228

[Cor08] CORSARO, Angelo: OpenSplice DDS / PrismTech. 2008. – Forschungsbericht

[Cra] *Vehicle Safety Past, Present And Future*. http://www.aesvn.org/resources/new-car-safety.pdf, Abruf: 20.09.2011

[csf] *Continental integrates driver assistance*. http://www.conti-online.com

[Eff09] EFFERTZ, J.: *Autonome Fahrzeugführung in urbaner Umgebung durch Kombination objekt- und kartenbasierter Umfeldmodelle*. 2009 http://books.google.de/books?id=WfDuSAAACAAJ

[Elf87] ELFES, A.: Sonar-based real-world mapping and navigation. In: *Robotics and Automation, IEEE Journal of* 3 (1987), june, Nr. 3, S. 249 –265. http://dx.doi.org/10.1109/JRA.1987.1087096. – DOI 10.1109/JRA.1987.1087096. – ISSN 0882–4967

[Foe00] FOESSEL, Alex: Radar Sensor Model for Three-Dimensional Map Building. In: DOUGLAS W. GAGE, Howie M. C. (Hrsg.) ; STEIN, Matthew R. (Hrsg.): *Proc. SPIE, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII* Bd. 4195, SPIE, November 2000

[FSO] *Federal Statistical Office of Germany*. http://www.destatis.de

[HMMT] HØILUND, Carsten ; MOESLUND, Thomas B. ; MADSEN, Claus B. ; TRIVEDI, Mohan M.: *Free Space Computation from Stochastic Occupancy Grids Based on Iconic Kalman Filtered Disparity Maps*

[HTA09] HAUTIÈRE, Nicolas ; TAREL, Jean-Philippe ; AUBERT, Didier: Free Space Detection for Autonomous Navigation in Daytime Foggy Weather. In: *Proceedings of IAPR Conference on Machine Vision Applications (MVA'09)*. Yokohama, Japan, 2009, S. 501–504. – http://perso.lcpc.fr/tarel.jean-philippe/publis/mva09.html

[Mat96] Matsunaga, Katsuya: Insufficient headway and unforeseen greater stopping distance as combined factors in traffic accidents. (1996), August

[OIC08] *World Motor Vehicle Production by Country: 2007-2008*. Version: 2008. http://oica.net/wp-content/uploads/all-vehicles-2007-2008.pdf

[PS04] Peden, Margie ; Scurfield, Richard: *World Report on Road Traffic Injury Prevention*. Version: 2004. http://www.who.int/violence_injury_prevention/publications/road_traffic

[Rie10] Rieth, Dr.-Ing. Peter E.: *Derzeitiger Stand der intelligenten Fahrerassistenz*. http://www.fsd-web.de. Version: 2010

[Rot11] Roth, Prof. S.: Slides of the lecture Computer Vision I / GRIS, TU Darmstadt. Version: 2011. http://www.gris.informatik.tu-darmstadt.de/teaching/courses/ss11/cv1/l2-image-formation-v1.pdf. 2011. – Forschungsbericht

[Sch10] Schilasky, Rex: Visualization and Testing of Software Components using wxWavE / Continental Chassis & Safety Division. 2010. – Forschungsbericht

[Sea] *The man who saved a million lives: Nils Bohlin - inventor of the seatbelt*. http://www.independent.co.uk/life-style/motoring/features/, Abruf: 19.08.2009

[Sie] Siemens: *Speedometer*. http://www.siemens.com/press, Abruf: 13.01.2008

[Ste67] Stein, Ralph: *The Automobile Book*. Paul Hamlyn Ltd, 1967

[SWTS03] Shioyama, Tadayoshi ; Wu, Haiyuan ; Takebe, Masaya ; Shimaoka, Naoya: Segmentation and free space detection using Gabor filters. In: *Proceedings of the 13th Scandinavian conference on Image analysis*. Berlin, Heidelberg : Springer-Verlag, 2003 (SCIA'03). – ISBN 3–540–40601–8, 311–319

[TBF01] Thrun, Sebastian ; Burgard, Wolfram ; Fox, Dieter: *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. 2001 http://www.amazon.com/Probabilistic-Robotics-Intelligent-Autonomous-Agents/dp/0262201623/ref=sr_11_1/105-3361811-4085215?ie=UTF8&qid=1190743235&sr=11-1

[tes] *Photo: Test Vehicle in PRORETA*. http://www.proreta.tu-darmstadt.de

[Tur] *U.S. Patent 912,831*. http://www.google.com/patents?vid=912831

[WHW09] Winner, Hermann ; Hakuli, Stephan ; Wolf, Gabriele: *Handbuch Fahrerassistenzsysteme*. http://ebooks.ub.uni-muenchen.de/17963/. Version: 2009

[Wika] Wikipedia: *Advanced Driver Assistance Systems*. http://en.wikipedia.org/wiki/Advanced_driver_assistance_systems

[Wikb] Wikipedia: *Airbag*. http://en.wikipedia.org/wiki/Airbag, Abruf: 29.09.2011

[Wikc] Wikipedia: *Vehicle Horn*. http://en.wikipedia.org/wiki/Vehicle_horn, Abruf: 03.07.2011

[YMOI08] Yoshizaki, Wataru ; Mochizuki, Yoshihiko ; Ohnishi, Naoya ; Imiya, Atsushi: Free Space Detection from Catadioptric Omnidirectional Images for Visual Navigation using Optical Flow. In: *The 8th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras - OMNIVIS*. Marseille, France, 2008