The Stixel World

A Compact Medium-level Representation for Efficiently Modeling Dynamic Three-dimensional Environments

David Pfeiffer

January 7, 2014

Abstract

The capabilities of machine vision systems are improving constantly. This development is driven by an increasing quality and quantity of the perceived scene information. For those systems that rely on the principle of stereoscopic vision, modern dense stereo matching schemes allow to obtain a depth measurement for almost every pixel of an image. In principle this is good news, but it results in a large amount of measurement data that has to be processed and evaluated in real-time. Typically, the vision algorithms are executed on highly integrated low-power processing units that leave little room for expensive algorithms. This problem is solved by introducing a pre-processing step to extract all required information in advance.

Therefore, this thesis proposes a novel and versatile three-dimensional medium-level representation called the Stixel World to efficiently bridge the gap between pixel-based processing and high-level vision. In this work Stixels are employed to represent the dynamic environment of a vehicle.

In the sense of a super-pixel, each Stixel approximates a certain part of an upright oriented object along with its distance and height. Stixels allow for tracking objects over time and are enriched with additional meta-information like confidence measures or class affiliation. Thus, the Stixel World is ideally suited to serve as the basic building block for today's increasingly complex vision systems.

We presents how to compute the Stixel World in a single unified optimization scheme. In this process, strong use is made of physically motivated a priori knowledge about our man-made three-dimensional environment. The actual Stixel extraction is performed efficiently by relying on dynamic programming, this way guaranteeing to extract the globally optimal segmentation for the entire scenario.

Subsequently, the Stixel tracking scheme is introduced. Kalman filtering techniques are used to precisely estimate the motion state of all tracked objects. Beyond that, a variety of optical flow computation techniques are proposed for estimating two-dimensional image displacements of Stixels between consecutive time steps.

Particular emphasis is put on a thorough performance evaluation. To this end, different comparative strategies are followed which include LIDAR, RADAR, and IMU reference sensors, manually created ground truth data, and real-world tests. The evaluation focuses on different aspects, namely accuracy investigations of the achieved measurement precision, an analysis regarding robustness and reliability properties as well as qualitative and quantitative investigations concerning the reliability of the extracted motion state information of the tracked objects. A subset of the ground truth data used for this evaluation has been made publicly available.

Altogether, the Stixel World is an elementary and extremely compact abstraction of the actual world giving access to the most essential information about the current scene. Objects are approximated in detail and along with a highly precise motion estimate. Hence, as a result of this thesis, the performance of subsequently executed vision algorithms and applications has been improved significantly.

Zusammenfassung

Die Leistungsfähigkeit und Einsatzvielfalt maschineller Sehsysteme nimmt stetig zu. Dies resultiert aus einer steigenden Qualität und Quantität der Daten über die unmittelbare Sensorumgebung, wie sie dank moderner Messverfahren zur Verfügung stehen. Heutzutage ermöglicht die Verwendung von Stereoskopie die Bestimmung einer Tiefenmessung für nahezu jeden Bildpunkt. Prinzipiell ist dies positiv zu bewerten, doch im Hinblick auf hoch integrierte Systeme, wie sie in kommerziellen Produkten zum Einsatz kommen sollen, ergeben sich hieraus auch neue Herausforderungen. Oft gelten für derartige Systeme strenge, mitunter auch baulich bedingte Vorgaben zu leistungsbestimmenden Faktoren wie Rechenkapazität, Bandbreite und Verlustleistung. Es ist daher nicht zielführend, wenn bei einer Vielzahl parallel ausgeführter Algorithmen dieselben Daten mit teilweise überlappender Zielstellung analysiert werden müssen.

Zur Lösung des Problems schlägt diese Arbeit die Verwendung einer Zwischenschicht zur Beschreibung dreidimensionaler Szenen vor. Diese sogenannte Stixel-Welt ist eine generische Abstraktion der Rohdaten des Sensors und der daraus zu extrahierenden Szenenmerkmale. Die Stixel-Welt bildet daher den Grundbaustein für zunehmend komplexer werdende Sehsysteme, ohne dabei jedoch zu anwendungsspezifisch zu werden.

Im Sinne eines Super-Pixels repräsentiert jeder Stixel individuell einen bestimmten Teil eines Objektes im Raum. Stixel eignen sich für das Tracken von Objekten und können darüber hinaus mit zusätzlichen Informationen attributiert werden. In dieser Arbeit werden Stixel speziell für die Repräsentation dynamischer Fahrzeugumgebungen verwendet.

Wir stellen ein Verfahren vor, um die Stixel-Welt mittels dynamischer Programmierung in einem einzigen globalen Optimierungsschritt in Echtzeit zu extrahieren. Dieser Prozess wird durch eine Vielzahl unterschiedlicher Annahmen über unsere von Menschenhand geschaffene Umgebung gestützt.

Darauf aufbauend wird ein stixelbasiertes Verfahren zur präzisen Bewegungsschätzung anderer Objekte vorgestellt. Für diese Zielstellung wird die Eignung unterschiedlicher Methoden zur Bestimmung optischer Flussfelder diskutiert. Darüber hinaus präsentieren wir ein neuartiges, auf dem KLT-Prinzip basierendes Verfahren für die Verschiebungsschätzung zweidimensional verhältnismäßig groß ausgedehnter Flächenmerkmale.

Die Arbeit stellt umfangreiche Bewertungen der zu erwartenden Leistungsfähigkeit aller vorgestellten Verfahren an. Dafür kommen unterschiedliche vergleichende Ansätze sowie diverse Referenzsensoren, wie beispielsweise LIDAR, RADAR oder hochpräzise Inertialmesssysteme, zur Anwendung. Einige der für die Auswertung verwendeten Daten wurden öffentlich zugänglich gemacht.

Die Stixel-Welt ist eine elementare und extrem kompakte Abstraktion der tatsächlichen dreidimensionalen Umgebung und bietet gleichzeitig einfachsten Zugriff auf alle essentiellen Informationen der Szene. Objekte werden detailgetreu approximiert und ihr Bewegungszustand wird präzise geschätzt. Infolge dieser Arbeit war es möglich, die Leistungsfähigkeit vieler auf der Stixel-Welt aufbauender Algorithmen deutlich zu verbessern.

Acknowledgments

This PhD thesis would not have been possible without the generous help and support of many people. It is thanks to you that the last three years have been challenging and instructive as well as an amazing experience.

First and foremost, I would like to sincerely thank Dr. Uwe Franke. Thank you as you are, for providing your inspiring ideas, your trust in me as well as your continuous support on so many different levels. I have benefited a lot from your experience, enthusiasm and dedication in teaching me how to conduct scientific research.

I am deeply grateful to Prof. Dr. Ralf Reulke for his support over all these years. It is him who gave me the opportunity to engage in this incredibly exciting field of machine vision during the time of my studies as well as during my time as a PhD student. I am particularly thankful for the freedom and the trust that you gave me to explore things with self-determination.

Again, I feel indebted to thank Prof. Dr. Peter Eisert for his cooperativeness and his willingness to function as second corrector.

I would like to thank my friends and colleagues at the Daimler Group Research and Advanced Engineering for starting many fruitful discussions, giving insightful comments and valuable advice. You are a true inspiration and reliability not to drown in such a demanding research environment.

In particular, I want to thank Dr. Stefan Gehrig and Dr. Clemens Rabe for our close collaboration and for making work so much fun. Special thanks are due to Dr. Fridtjof Stein, Dr. Hernán Badino, Dr. Alexander Barth, Nicolai Schneider, Thomas Müller, Annemarie Meissner, Dr. Markus Enzweiler, Matthias Hummel, Heiko Folkerts, Dr. Carsten Knöppel, Dr. Andreas Wedel, Heidi Loose, Friedrich Erbs, Maximilian Muffert and Timo Scharwächter. There is no doubt that I cannot think of a better team to work with than putting all of you guys together in the same place at the same time.

Thanks are due to Felix Eberli, David Stadelmann and Reto Stalder from the Supercomputing Systems AG. Working with you has been absolutely delightful and gave me the opportunity to gain many new insights and perspectives.

Also, I would like to express my gratitude to the co-readers Dr. Stefan Gehrig, Dr. Markus Enzweiler, Dr. Robert Huth, Hauke Hanke, Fanny Hamann and my dear mother Sigrid Pfeiffer.

Last but not least, I am profoundly thankful to my parents and family for their continuous support, encouragement and the chance to pursue my own goals on my own terms. My deepest thanks go to my friends in Berlin for always being there for me. Thanks for your friendship, for giving me strength, for putting up with me and for making me feel comfortable about doing all the things I do.

Contents

Nomenclature

The following nomenclature is not exhaustive, but denotes the most commonly used variables within the elaboration at your hands.
Depending on the local context, a self-contained notation scheme is not always guaranteed as it can lead to a complexity that is higher than required to make a particular statement. However, we were committed to remain as consistent as possible throughout the whole work.
Thus, for the most significant variables the following notation scheme is used:

I	Two dimensional image of a camera. For standard stereo camera systems (and if not stated otherwise) the variable I denotes the left camera image.
$w,\ h$	Size parameters of a two-dimensional matrix, an image, or a region within an image with $w \in \mathbb{N}^+$ as the width and $h \in \mathbb{N}^+$ as the height component.
$u,\ v$	Image coordinates, such that u is the column coordinate and v is the row coordinate with $u, v \in \mathbb{R}$. When used as index to access an image the following range limitation applies $0 \leq u \leq w$ and $0 \leq v \leq h$ and $u, v \in \mathbb{N}$.
v_{hor}	Image row coordinate of the horizon, typically with $0 < v_{\text{hor}} < h$. This value is either computed from the assumption of a flat ground surface or is supplied by an external process.
b	Base length of the stereo camera system given in meter [m].
$f_u,\ f_v$	Horizontal and vertical focal length of the left (rectified) camera given in pixels [px].
$u_p,\ v_p$	Principal points of the left (rectified) camera given in [px].
$\mathbb{D}$	The set of all possible disparity values with $\mathbb{D} \subset \mathbb{R} \cup \{d_{\text{inv}}\}$. In this context, the term 'd_{inv}' denotes an invalid disparity measurement.
D	Depth map (or disparity image) $D \in \mathbb{D}^2$ computed with respect to the left rectified input image I of a stereo camera system.
d	A disparity value $d \in \mathbb{D}$ given in [px]. A disparity denotes the point-to-point coordinate distance between the projections of a three-dimensional world point into the left and right camera image. Its actual value is inverse proportional to the longitudinal distance to the object. If annotated with an

index d_v it denotes a disparity measurement at row v on a certain column u such that $d_v = D(u, v)$. If not stated explicitly, u should emerge from the local context.

$d_{\min}$, $d_{\max}$ These variables denote the minimum and maximum disparity value, such that $d_{\min} = \min(\mathbb{D} \cap \mathbb{R})$ and $d_{\max} = \max(\mathbb{D} \cap \mathbb{R})$. Typically, for our experiments $d_{\min} = 0$ and $d_{\max} = 128$.

s_n Segment as part of a column labeling. Each segment consists of a base point v_n^b and a top point v_n^t in image coordinates, a class assignment c_n and a representative function $f_n(v)$. Hence a segment is given as $s_n = \left(v_n^b, v_n^t, c_n, f_n(v)\right)$.

k A discrete step in time with $k-1$ being the predecessor and $k, k-1 \in \mathbb{N}^+$. If k is attached to an image or a matrix (e.g. I_k or D_k) then the item is valid for this time step. The actual time in seconds is denoted by t_k and t_{k-1}.

Δ The symbol Δ is always used in combination with a variable, such as Δu. This notation denotes a certain range or interval, e.g. in this case the horizontal length and direction of an optical flow vector.

Δt An actual gap in time between the time steps k and $k-1$ with $t_k - t_{k-1} = \Delta t \in \mathbb{R}^+$.

X, Y, Z Capital letters are used to denote 3D world coordinates or estimated states. With regard to a given reference system (e.g. the left camera or the ego-vehicle), X is the lateral, Y is the vertical and Z corresponds to the longitudinal position.

$\dot{V}$ The symbol $\dot{}$ as superscript to a variable V denotes, that this is a derivative value with $\dot{V} = \frac{dV}{dt}$. For instance, that notation is used to encode the velocity estimate $\dot{X}, \dot{Y}, \dot{Z}$ for the motion state of an object or the yaw rate $\dot{\psi}$ of a vehicle.

$\hat{V}$ The symbol $\hat{}$ as superscript to a variable V denotes, that this is a predicted value. For instance, such value $\hat{V}$ is obtained from the prediction of a Kalman filter system.

$^{\text{meas}}$ The additive $^{\text{meas}}$ as superscript to a variable denotes, that this is not a filtered state, but an actually measured value.

$\underline{V}$ Underlined symbols denote vectors of scalar elements $\underline{V}^T = (V_1, \ldots, V_N)$, such as the position and motion states $\underline{X}$ and $\dot{\underline{X}}$ of other objects.

R Similar to an image, a capital symbol can also describe a two-dimensional matrix of scalar elements. The actual matrix dimensions should emerge from the local context.

1 Introduction

The desire for assisted or even autonomous driving, and thus navigating on public roads without the need of human intervention, is currently one of the most prestigious and appealing challenges in machine vision [23, 132, 146, 163, 181]. During the last decades numerous international competitions have been held with remarkable success and commitment both by universities and industrial partners. Well-known events in this context were the EUREKA Prometheus Project [90] from 1987 to 1995, the DARPA Grand Challenge in 2004 and 2005 (for surveys see [102, 103, 211]) or the DARPA Urban Challenge in 2007 ([30, 41, 108, 148, 153, 178, 216]).

The purpose of such events is to improve technical skills and to foster scientific exchange. It was through these competitions that our knowledge of the challenges of performing information retrieval, using multiple sensor sources, and understanding complex traffic situations has been furthered.

Nonetheless, in order to achieve the goal of fully automated driving, there is still a lot to be done. However it is not in small part due to the advances made at the above mentioned competitions that driver assistance systems recently progressed at a tremendous rate.

Today advanced driver assistance systems are available across all major automotive brands and offer a wide variety of security and comfort features. Typically their feature list includes emergency braking, automated distance control (adaptive cruise control or ACC), lane keeping assistance, traffic sign recognition, intelligent headlight control, automated parking and many other features.

In a few limited driving situations, the technological advances already enable autonomous driving. For example the state-of-the-art traffic jam assistant systems (e.g. the research project "Autopilot" by BMW [29] or the Traffic Jam Assistance by Daimler [50]). With the aid of these systems the car automatically performs all required acceleration, braking and steering maneuvers. Certainly, this functionality is designed for comfort and underlies severe environmental as well as legal restrictions, but distinctively marks a proof of concept. Gaining further experience in this field of research unquestionably leads the way to even more powerful systems.

In this process, it has been realized that one of the key elements for advanced driver assistance systems is to anticipate imminent future events. Faultless operation is crucial, and safely intervening with the driving process calls for a comprehensive awareness of all substantial aspects of the three-dimensional vehicle environment.

Again, this requires suitable sensors providing the measurement data needed to extract all task-relevant information. Available sensor technologies, such as RADAR, LIDAR or cameras are widespread for this task and, naturally, all these technologies have their individual assets and drawbacks.

RADAR, for instance, is probably one of the best-engineered sensor technologies avail-

able. However, the total sensor unit production costs, the working principle (preference for metal reflectors) as well as the achievable richness of the measurement data manifest clear boundaries for its applicability.

LIDAR sensors are particularly characterized by their outstanding measurement accuracy and low failure rates. Yet, they do not allow for a compact construction and are overly expensive. In addition, with regard to real-time applicability, they are often limited in angular resolution.

Using cameras for stereoscopic vision is not a young area of research either, but in the last years has made a remarkable leap. This development is being advanced by recent progress in the consumer electronics market, such as for mobile devices and digital cameras. As a consequence of technological achievements in this field, extremely compact and high-quality sensor devices are available in large amounts at very low cost. Currently, this technology for spatio-temporal cognition is on the doorstep for integration into many safety-related driver assistance systems.

Yet, having qualitative imaging sensors alone does not suffice, because reliably extracting environmental information from an image data stream is a computationally costly and algorithmically complex undertaking. Especially with respect to highly integrated and distributed systems with limited computational resources and bandwidth, a thought-out system architecture as well as clear-cut data processing chains are the basic prerequisites for future-proof vision systems.

1.1 Motivation

Today, modern dense stereo matching schemes allow for obtaining a disparity and thus a distance measurement for almost every pixel in the image. For example, such a dense stereo depth map (also referred to as disparity image) is illustrated in Figure 1.1. Thus, for a common stereo image pair with $1024 \times 440\,\mathrm{px}$ this method results in over $400,000$ single disparity measurements. With regard to real-time capability (i.e. 25 frames per second or higher) the resulting data rate just for the raw depth information sums up to more than $20\,\mathrm{MB}$ per second.

Recently, for vision systems deployed in the automotive domain the total number of different vision tasks demanding to access these three-dimensional data is growing steadily. As a consequence of this development the system complexity increases. At the same time, this development entails an increase of the computational effort and hardware requirements.

Typically, in reference to classic system architectures, every vision task is responsible for extracting all required information individually. Unfortunately, this strategy requires providing the input data to every sub-module separately and, beyond that, holds the risk of identical computations being carried out multiple times.

Obviously, in terms of integrated systems, this practice is inefficient and therefore not conducive, because most of those integrated end-product platforms are highly restricted in terms of processing power, system memory, bandwidth and power consumption.

Hence, approaching this problem demands a more structured design of the system

Figure 1.1: A stereo depth map encodes the actual depth of every pixel. Red represents points that are close and green denotes those that are far away. Only a few pixels (e.g. at the left image border or at left edges of objects) are without a depth measurement. Typically, this is due to left-to-right occlusion or failed depth consistency checks.

architecture and thus requires a re-organization of the data processing chains. A possible way to achieve this is to build on a medium-level representation that easily gives access to all essential scene content and task-relevant information.

Reasonably, such a representation should be extracted beforehand in a pre-processing step in order to be of use for all subsequent vision tasks. With regard to driver assistance this representation could include useful information such as freespace data, object and obstacle information, their motion states and many other data. Furthermore, to ensure efficient and universal usage, it should meet each of the following properties:

- Compactness: The representation allows for a significant reduction of the data volume. Yet, at the same time, all task-relevant information is preserved.
- Stability: Small temporal changes of the underlying input data (e.g. the stereo depth map) do not cause rapid changes within the resulting representation.
- Robustness: Outliers in the input data have very little or even no impact on the representation result.
- Explicitness: All information of interest is available without the need for data re-organization. Accessing the data directly must simply yield all significant object characteristics without the need for additional computation.

Having such a representation at hand clearly enables a more structured design of increasingly complex vision systems and, at the same time, allows for a more efficient use of available hardware resources.

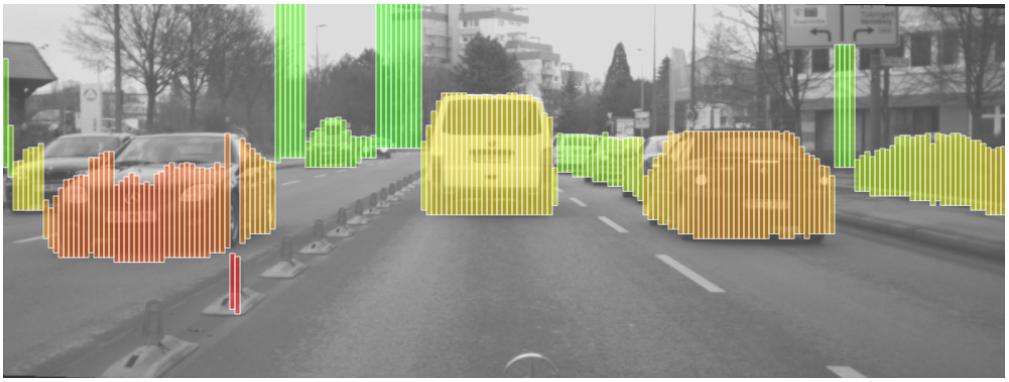


Figure 1.2: The Stixel World as initially published in [9]. Every Stixel represents a certain part of the first obstacle to encounter along that particular viewing direction. The Stixel coloring encodes the distance to the object. Red means close and green is far away. Stixel boundaries are marked white.

1.2 The Stixel Primitive

To tackle this challenge, we propose a novel medium-level representation called the *"Stixel World"* that is used to efficiently model the scene content of arbitrary three-dimensional urban traffic environments [9]. While the cited work was done prior to this thesis, in this work we suggest a more evolved scheme that grants substantially more flexibility and detail with respect to the obtained scene reconstruction result.

The space in front of the camera is split into two adjacent regions: horizontal freespace up to the base point of the first obstacles and a set of Stixels approximating that obstacle. A single Stixel is defined as a thin and fronto-parallel rectangle with a fixed pixel width and vertical pose. Originally, it is assumed earthbound and thus is described by just two parameters, namely a disparity (i.e. distance) and a height value. In doing so, this representation achieves an enormous reduction of the input data volume of about half a million disparity measurements to a few hundred Stixels only, while encoding freespace and obstacle information for the whole scenario. An exemplary result of this proceeding is depicted in Figure 1.2. Note that every Stixel is valid for 2D and 3D, since it describes an object within the image space as well as in real-world coordinates.

The Stixel representation is characterized to be compact, robust to outliers and easy to access. According to the initial approach, the Stixel World is constructed by cascading multiple independent algorithms: mapping disparities to occupancy grids, a freespace computation, a height segmentation, and a final Stixel extraction step. However, such a cascade is prone to errors, e.g. missed objects in the freespace computation can not be corrected in subsequent steps. Further, the proposed scheme contains multiple thresholds and nonlinearities (e.g. a height constraint when creating the occupancy grid). Above all, taking into account only the first obstacle along every viewing angle can cause missing relevant objects (e.g. a pedestrian standing behind the engine hood of a car).

To circumvent these limitations, we propose an extension of the Stixel data type.

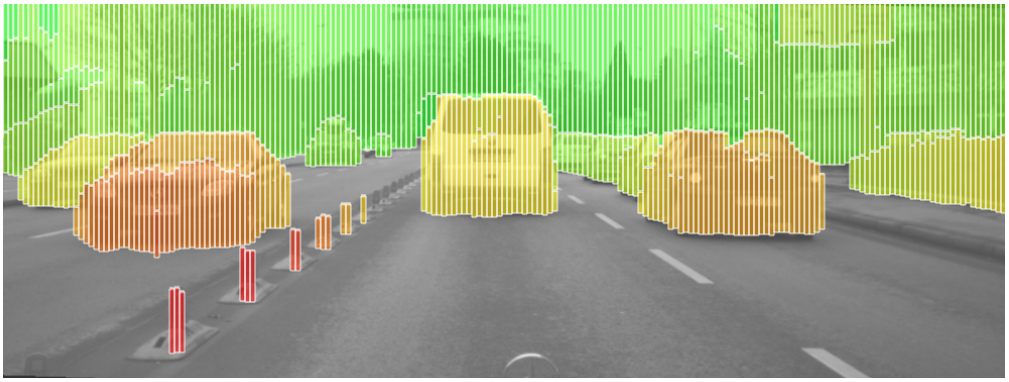


Figure 1.3: The Stixel World as output of our optimization approach. The captured scene is segmented into piecewise planar Stixel segments that correspond to either ground or object. Compared to Figure 1.2, this approach allows for segmenting multiple objects within the same column.

Hereby, the constraint for the Stixels to touch the ground surface is dropped and we also allow for multiple Stixels along every column of the image. These changes enable to precisely represent arbitrary sets of objects located at different depths within the scene.

Accordingly, due to this characteristic, this representation is referred to as the multi-layered Stixel World [170]. However, doing so also requires severe changes in the Stixel computation scheme. Therefore, we propose a novel technique for extracting the Stixel World. Given a dense stereo depth map and physically motivated world model priors, our approach yields the most probable and thus optimal Stixel representation. An exemplary result of our method is depicted in Figure 1.3.

In addition to encoding the location of the objects within the three-dimensional scene, the Stixel representation may be easily enriched with further information. This includes motion properties as well as type related attributes (e.g. class information such as *pedestrian, car, traffic sign* or similar) or further meta data.

The Stixel representation is not limited to a specific field of application. Due to its striking simplicity, it is well suited for arbitrary purposes. For instance, it is straightforward to use in terms of driver assistance, robotics, navigation, global information systems (GIS), mapping or even augmented reality tasks.

1.3 Organization of this Thesis

This thesis is organized as follows: To begin with, Chapter 2 discusses related work. Since the field of computer vision is a fairly wide and active topic, this part focuses on the most relevant contributions of other research groups from different scientific domains with overlap or similarities to our methods and objectives.

Within Chapter 3, the novel Stixel computation scheme is presented that allows for extracting arbitrary staggered Stixels by means of a unified optimization approach. For estimating the motion properties of other objects, Chapter 4 outlines the required motion-

state estimators as well as the corresponding filter design. Further, we discuss different techniques for determining Stixel correspondences between consecutive time steps.

In order to rate the quality of the extracted scene and object data, Chapter 5 addresses different strategies for both quantitative as well as qualitative evaluation procedures. To this end, we present performance results by making use of different reference sensors, manually annotated data, simulated ground truth as well as direct comparison against other vision algorithms.

Subsequently, Chapter 6 focuses on the applicability of the Stixel medium-level representation for other vision tasks. To this end, we briefly sketch different use-cases as well as further related work that could largely benefit from relying on Stixels either in terms of control of attention or as direct measurement input.

Finally, Chapter 7 concludes this thesis by addressing open topics and future work.

2 Related Work

Sensors for obtaining three-dimensional scene data are available in a very wide variety. This includes imaging sensors, time-of-flight approaches such as the PMD-camera and LIDAR (light detection and ranging) [52, 102, 103], RADAR (radio detection and ranging) [94, 185], and SONAR (sound navigation and ranging) [54, 157]. For using cameras, stereoscopic vision [36, 183, 184], structure-from-motion (SFM) [83, 126], structure-from-texture (SFT) [3, 87, 105, 226] and shape-from-shading [96, 106] as well as active-light approaches [5, 42] are very popular.

For this thesis, the primary focus is on processing dense stereo depth maps computed from standard stereo image pairs. Such a stereo setup consists of two cameras that are mounted to a bar with a defined base line. The camera's optical axes are oriented in parallel pointing into the same direction. A schematic view of this setup is illustrated in Figure 2.1.

Working With 3D Data Processing stereo image pairs has quite a long history [36, 184]. Starting with a few measurements per image, today modern stereo matching algorithms allow to compute a depth estimate for nearly every pixel of an image.

The resulting data volume has to be processed in order to extract the task-relevant three-dimensional scene content. When using stereo cameras for building highly integrated systems (e.g. for advanced driver assistance), the obtained information needs to be distributed to all further processing tasks. Since the latter are typically executed on efficient low-power industrial hardware with limited computational resources, one can not afford to evaluate and interpret this three-dimensional point data independently and thus repeatedly. For this reason, a dedicated pre-processing step is required to extract

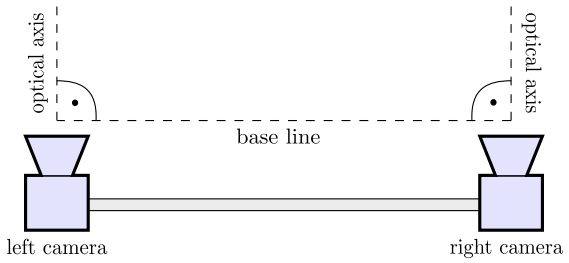


Figure 2.1: This illustration schematically shows a standard stereo camera setup. Both cameras are mounted to the same bar at a fixed distance. This distance is referred to as base line of the stereo rig. The camera's optical axes are oriented in parallel pointing into the same direction.

task-relevant information and subsequently make it available using a compact and easy-to-handle medium-level representation.

Even though the variety of existing schemes for this purpose is enormous, they commonly rely on models in order to abstract from the real world and thus intend to achieve a reduction of the data volume. Ideally, these models reflect all task-relevant information and thus allow to serve as input for further processing. The models themselves can be categorized with respect to numerous criteria. Thus, a brief overview is provided in the following.

Probably the most significant characteristic concerns the model's orientation that defines in which layer or extend the real world is approximated. For instance, horizontal approximations of the scene commonly rely on two-dimensional grid-based structures. Object representations, however, such as for cars, pedestrians or infrastructure, often use vertical and thus upright-oriented models that put emphasis on slanted continuous surfaces. Apparently, the individual vision task determines the underlying model decisively. Thus, for example, implementing a curb detection using a vertical instead of a horizontal model is not likely to yield satisfactory results.

Secondly, the strategy how information is contained differs significantly. E.g. certain approaches describe scene content using explicit object lists while other methods, such as most map-based approaches, provide their information stochastically. Furthermore, certain representations (such as level-sets) allow for functional data access. There are many more ways to provide scene information but, obviously, the choice for a specific scheme is mainly determined by those subsequent processing steps that depend on the representation as computational input. Regarding the type of contained information, a difference is made between spatial and temporal information. Within the scope of advanced driver assistance, especially the latter is desirable to have, because knowledge about the motion state of other objects is an essential aspect for a full and comprehensive understanding of the three-dimensional environment.

Thirdly, an important characteristic regards the actual processing chain until a final representation is extracted. For this objective, certain methods rely on a cascade of multiple, mostly independent processing steps while other methods rely on unified schemes. Also, one has to distinguish between greedy approaches and those that rely on some sort of optimization technique. For this purpose, minimization of an energy or a cost functional as well as maximization of an a posteriori likelihood (MAP) are increasingly popular. With regard to optimization, a distinction has to be made between those methods that either explicitly guarantee to find a global and thus unique optimum solution and those that only yield an approximation.

Fourthly, for this overview, the last distinction is made with respect to regularization and a priori constraints. In this context, expert knowledge of the three-dimensional world (e.g. that objects are likely to have vertical pose) is incorporated directly within the scene reconstruction process. Typically, that procedure allows to improve or stabilize the reconstruction quality significantly and thus is a recurring key idea of many three-dimensional reconstruction methods.

Dense Stereo Matching As a matter of fact, the majority of structures in man-made environments consists of piecewise smooth and planar surfaces that have either horizontal or vertical orientation. Reasonably, such prior knowledge should be incorporated in the reconstruction process at an early stage, such as done in terms of the dense stereo scheme semi-global matching (SGM) that has been proposed by Hirschmüller et al. [88]. SGM works in a dynamic programming (DP) [22] fashion. Slight disparity changes are penalized with rather small and constant costs by which the reconstruction of continuous, slanted surfaces is preferred. To allow for depth discontinuities at the same time, SGM also limits the maximum cost penalty if a larger jump in disparity is required.

Another stereo scheme that by design massively leverages three-dimensional surface planarity is plane sweeping stereo [43]. Discriminative-oriented planes are slid through the image space while computing disparity votes in a correlation-based fashion [104, 184]. A special feature of that technique is the capability to deal with unrectified images. An extension for that scheme has been presented by Gallup et al. [70]. The authors incorporate prior knowledge about the location and orientation of those planes directly into the reconstruction process. Further, dedicated work has been done in terms of directly supporting slanted surfaces for disparity estimation [14, 27, 239].

Other stereo schemes that allow for the computation of dense depth maps and to this end rely on the principle of energy minimization are belief propagation stereo (e.g. [202, 232, 229]), total variation stereo (e.g. [147]), and dynamic programming stereo (e.g. [164, 219]). Additionally, certain correlation stereo-based methods allow to compute dense depth maps (e.g. [158, 237, 238]).

Focused on stereo cameras and guided by the Middlebury database [183], in 2006 we found SGM to be the best-performing and thus most promising stereo matching scheme in terms of our objectives. When choosing the right matching criterion [89, 155], SGM proved to be robust against differences in local as well as global illumination and thus is well suited for automotive application. Even though SGM is computationally expensive, Gehrig et al. [73] have proposed a variant of that scheme allowing for real-time computation using efficient low-power FPGA hardware. If using specialized FPGA hardware is not an option (e.g. due to high development costs), an efficient scheme of the SGM computation on a standard CPU (central processing unit) has been proposed by Gehrig et al. [74]. Alternatively, variants that utilize the GPU (graphics processing unit) have been suggested by Ernst et al. [58] or Haller et al. [82].

Irrespective of the actual stereo matching scheme, the amount of data as output of such a system is enormous. Consequently, suitable algorithms are required to process this data accordingly.

Point Accumulation The idea of point accumulation is to collect many three-dimensional point measurements from different views in a sole and unified representation. A popular way to do this is accumulating and fusing three-dimensional point data over time. When working with mobile platforms, the most challenging part is the ego-motion estimation between consecutive time steps. Additionally, if not compensated appropriately or masked by some kind of intelligent pre-processing step, most approaches have

difficulties dealing with dynamic scene content.

The final output of such schemes is quite manifold. Certainly, the most straightforward way is to present the information just the same and identical to the input data as three-dimensional point clouds, such as done by Geiger et al. [75]. Their approach matches two consecutive image pairs (i.e. four frames) in real-time and fuses the reconstruction results accordingly. Also, Kitt et al. [112] have shown a very similar scheme. Here, the authors rely on a trifocal sensor geometry between image triples (i.e. the current left and right as well as the previous left image). Using an iterated unscented Kalman filter [221], the algorithm allows for precise frame-to-frame computation of the vehicle odometry and thus for a subsequent fusion of the depth map results. Note that similar to these approaches, plenty of work has been done in the field of visual odometry [7, 67, 99, 131] or SLAM (simultaneous localization and mapping) [123, 125, 128, 210].

In recent years, the idea of using arbitrary photographs (and thus pictures with an unknown camera geometry and orientation) for three-dimensional scene reconstruction has become very popular under the slogan *"Photo Tourism"*. For this task, it is common practice and considered a challenge to process publicly available pictures from the Internet. By now, different approaches exist with impressive results [1, 35, 65, 182, 194].

Nevertheless, up to this point, the main objective has either been an ego-motion estimation problem or a camera calibration and pose orientation task. The actual depth information itself, however, was accumulated in a raw unprocessed form and a reduction or interpretation of the three-dimensional data volume was not targeted. In certain cases, the aggregation of three-dimensional point clouds is often accompanied by a triangulation step for extracting polygon surfaces of the scene geometry. For this purpose, numerous approaches exist that show promising results.

Triangulation and Mesh Building The idea of mesh building is to create a polygonal representation from point-based input data. For instance, Koch et al. [113] show an approach for extracting high-resolution object mesh models from sequences of uncalibrated images that offer different views on the captured objects. Using the technique proposed in [21], the authors compute the camera calibration (which includes the internal calibration as well as the relative position and orientation of the camera) by tracking point features throughout the sequence. Using the estimated calibration and camera poses, they perform a rectification step on consecutive images. Subsequently, the authors compute dense depth maps by using a variant of the stereo matching scheme proposed in [48] that makes strong use of ordering and uniqueness constraints. However, for the whole approach to work, prior three-dimensional scene knowledge is neither incorporated nor required.

Secondly, the work of Micusik et al. [150] targets a GIS (geographic information system) application. While driving trough an urban environment, their cascaded bottom-up approach constructs three-dimensional mesh models incrementally over time. For this task, the authors rely on 360 degree panoramic images. The construction process is supported by making strong use of so called *"L-shape"* and planarity priors regarding the three-dimensional world.

Last but not least, Prankl et al. [175] present a system that aims at combining the advantages of grid-based and point-based representations. The authors suggest an incremental mesh-building system that tracks multiple planes across consecutive images. Planes are selected using randomly sampled and weighted interest points. Once detected, each plane serves as prior for creating new planes in subsequent images.

Certainly, these computer-graphics oriented approaches obtain detailed and visually pleasing results. However, in terms of an automotive application, they also come with a couple of drawbacks. Typically, these mesh models require multiple views to be built up. Their inherent irregular structure complicates the handling process significantly. Thus, they are hardly usable for automotive application or similar tasks that put emphasis on fast detection rates and versatility. Further, by just having a three-dimensional mesh model, the contained information is not interpreted in any way. This means that for instance the inference of occupied and free regions can not be performed without further processing of this data. Apart from that, these polygon grids do not provide motion state information about the contained objects and the proposed methods are often not tested with respect to robustness.

Feature Based Motion Estimation To tackle the issue of object motion estimation, Franke et al. [66, 176] have presented a point-based tracking scheme called *"6D-Vision"* that allows to independently track the position and motion state of sparse image point features. Those are selected by a non-maximum suppression using the Harris corner detector [84]. In order to decide which features to track, their quality is ranked using the eigenvalues of their small neighborhood patches [213, 214]. Temporal correspondences between consecutive time steps are resolved using the well known KLT-feature tracker [137], and the motion state itself is estimated using an extended Kalman filter [225]. Hereby, the motion model assumption is to have a constant velocity state. Yet, their approach can be extended easily to estimate higher order components as for instance acceleration. While 6D-Vision started with a few thousand point features, Müller et al. [159, 177] have extended that scheme to cope with dense input data. They allow for a motion estimate for almost every pixel of the image.

Further, under the designation *"Scene Flow"*, numerous approaches aim to estimate three-dimensional image motion fields using global energy minimization schemes. While Huguet et al. [100] and Wedel et al. [224] rely on point features for that task and thus estimate the image displacement and change of depth for consecutive frames, Popham et al. [172] suggest to do the same thing for small planar surface patches. These patches are selected using the method proposed in [81] and tracked in the presence of a smooth-motion prior. Their approach computes a translation and rotation estimate for every patch individually.

Up to this point, except for the mesh-building solutions, no reduction of the data volume was performed. In fact, for most of the discussed techniques the opposite is the case as they aimed at extracting further information such as object motion states. In addition, so far, all considered approaches are feature-based and (except for minor aspects such as the patch condition proposed in [172]) do not offer inherent support for

regularization (e.g. topology arrangement priors or model orientation constraints).

Grid-Based Approaches In the following, the focus is redirected on work that puts emphasis on layered world representations starting with map-based structures.

Map-based structures are used for representing three-dimensional environments horizontally. To this end, mapping depth information to the cells of digital elevation maps [119, 121] or 2D/3D occupancy grids [54, 156, 210] is probably the most common practice for modeling either the height of the ground surface or the likelihood of the environment to be occupied. The resulting grid information is utilized further within the scope of the most diverse processing task, e.g. for extracting scene attributes such as freespace [8, 94, 223], obstacle information [9, 129, 167, 233] or the location of curbs and sidewalks [165, 174, 191]. In [133], an approach is presented that uses probabilistic occupancy grids for self-localization in three-dimensional environments. Other approaches employ occupancy grids in terms of autonomous navigation [4, 54, 209].

Apparently, one of the most striking benefits of using such grids is their usability and thus versatile field of application. In addition, using grids allows to directly consider sensor-specific noise and failure characteristics. For this reason, they are best suited to perform multi-sensor data fusion [2, 144, 162, 199].

Certainly, grid-based representations also entail certain drawbacks themselves. For instance, their finite spatial resolution implicitly leads to discretization effects. Aside from that, environmental scene information in terms of object information or similar is (with a few exceptions [33, 51, 68]) represented non-explicitly, such that using grids unavoidably implies to require further processing steps for their interpretation. Along with the point-based aggregation schemes, difficulties occur when dealing with dynamic scene content and temporal map integration techniques [134, 166, 218].

Beyond that, another downside concerns their memory requirements, especially when using three-dimensional grid structures to represent large-scale territories. Specifically this aspect is targeted by the work of Schmid et al. [185] or Wurm et al. [230]. Their approaches focus on three-dimensional probabilistic occupancy grids with the main objective to be efficient with respect to memory consumption. Hence, they propose to rely on dynamic grid structures by using a sophisticated level-of-detail octree system. In return, they provide spatial resolution specifically when needed and keep the memory consumption low otherwise. Nevertheless, due to an increased structural complexity, accessing data is no longer a plain and atomic operation while registering measurements occasionally requires data re-organization of the grid. Consequently, both operations are connected with additional computational effort.

Within the scope of tracking objects, Brechtel et al. [33] and Danescu et al. [51] have presented two grid-based approaches. Both schemes rely on probabilistic occupancy grids and use particle filtering [53, 208] to infer object motion states. For this purpose, Brechtel et al. equip a 360 degree LIDAR scanner, and Danescu et al. rely on dense stereo depth maps.

Layered Representations Exploiting environmental regularities does not end with the plain occupancy grid result. Thus, in the following, selected object detection schemes are discussed that rely on grid maps as an auxiliary pre-processing step. A good example for such a proceeding is the original approach to extract the Stixel World as we proposed prior to this work in [9]. However, as this is the central topic of this thesis, that approach is discussed separately and in more detail in Chapter 3.

An approach closely related to the Stixel principle has been published by Gallup et al. [71] with the objective to create 3D volumetric object models. Multiple depth maps from different views are accumulated in a single Cartesian histogram-based elevation map. Thereafter, each cell is split into alternating *empty* and *occupied* vertical box volumes. By relying on DP, the authors achieve an optimal segmentation for every cell of the grid. Apparently, their approach is a combination of both, a horizontal model to build the elevation map and a vertical model for the object extraction step. Further note that the work presented by Wurm et al. [230] is related to Gallup's approach.

In [192] we have introduced a method for curb detection using stereo vision and thus rely on a digital elevation map (DEM). The curb is modeled as a third order polynomial to separate road from sidewalk. For this purpose, the obtained DEM is projected to a conditional random field [122, 204, 205] with the goal to assign its nodes to the class labels *road*, *sidewalk* and *outlier*. The curb itself is extracted in an iterative scheme by repeatedly using a surface estimation for the ground models (quadratic surfaces for both sidewalk and road) and a loopy belief propagation [138, 205] step for inference of the class affiliations. The approach yields good results, but, on the downside, is computationally very expensive. For this reason, Siegemund et al. [191] modified this scheme to use an extended Kalman filter for tracking the curb parameters over time. As a result, the achieved result is stabilized and the labeling step allows to be initialized more efficiently and thus requires significantly less iterations to converge. Another scheme for curb detection that uses a DEM has been proposed by Oniga et al. [165]. Hereby, the authors use edge-based cues extracted from the DEM to determine their model parameters. However, they do not rely on a global scheme for the curb estimation.

In the field of remote sensing, Baillard et al. [11, 12] have presented a multi-step method for reconstructing piecewise planar three-dimensional models of urban areas from multiple aerial image views. A DEM is built up by using the cross-correlation based stereo approach suggested in [48]. Then, using a plane sweeping strategy [43], the authors extract lines and three-dimensional data from both the DEM and the images. These lines relate to object boundaries and interact with each other using geometric (e.g. coplanarity) and photometric (e.g. texture) constraints.

Object Modeling In order to represent people or objects explicitly, different procedures are available. These do not necessarily rely on grid maps but suggest other techniques to model objects within the environment. The most common and straightforward one is to utilize three-dimensional box volumes, such as done by Leibe et al. [126, 127] or Ess et al. [59] within the scope of pedestrian detection and tracking. Ideally, these bounding boxes surround the object completely. However, due to the partly complex, irregular and

deformable appearance of these objects (e.g. pedestrians), this box representation is to be considered a comparatively rough approximation. As a result, boxes are convenient to use, but entail severe limitations when it comes to representing objects in detail.

Nevertheless, with regard to vehicles and thus rigid objects, Barth et al. [15, 16, 17, 19] have proposed a bounding box tracking scheme to describe and track vehicles in three-dimensional space. Using the 6D-Vision [66, 176] approach and a sophisticated motion model and filter techniques [13, 28], the authors reliably estimate the location, pose and motion state of vehicles including their acceleration, velocity and yaw rate information.

Further, Leibe et al. [127] present a monocular structure-from-motion based multi-view approach to detect and track objects and pedestrians in a unified scheme. These are detected in two-dimensional image space and converted to three-dimensional observations. For the representation the authors rely on 2D and 3D bounding boxes. For the tracking part a globally optimal set of so called *"spacetime trajectories"* is computed that provides the best explanation for all object evidence collected.

Additionally, level sets are a popular method in order to yield a two-dimensional representation of objects when putting emphasis on contour accuracy within the image [49, 69, 189]. This representation allows for a very flexible description of arbitrarily shaped objects using mathematical formulas that define sets of related pixels. Also, this method allows to describe partitioned regions using a single function. However, this approach also has its downsides. Firstly, the required computational background to find the corresponding function is rather complex. Secondly, level sets are not intuitive to handle, since objects are only described implicitly on a functional level.

With the goal to track objects and vehicles, several level set methods have been proposed [72, 95, 240]. A sophisticated method focusing on contour-based tracking of non-rigid objects (exemplified using sequences with pedestrians) was presented by Yilmaz et al. [234]. The authors rely on a Bayesian motivated inference scheme and thus perform an online training of color and texture models to use those as feature priors for object tracking with an a posteriori expectation maximization technique.

Semantic Segmentation Recently, an incredible amount of work has been done in the field of semantic and appearance-based segmentation of street scenes, mostly with focus on complex urban districts [6, 38, 46, 60, 62, 80, 151, 190, 201, 228]. Definitely, this is not a novel topic (e.g. see [45]). Nevertheless, modern algorithms and techniques (e.g. see [204, 205] or [31, 115]) have led to new perspectives with partly promising results. Thus, in the following, those with the highest affinity for our purpose are discussed briefly.

In [63], Felzenszwalb et al. present a probabilistic image segmentation approach that uses appearance to assign semantic information to certain regions of an image. The image is segmented using a continuous upper and lower bound. The resulting upper part is called *background*, the middle region is assigned to *object* and the bottom region to *floor*. For the segmentation step the authors rely on DP. However, even though they allow for further vertical segmentation of the centered *object* part, their approach is limited to one object per column.

Hoiem et al. [92] have presented a scheme to assign the labels of type *sky*, *vertical* (ob-

ject) and *planar* (ground) to super-pixels generated using the method proposed in [61]. The assumption is that every super-pixel has a unique and thus homogeneous labeling. Their orientation is determined by estimating the vanishing points of the surfaces using the method proposed in [116]. For the labeling itself, the authors rely on a greedy algorithm while exploiting pairwise patch affinities. Further, an appearance-based boosted decision-tree classifier on a trained data set is used to infer the probabilities for the class affiliation.

In [135], Liu et al. use appearance cues to assign semantics to image regions using a *five parts model* (to distinguish between *top*, *left*, *right*, *bottom*, and *center*). The novelty of their approach is to introduce global topological ordering constraints to graph cuts segmentation [32, 78]. However, their used constraints (that basically define allowed arrangements of the used classes) are quite strict and therefore inflexible. For the solving step the authors rely on an extension of the α-expansion graph cuts method [31, 32]. Their approach allows to yield a two-dimensional approximation of an unknown optimum solution. Further, an approach closely related to the work of Liu et al. has been proposed by Kohli et al. [114]. They suggest an extension of the typically used pairwise smoothness potentials as well as a generalization of the P^n Potts model. As a result, they allow for higher order conditional random fields and thus permit to deploy regularization priors of higher magnitude (e.g. global consistency constraints).

Teboul et al. [207] have presented an approach for façade reconstruction using taxonomic grammars. Even though grammars originate from language processing, in this work they are used to constrain the extracted segmentation result in the sense of a vocabulary (i.e. *wall, roof, window, balcony*, etc.) and thus implicitly define a class of allowed reconstruction results. The interesting aspect of this idea is that the authors yield a hierarchical representation of the segmented object while being able to express explicit ordering and semantic regularization priors. For the segmentation step the authors rely on expectation maximization techniques and use dynamic programming [22].

Further, Barth et al. [20] have presented an approach for a probabilistic per-pixel multi-class labeling of traffic scenes. To this end, the authors distinguish between the label types *static background/obstacle*, *ground*, or *moving object*. Prior knowledge on class topological ordering is accounted for by global ordering constraints (similar to [135]). The data input is obtained from scene flow such as proposed by Wedel et al. [224]. In addition, location priors for tracked vehicles are considered by using the tracking output of [19]. Affinities between neighboring pixels are modeled using a conditional random field, and loopy belief propagation is used for inference of the class assignments.

Our Contribution With this background of related work, we suggest using multi-layered Stixels for representing and tracking arbitrary three-dimensional objects in dynamic, real-world environments. For this purpose, we primarily focus on the use of stereo vision. Even though this new Stixel type is a direct extension of our work published prior to this thesis [9], the underlying methods of computation differ significantly [170].

We propose a novel approach for Stixel extraction that makes explicit use of physically motivated world model knowledge. Thus, to a certain extent, our procedure has overlap

to the work of Felzenszwalb et al. [63], Gallup et al. [71], Liu et al. [135], or other scene labeling techniques [6, 20, 60, 93, 151, 207]. However, we neither require multiple views nor prior appearance-based training steps. In contrast to the segmentation procedure of Felzenszwalb or the approach of Liu, our method allows for additional horizontal partitioning of the image into several vertically staggered objects. As a result, it allows to represent the captured scenario in more detail.

Cascading multiple algorithms in a chain of independent processing steps, such as done in the work of Gallup et al. [71], Hoiem et al. [92], Micusik et al. [150], Prankl et al. [175], or in our previous work concerning Stixel computation [9], has turned out to be an error-prone venture. Consequently, we suggest to perform both the freespace computation and Stixel extraction in a sole, unified scheme.

Aside from working with stereo data, the proposed approach is also simple to use with other sensor types (e.g. sparse LIDAR data [85, 170]). Furthermore, it is best suited to be used in terms of multi-sensor data fusion [2, 144, 162, 199].

Temporal information is taken into account by relying on a Stixel-based tracking scheme. This allows for estimating the absolute motion state of other objects independent of our own motion. For this purpose, we closely follow the Kalman-filter based working principle of the 6D-Vision approach [66, 176].

In this process, we choose from a variety of optical flow computation techniques, such as a [160, 196] or [213]. Additionally, we propose an extension of the KLT-feature tracker [213] that relies on two-dimensional feature patches. While affine transformation for KLT-based tracking has already been discussed in [143] and [214], the novelty of our method is to combine this idea with pixel-wise depth information. This way, we explicitly exploit the fundamental model assumption of the Stixels to approximate rigid and upright oriented surfaces. As a result, our approach leverages available texture information from the input images and depth data from stereo much stronger than comparable methods.

In most work concerning object tracking, the measurement steps and the actual tracking process are carried out independently (cf. [16, 66, 109, 149, 189, 191, 213]). However, with respect to the optical flow computation, we decided to unite both closely. This is achieved by using the Kalman-filter based state predictions in order to support the initialization process of the optical flow scheme. As a result, the performance with respect to large and typically hard to compute optical flow vectors improves. In return, this helps to increase the performance of the object tracking scheme considerably.

Finally, given the fact that the proposed algorithms are applied in the field of mobile vision and driver assistance, concerns such as measurement accuracy, reliability and robustness are of utmost importance. Therefore, instead of benchmarking on images taken in controlled environments [89, 183, 184, 215], it is essential to evaluate new algorithms thoroughly under different practical conditions and on large data sets.

The methodical building blocks of our work, as coarsely outlined above, are explained in more detail in the following chapters of this thesis.

3 The Multi-Layered Stixel World

The following chapter describes the necessary steps in order to obtain the multi-layered Stixel World in detail. Thereby it extends the publication *"Towards a Global Optimal Multi-Layer Stixel Representation of Dense 3D Data"* [170].

The main contribution of this work is a probabilistic approach to compute the *Stixel World* for a stereo image pair in a single global optimization step. In addition, the constraint of the Stixels to touch the ground surface is dropped. The approach allows for multiple *Stixels* along every column of the image. For this purpose, the *Stixel* generation problem as proposed in [9] is altered into a segmentation problem similar to the work of Felzenszwalb et al. [63] or Gallup et al. [71]. Such an exemplary object segmentation is illustrated in Figure 1.3.

To this end, a Bayesian formulation of the segmentation task is deduced, which leads to a maximum a posteriori (MAP) estimation problem. We show how this problem is solved optimally by means of dynamic programming [22]. Numerical concerns are discussed along with particular characteristics of the relation between probabilities and costs. Finally, insight into some of the implementation details is given and possible extensions of the segmentation algorithm are discussed.

The proposed approach is not limited to a certain type of input data or sensor. Instead it can be used for segmentation of measurement input taken from arbitrary 3D sensor sources, such as LIDAR or time-of-flight sensors (e.g. the PMD camera) without applying significant changes to the processing scheme. Additionally, it can be applied to input data in either sparse or dense form. However, since we consider the stereo camera to be the main sensor for the application in driver assistance, the focus lies on using dense depth maps. Whenever the use of other sensors requires to modify the presented approach, we will explicitly point this out it and suggest where and how to apply the particular changes.

3.1 The Original Stixel Generation

In [9], we first presented the Stixel World, a medium-level representation that focuses on providing the first object to encounter along every column of an image. Each Stixel describes a particular area of an object in terms of its base and top point. Altogether, the Stixels encode the three-dimensional content of the scene in consideration of freespace and obstacle information. According to the initial definition of the Stixel primitive, the described obstacle touches the ground surface and is located at the end of the freespace. The Stixel World is applicable for a wide variety of tasks in computer vision, robotics, and driver assistance, e.g. for mapping tasks or as control of attention for object detection and classification.

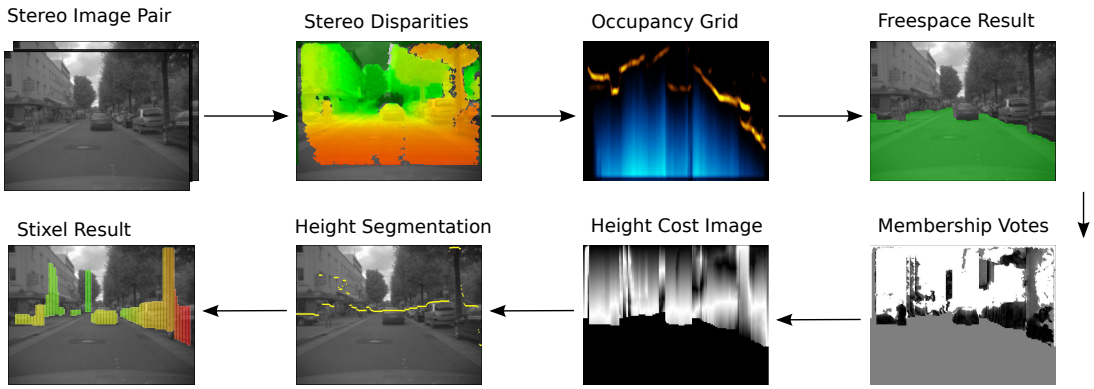


Figure 3.1: Visualization of the construction process for the Stixel World as published prior to this thesis [9]. The Stixel representation is extracted in a cascade of multiple independent algorithms. This includes the computation of a polar occupancy grid, the freespace computation, the height segmentation and the final Stixel extraction.

3.1.1 The Creation Process

According to [9], the Stixel World is constructed from raw stereo data in a cascade of multiple consecutive and independent steps:

1. Stereo disparities are mapped to a polar occupancy grid.
2. The occupancy grid is segmented from left to right in order to extract the freespace information. For this step dynamic programming (DP) [22] is used. The end of the freespace corresponds to the base point for the adjacent obstacle.
3. A membership and a cost image is created, in order to let disparities vote whether they belong to an object located at the end of the freespace or not.
4. The resulting cost image is segmented in a second DP step in order to obtain the upper outline and thus the top point of the first obstacle along every column of the input image.
5. The resulting base and top points for every column are used to extract the final Stixel representation for the three-dimensional scenario.

In order to clarify this procedure, Figure 3.1 visualizes the individual steps. To this end, it illustrates the subsequent intermediate results when extracting the Stixel representation for an exemplary urban scenario. In order to find a more detailed description of this procedure, we refer to [8] for the freespace computation and to [9] for the described Stixel extraction steps.

3.1.2 Discussion of the Original Approach

Even though the proposed procedure yields good results on a wide variety of scenarios it also has a few inadequacies and drawbacks.

Typically, a cascade as sketched in the previous section is prone to errors, e.g. missed objects in the freespace computation can hardly be corrected in subsequent steps. Further, this scheme contains multiple thresholds and nonlinearities. For instance, the decision whether a disparity measurement is registered into the occupancy grid or not is a case in point. According to Badino [8], only points between a minimum and a maximum height are considered. For this reason, the resulting representation strongly depends on these parameters. Further, the performance at larger distances is a direct result of the accuracy of the camera orientation with respect to the tilt and roll angle.

Moreover, taking into account only the first obstacle along every viewing angle can cause to miss relevant objects, such as a pedestrian entering the vehicles driveway from behind the engine hood of a car that is parked at the side of the road.

Additionally, the constraint of the first obstacles to touch the ground surface is a rather rough and strict approximation of the real-world. There are several natural examples which violate this assumption, such as overhanging load from a truck, crash barriers or gates at railroad crossings.

In order to make the representation more consistent with our actual world geometry, the initial idea for the Stixel representation is extended accordingly.

3.2 Extension of the Stixel Primitive

In [63], Felzenszwalb et al. proposed a labeling scheme that segments an arbitrary image into three continuous horizontal labels. For this purpose a *bottom*, *middle* and *top* region is computed. While the *bottom* and *top* region describe floor and background, the *middle* region describes the actual object content of the scene. The authors allow additional but limited vertical sub-partitioning of the middle region, such that the objects can be segmented further with a little more detail. However, they do not perform any further horizontal segmentation of the object layer. Thus, only one object region per column is extracted which does not suffice to model the level of detail that typical road scenarios exhibit. Anyway, the basic idea for this labeling scheme has similarities to what we intend to obtain by means of the multi-layered Stixel representation. Also, the authors rely on DP for the solving step.

Our technique is also closely related to the work presented by Gallup et al. [71]. Their suggested approach creates a variant of a Cartesian three-dimensional (volumetric) occupancy grid referred to 'n-layer heightmap'. Therefore, it requires to collect a certain amount of data from multiple views until applying the segmentation step is feasible. Thus, it is not suited to work on a single frame basis and has difficulties to consider dynamic scene content. For the actual object extraction step, they rely on DP and segment every cell of the obtained map to adjacent regions while altering between the labels *free* and *occupied* (by an object).

For our purpose, a single frame approach is described. Every column of an image is segmented into multiple vertical segments. In addition, every segment is assigned to exactly one of the three possible labels of type *ground*, *object* and *sky*. Those labels are given the following meanings:

- *ground*
  All areas of the three-dimensional world with horizontal orientation, such as floor, road or sidewalks.
- *object*
  That content of the scene that has vertical and thus upright orientation, such as obstacles, pedestrians, solid infrastructures or other road users.
- *sky*
  All areas of the image that belong to the sky or vertically oriented background at a very large distance.

Our approach is not limited to a single object segment along every column, but allows for a variable and by certain means physically plausible combination of those three classes.

That way, the segmentation result is capable to correctly describe even complex scenarios, such as staggered objects located at multiple depths or changing class occurrences between *ground*, *object* and *sky*. An example for such a scenario is depicted in Figure 1.3.

3.2.1 Mapping the Stixel World to a Labeling Problem

Given the left rectified camera image I of size $w \times h$ of a stereo image pair and the set of possible labelings $\mathbb{L}$, a multi-layered *Stixel World* corresponds to a column-wise segmentation $L \in \mathbb{L}$ of I into the classes $\mathbb{C} = \{\text{g,o,s}\}$ (*i.e. ground/road*, *object* and *sky*) of the following form:

$$\begin{aligned} L &= \{L_u \mid 0 \leq u < w\} \\ L_u &= \{s_n \mid 1 \leq n \leq N_u \leq h\} \\ s_n &= \left(v_n^b, v_n^t, c_n, f_n(v)\right), \text{ with } 0 \leq v_n^b \leq v_n^t < h \text{ , } c_n \in \mathbb{C} \end{aligned} \tag{3.1}$$

L_u denotes the labeling for an image column u, that may consist of multiple segments s_n. The total number of segments for each particular column is given by N_u. Their maximum number is implicitly limited by the height of the image, such that $0 < N_u \leq h$. In this notation the image row coordinates v_n^b (base point) and v_n^t (top point) mark the beginning and end of segment s_n.

Further, the segment is assigned to a class $c_n \in \mathbb{C}$. The term $f_n(v)$ is an arbitrary function $f : \mathbb{N} \to \mathbb{D}$ that computes the disparity (i.e. the depth) of that segment. It is only defined at row v with $v_n^b \leq v \leq v_n^t$.

All segments s_{n-1} and s_n are vertically adjacent such that for each segmentation $L_u \in L \in \mathbb{L}$ of column u the following ordering applies:

$$\begin{gathered} 0 = v_1^b \leq v_1^t < \ldots < v_{N_u}^b \leq v_{N_u}^t = h - 1 \\ \text{with } v_{n-1}^t + 1 = v_n^b,\ 1 < n \leq N_u \end{gathered} \tag{3.2}$$

Since every segmentation $L \in \mathbb{L}$ conforms to (3.2) it is implicitly guaranteed that every image point is assigned to exactly one label. Two examples for valid column segmentations are depicted in Figure 3.2.

According to [9], all Stixels share the same width w_s of typically 5 px in the image. With regard to the specifications of our camera system (which typically is 1024×440 px with about 45° field of view), this has proven a good working choice. Thus, we maintain this strategy for our purpose as well. However, depending on the particular use case or given a different sensor setup, adapting the Stixel width might prove beneficial. Nevertheless, irrespective of the particular choice, every w_s columns are labeled together.

3.2.2 The Data Model

During the labeling step all segments are assigned to be either *ground, object* or *sky*. For this purpose, all segments are assumed to approximate piecewise planar surfaces of the three-dimensional environment. Consequently, the choice for a function f_n is reduced to the set of linear functions. Given that our world geometry in man-made environments mainly consists of surfaces with either vertical or horizontal orientation, this function set can be reduced even further. Segments assigned to *object* are assumed to have a constant disparity, such that the corresponding function is given as $f_n^o(v) = d_n$, where d_n is the representative disparity within the segment s_n. Note that this approximation is only valid for small tilt angles of the stereo rig. If the camera tilt angle is known, this effect is straightforward to correct within the stereo data (e.g. by using rotation). However, with regard to the typical camera installation as used for detecting objects and freespace in traffic environments, that error is negligible.

The expected surface for every segment labeled as *ground* is modeled as the linear function $f_n^g(v) = \alpha \cdot (v_{\text{hor}} - v)$, where α is the expected ground disparity gradient and v_{hor} is the row coordinate of the horizon. Both parameters α and v_{hor} are extracted from the known camera geometry.

For class *sky* a function disparity of $f_n^s = d_n$ near 0 is assumed. This is because sky is expected at an infinite distance. Note that the definition of the function set for *object* and *sky* is overlapping, since both functions f_n^{o} and f_n^{s} are constant. Further note that objects can also be located near disparity $d_n = 0$. The actual difference lies within the model parametrization that is dealt with later in this section. The idea of using these linear functions is illustrated in Figure 3.2. It shows, how object segments have approximately constant disparity profiles. Ground segments on the other hand follow the disparity gradient of the road surface.

Just like any other sensor, camera sensors and stereo matching algorithms are prone to errors. This leads to noise, missing disparity measurements or outliers as illustrated in Figure 3.2 by means of the actual disparity measurement vectors for both exemplary marked columns. For this reason, the next section introduces this labeling task in terms of a probabilistic approach that is capable to deal with such artifacts and sensor characteristics.

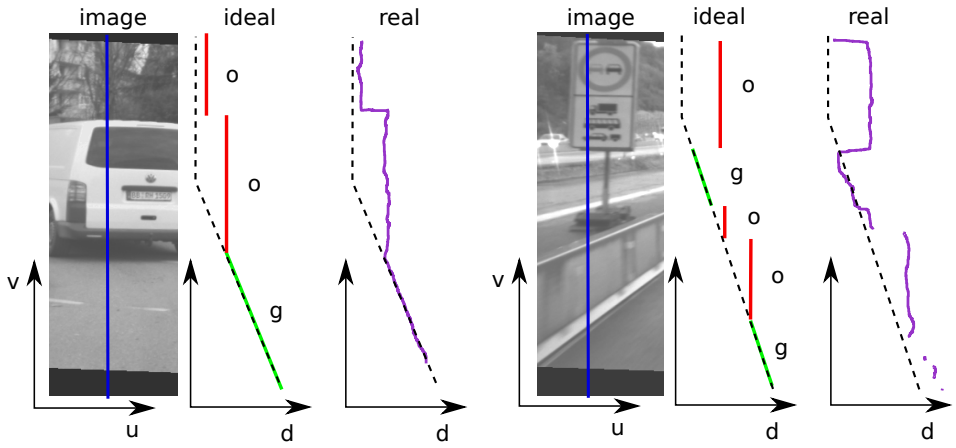


Figure 3.2: Visualization of the data model. The blue line across the image marks an exemplary column that is segmented. Red and green denote the ideal data and segmentation into *object*, *ground* and *sky*. The dashed line is the expected ground profile. The actual disparity measurement vector for the column of each particular scenario is marked purple.

3.2.3 The Stixel World as a Probabilistic Approach

The reconstruction process of the three-dimensional environment does not solely depend on the measured input data. Instead, it is also supported by a certain set of world model assumptions. In contrast to hard thresholds, priors are incorporated in a probabilistic fashion with the objective to find the most likely, most probable and thus optimal reconstruction for the three-dimensional scenario. The reconstruction process described in the following aims to model physically motivated assumptions and expectations of our three-dimensional environment. That includes the following:

- Bayesian information criterion (BIC): The number of objects captured along every column is usually small. Dispensable cuts should be avoided (cf. [25, 71, 187]).
- Gravity constraint: Flying objects are unlikely. The ground-adjacent object segment should stand on the ground surface.
- Ordering constraint: The upper of two staggered *object* segments is expected to have a greater depth than the lower one. Reconstructing otherwise (e.g. for traffic lights, signs or trees) is still possible if sufficiently supported by the input data.
- Staggering constraint: Certain constellations of different segment types are physically implausible and thus must not occur. For instance, this includes *ground* segments above *sky* or multiple vertically adjacent *sky* segments.
- Diving constraint: Objects are not expected to intersect into the ground. Thus, the base point for objects should lie above the ground surface.

3.3 Formulation as MAP Problem

Searching for the Stixel representation that matches best with the above criteria is a typical MAP estimation problem. To this end, let Z be our measurement that can consist of multiple arbitrary input cues. Let $P(L \mid Z)$ express the probability of L for being a valid labeling of Z. Now, the goal is to determine the most probable labeling $L^* \in \mathbb{L}$, such that:

$$L^* = \arg\max_{L \in \mathbb{L}} P(L \mid Z) \tag{3.3}$$

Unfortunately, any prior knowledge about that posterior probability $P(L \mid Z)$ is hard to infer. The only strategy at this point would be to check for all possible elements $L \in \mathbb{L}$, which is obviously not an option. However, applying the Bayes' theorem allows to express $P(L \mid Z)$ as:

$$P(L \mid Z) = \frac{P(Z \mid L) \cdot P(L)}{P(Z)} \tag{3.4}$$

In this notation, $P(Z)$ is a normalization factor that expresses the likelihood of the observed measurement input Z. That means, $P(Z)$ denotes the probability to receive exactly those sensor measurements (i.e. the stereo depth map D for the left input image I) as we just did. However, neither do we posses any knowledge or expectation about that at this point nor is this relevant, since the measurement input Z is not altered while computing the maximum of the posterior probability (in order to find L^*). Therefore, $P(Z)$ is neglected for the subsequent steps, which results in $P(L \mid Z) \sim P(Z \mid L) \cdot P(L)$.

From a theoretical point of view it should be mentioned that this will consequently yield the maximum likelihood, but not the actual maximum posterior probability $P(L \mid Z)$. Since we focus on processing stereo disparities and for now do not incorporate further input cues, we substitute the input measurement Z with the stereo depth map D, which yields

$$P(L \mid D) \sim P(D \mid L) \cdot P(L), \tag{3.5}$$

the product of the conditional probability of D given L and the prior probability $P(L)$ of L. $P(D \mid L)$ rates the probability to compute the depth map D given a certain labeling L. Later on, this will be the data term for the optimization. The second term $P(L)$ embodies the overall probability for each individual labeling L and is the lever to model such regularization constraints as listed above in Section 3.2.3.

3.3.1 Choosing an Optimization Space

In order to proceed with the segmentation task it is required to discuss the coordinate space, which is used to optimize for L^*. For this purpose, we decide to work in image coordinates by using the v-disparity space [120, 168, 195]. This proves advantageous for a couple of reasons. Firstly, when using stereo disparities no extra computation time is required for triangulation or projection between different coordinate spaces, which renders this choice as quite efficient. Secondly, that coordinate space has inherently finite boundaries, which is of benefit later on when working with probability densities.

Besides that, additional quantization artifacts are avoided. For instance, this is a common side effect of mapping measurements to grids or voxel spaces, such as occupancy grids or elevation maps. In addition, the noise characteristic of the depth measuring sensor is preserved and thus can be considered directly.

When utilizing other sensor types one might reconsider that choice, because changing to a different space could prove beneficial. E.g. when using a 360 degree LIDAR, such as the Velodyne HDL-64E S2 [85], relying on cylindrical coordinates might be advisable. Anyhow, this step is optional, as we will show in the results section.

3.3.2 Requirements for the Data and Labels

In order to proceed with the expression given in Equation (3.5), the conditional probability $P(D \mid L)$ and the prior probability $P(L)$ have to be transformed, such that we are able to express both explicitly. This requires to make some fundamental assumptions about the properties of the input data D, the labeling L and their semantic relation.

In the following, every single disparity $d \in D$ is stated as mutually independent from all other $\overline{d} \in D \backslash d$. Naturally, this is barely correct when using stereo matching schemes such as SGM [88], belief-propagation stereo [202, 232, 229], or approaches based on total variation [147, 236]. All of these techniques optimize for smoothness and thus exhibit correlations to neighboring disparities. However, stating the assumption of mutual independence is a necessary step in order to get hold of the vast complexity this problem exhibits. Due to this decision, the disparity input is generalized to the vertical disparity vector $D_u \in D$, with $D_u = \{d_v\}$ and $d_v = D(u, v)$, $0 \leq u < w$, $0 \leq v < h$. Hence, this procedure allows to transform $P(D \mid L)$ as follows:

$$
\begin{aligned}
P(D \mid L) &= P(D_0, \ldots, D_{w-1} \mid L) \\
&= P(D_0 \mid L) \cdot \underbrace{P(D_1 \mid L, D_0)}_{P(D_1|L)} \cdot \underbrace{P(D_2 \mid L, D_0, D_1)}_{P(D_2|L)} \\
&\quad \cdot \ldots \cdot \underbrace{P(D_{w-1} \mid L, D_0, \ldots, D_{w-2})}_{} \\
&\sim P(D_0 \mid L) \cdot P(D_1 \mid L) \cdot \ldots \cdot P(D_{w-1} \mid L) \\
&= \prod_{u=0}^{w-1} P(D_u \mid L)
\end{aligned}
\tag{3.6}
$$

According to Equation (3.6), the different columns of the image can be computed independently. Furthermore, the measured data from D_u is considered as mutually independent from all remaining labeled columns $L_{\overline{u}} \in L \backslash L_u$. Rewriting the likelihood $P(D_u \mid L)$ as $P(D_u \mid L_1, \ldots, L_w)$ allows to reduce this term to $P(D_u \mid L_u)$. As a result we obtain:

$$
P(D \mid L) = \prod_{u=0}^{w-1} P(D_u \mid L) = \prod_{u=0}^{w-1} P(D_u \mid \underbrace{L_0, \ldots, L_{w-1}}_{L_u}) \sim \prod_{u=0}^{w-1} P(D_u \mid L_u) \tag{3.7}
$$

Additionally, for the prior $P(L)$ mutual dependencies between the columns $L_u, L_{\overline{u}} \in L$ with $u \neq \overline{u}$ are not modeled. This makes it possible to transform $P(L)$ as follows:

$$\begin{aligned} P(L) &= P(L_0, \ldots, L_{w-1}) \\ &= P(L_0) \cdot \underbrace{P(L_1 \mid L_0)}_{P(L_1)} \cdot \ldots \cdot \underbrace{P(L_{w-1} \mid L_0, \ldots, L_{w-2})}_{P(L_{w-1})} \\ &\sim \prod_{u=0}^{w-1} P(L_u) \end{aligned} \tag{3.8}$$

Finally, the posterior probability $P(L \mid D)$ from Equation (3.5) can be expressed as:

$$P(L \mid D) \sim \prod_{u=0}^{w-1} P(D_u \mid L_u) \cdot P(L_u) \tag{3.9}$$

The following sections discuss how the likelihood $P(D_u \mid L_u)$ and the prior probability density function $P(L_u)$ are split up further.

3.4 Definition of the Data Terms

The conditional probability density $P(D_u \mid L_u)$ has the objective to rate the likelihood of the input data given a labeling L_u. In order to disassemble $P(D_u \mid L_u)$ further, this term is rewritten as follows:

$$\begin{aligned} P(D_u \mid L_u) &= P(d_0, v_0, \ldots, d_{h-1}, v_{h-1} \mid L_u) \\ &= P(d_0, v_0 \mid L_u) \cdot \underbrace{P(d_1, v_1 \mid L_u, d_0, v_0)}_{P(d_1, v_1 \mid L_u)} \\ &\quad \cdot \ldots \cdot \underbrace{P(d_{h-1}, v_{h-1} \mid L_u, d_0, v_0, \ldots, d_{h-2}, v_{h-2})}_{P(d_{h-1}, v_{h-1} \mid L_u)} \\ &\sim \prod_{v=0}^{h-1} P(d_v, v \mid L_u) \end{aligned} \tag{3.10}$$

To clarify the notation, d_v denotes the disparity value at row v within the disparity column vector D_u with $0 \leq u < w$ and $0 \leq v < h$. All terms $P(d_v, v \mid L_u, d_1, \ldots, d_{v-1})$ are reduced to $P(d_v, v \mid L_u)$. This is possible since all disparity measurements are assumed as mutually independent (see Section 3.3.2). Therefore, we continue with:

$$\begin{aligned} P(D_u \mid L_u) &= \prod_{v=0}^{h-1} P(d_v, v \mid L_u) \\ &= \prod_{v=0}^{h-1} P(d_v \mid s_1, \ldots, s_{N_u}, v) \cdot P(v \mid s_1, \ldots, s_{N_u}) \\ &\sim \prod_{v=0}^{h-1} P(d_v \mid s_1, \ldots, s_{N_u}, v) \end{aligned} \tag{3.11}$$

For our purpose, the term $P(v \mid s_1, \ldots, s_{N_u})$ is modeled as a uniform distribution and thus simply has a normalizing effect. Therefore, when searching for the maximum of the likelihood $P(D_u \mid L_u)$, it can be neglected.

For the next steps, another assumption about the measurement data and the segment labels has to be made. We state that a mutual dependency of d_v must only exist to that segment s_n where $v_n^b \leq v \leq v_n^t$. Given that property, $P(D_u \mid L_u)$ is split up further:

$$\begin{aligned} P(D_u \mid L_u) &= \prod_{v=0}^{h-1} P(d_v \mid s_1, \ldots, s_{N_u}, v) \\ &= \prod_{v=0}^{h-1} \begin{cases} P(d_v \mid s_n, v) & , v_n^b \leq v \leq v_n^t \\ 1/(d_{\max} - d_{\min}) & , \text{otherwise} \end{cases} \end{aligned} \tag{3.12}$$

In this notation, $d_{\min}$ and $d_{\max}$ denote the minimum and maximum disparity value. Finally, the term $P(D_u \mid L_u)$ can be rewritten as the overall product of

$$P(D_u \mid L_u) = \prod_{n=1}^{N_u} \prod_{v=v_n^b}^{v_n^t} P(d_v \mid s_n, v)\,. \tag{3.13}$$

3.4.1 Generic Sensor and Measurement Model

According to Equation (3.13) the probability density $P(d_v \mid s_n, v)$ for a disparity measurement d_v at row v within segment s_n has to be defined. This term will play an essential role for the optimization in order to estimate the overall maximum posterior probability. Referring to Thrun et al. [209], the term $P(d_v \mid s_n, v)$ is a typical forward model, because it embodies a generative description of the physical characteristics of the used sensor.

Thus, in order to continue, the sensor characteristics have to be defined and modeled, such that the data model described in Section 3.2.2 can be applied. These topics are covered within the next sections.

3.4.1.1 The Measurement Model for Stereo Sensors

In the following we proceed with the intermediate result of Equation (3.13) and the decision for the v-disparity space. Its applicability for the optimization step was motivated in Section 3.3.1. All sensors are prone to errors under particular conditions. Referring to the stereo sensors, this means that not every pixel necessarily has a valid disparity measurement and if it does, it might be an outlier.

In the following this particular sensor characteristic is considered in two steps. Firstly, we account for an outlier rate p_{out} that models the probability to encounter outliers.

Secondly, the case of not having a valid disparity measurement has to be dealt with. To this end, a mapping $\exists : \mathbb{D} \rightarrow \{0, 1\}$ and the probabilities $p_{\nexists}^c$, with $c \in \{\text{g}, \text{o}, \text{s}\}$ are defined. The mapping $\exists(d_v)$ equals '1' if the disparity d_v is valid (i.e. $d_v \neq d_{\text{inv}}$) and '0' otherwise.

The meaning of the variables $p_{\nexists}^{c}$ is explained as follows: Given that d_v is an invalid disparity measurement (i.e. $\exists(d_v) = 0$), the term $p_{\nexists}^{c}$ denotes the probability for pixel v (in row u) to be of class type c. Accordingly, for an arbitrary pixel within the segment s_n follows $p_{\nexists}^{c_n} = P\left(c_n \mid \exists(d_v) = 0\right)$.

It is important to note that the choice for a pixel to be labeled as *object, ground* or as *sky* is exclusive, since every pixel must have a unique class assignment. Further note that for the terms $p_{\nexists}^{g}$, $p_{\nexists}^{o}$ and $p_{\nexists}^{s}$ the condition $p_{\nexists}^{g} + p_{\nexists}^{o} + p_{\nexists}^{s} = 1$ has to apply.

At this point, however, we are not particularly interested in the term $P\left(c_n \mid \exists(d_v) = 0\right)$, but in $P\left(\exists(d_v) = 0 \mid c_n\right)$. Their connection is given by applying the the Bayes' theorem to $P\left(c_n \mid \exists(d_v) = 0\right)$, such that:

$$P\left(\exists(d_v) = 0 \mid c_n\right) = \frac{P\left(c_n \mid \exists(d_v) = 0\right) \cdot P\left(\exists(d_v) = 0\right)}{P\left(c_n\right)} = \frac{p_{\nexists}^{c_n} \cdot P\left(\exists(d_v) = 0\right)}{P\left(c_n\right)} \tag{3.14}$$

In this process, the terms $P(\exists(d_v) = 0)$ and $P(c_n)$ are modeled as constant. Alternatively, $P\left(c_n\right)$ and $P\left(\exists(d_v) = 0\right)$ can be statistically learned and, where required, can be adapted depending on the particular driving scenario or environmental conditions.

With that background, $P\left(d_v \mid s_n, v\right)$ is defined as:

$$P\left(d_v \mid s_n, v\right) = \begin{cases} P_{\mathrm{D}}\left(d_v \mid s_n, v\right) \cdot \left(1 - P\left(\exists(d_v) = 0 \mid c_n\right)\right) & \text{, if } \exists(d_v) = 1 \\ P\left(\exists(d_v) = 0 \mid c_n\right) & \text{otherwise} \end{cases} \tag{3.15}$$

The term $P_{\mathrm{D}}\left(d_v \mid s_n, v\right)$ (also referred to as *"data term"*) denotes the probability for a particular disparity measurement d_v given the row v and the segment s_n. Pursuant to the properties of our sensor model, it is defined as a mixture model that consists of a uniform distribution with the objective to model the chance to encounter outliers and a Gaussian distribution in order to rate the affinity of d_v to s_n. The idea of using such a mixture model is motivated in Figure 3.3.

The definition of the stereo sensor model $P_{\mathrm{D}}\left(d_v \mid s_n, v\right)$ is given by:

$$P_{\mathrm{D}}\left(d_v \mid s_n, v\right) = \frac{p_{\mathrm{out}}}{d_{\max} - d_{\min}} + A_{\mathrm{norm}} \cdot e^{-\frac{1}{2}\left(\frac{d_v - f_n(v)}{\sigma_{c_n}(f_n, v)}\right)^2} \tag{3.16}$$

The expected outlier rate is given by p_{out}. The standard deviation parameter $\sigma_c(f_n, v)$, $c \in \mathbb{C}$ of the Gaussian distribution incorporates the noise model for the disparity measurement with respect to the class type c_n. Thus, a model-based normalization of the individual probability densities is achieved. The term A_{norm} re-normalizes the Gaussian function, such that the integral of $P_D(d_v \mid s_n, v)$ for the actual disparity domain equals one. It is defined as:

$$A_{\mathrm{norm}} = \frac{1 - p_{\mathrm{out}}}{A_{\mathrm{range}}} \cdot \frac{1}{\sigma_{c_n}\left(f_n, v\right) \cdot \sqrt{2\pi}} \tag{3.17}$$

Again, A_{norm} contains a further parameter A_{range} that is required for interval normalization. This is because the domain for valid disparity values is limited to $d_v \in [d_{\min}, d_{\max}]$, whereas the Gaussian distribution is originally defined for an infinite domain. The two parameters $\sigma_{c_n}\left(f_n, v\right)$ and A_{range} are determined within the next two subsections.

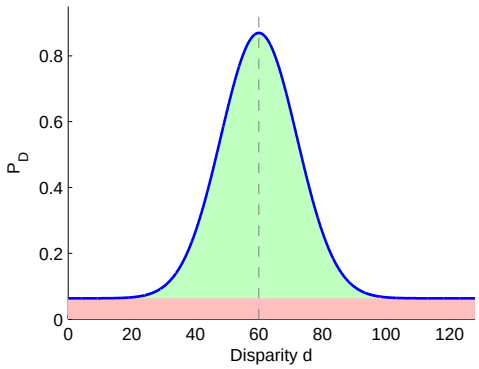


Figure 3.3: Illustration of a mixture model that consists of a uniform and a Gaussian distribution. The uniform distribution (red area) allocates for the chance of outliers, the Gaussian distribution (green area) rates actual disparity measurements with respect to the estimated mean (dashed line).

3.4.1.2 Model-Based Normalization by Error Propagation

The class-dependent standard deviations $\sigma_{c_n}(f_n, v)$ with σ_o, σ_g or σ_s for *object*, *ground* and *sky* are determined dynamically. They all stand in relation to the expected sensor measurement noise, the camera geometry and tilt angle accuracy. Thus, they directly depend on the used sensor type and its particular parametrization. For our purpose, they are derived from the stereo sensor model by error propagation.

These noise parameters take into account by what degree single disparity measurements are allowed to mismatch with the assumed segment function f_n. For this purpose the following parameters regarding the stereo sensor and world model are considered:

- the disparity uncertainty σ_d of the stereo camera system
- the camera installation height h_{cam}, the camera tilt angle α_{cam} as well as their the corresponding uncertainties $\sigma_{h_{\text{cam}}}$ and $\sigma_{\alpha_{\text{cam}}}$
- the range of depth Δ_Z into which objects are allowed to extend, hence violating the constant disparity assumption for segments labeled as *object*

The uncertainties $\sigma_o\left(f_n, v\right)$ and $\sigma_g\left(f_n, v\right)$ for *object* and *ground* are constructed as follows:

$$\sigma_o\left(f_n, v\right)^2 = \sigma_d^2 + \sigma_Z\left(f_n(v)\right)^2 \tag{3.18}$$

$$\sigma_g\left(f_n, v\right)^2 = \sigma_d^2 + \sigma_r\left(v\right)^2 \tag{3.19}$$

Apparently, both equations contain two additional terms: The disparity uncertainty $\sigma_Z\left(f_n(v)\right)$ resulting from Δ_Z and the uncertainty $\sigma_r\left(v\right)$ for the actual road model. Due to perspective effects they are adaptive, which is why $\sigma_Z\left(f_n(v)\right)$ depends on the value of the disparity function $f_n(v)$ for a possible segment s_n. On the other hand, the road model uncertainty $\sigma_r\left(v\right)$ directly depends on the row position v.

In detail, the noise term $\sigma_Z\left(f_n(v)\right)$ is derived from the projection equation using error propagation. This step assumes the pinhole camera geometry as well as a standard stereo camera setup, such that:

$$\begin{aligned} Z_v(d_v) &= \frac{f_u \cdot b}{d_v} \\ \frac{\partial Z}{\partial d} &= -\frac{f_u \cdot b}{d^2} \end{aligned} \tag{3.20}$$

This leads to the expression

$$\sigma_Z\left(f_n(v)\right) = \frac{f_n(v)^2}{f_u \cdot b} \cdot \Delta_Z, \tag{3.21}$$

where b is the base line and f_u, f_v correspond to the horizontal and vertical focal lengths of the left rectified camera. Further, the road model uncertainty $\sigma_r(v)$ is computed from:

$$Z_v = \frac{h_{\text{cam}}}{\frac{v_{\text{hor}}-v}{f_v} + \alpha_{\text{cam}}} = \frac{f_u \cdot b}{d_v} \tag{3.22}$$

The middle term is approximate and valid for small values of α_{cam}. The coordinate Z_v represents the expected distance to the ground surface at v under the assumption of a flat surface. v_{hor} is the image row coordinate of the horizon and extracted from the known camera orientation. Naturally, this term is only valid for $v - v_{\text{hor}} < f_v \cdot \alpha_{\text{cam}}$. However, as discussed in Section 3.4.1.1, road occurrence is not expected above the horizon anyway. Following Equation 3.22 results in:

$$d_v = \frac{f_u \cdot b}{h_{\text{cam}}} \cdot \left(\frac{v_{\text{hor}} - v}{f_v} + \alpha_{\text{cam}}\right), \text{ and thus} \tag{3.23}$$

$$\begin{aligned} \sigma_r(v)^2 &= \left(\frac{\partial d_v}{\partial h_{\text{cam}}}\right)^2 \cdot \sigma_{h_{\text{cam}}}^2 + \left(\frac{\partial d_v}{\partial \alpha_{\text{cam}}}\right)^2 \cdot \sigma_{\alpha_{\text{cam}}}^2 \\ &= \left(\frac{f_u \cdot b}{h_{\text{cam}}}\right)^2 \cdot \left[\frac{1}{h_{\text{cam}}^2} \cdot \left(\frac{v_{\text{hor}} - v}{f_v} + \alpha_{\text{cam}}\right)^2 \cdot \sigma_{h_{\text{cam}}}^2 + \sigma_{\alpha_{\text{cam}}}^2\right] \end{aligned}$$

At last, the noise parameter σ_s for *sky* labeled segments requires a few extra words. According to a statement made in Section 3.2.2, *sky* segments are expected to have an approximating function with $f_n^s(v) \approx 0$. This is self-evident, since *sky* segments are modeled as vertically oriented surfaces at an infinite distance. However, stereo matching algorithms are commonly subject to encounter severe problems in *sky* regions, especially when those parts of the image exhibit insufficient texture information. This scenario is not unlikely and for instance occurs on cloudless situations or for homogeneously misted skies. This particularity has to be dealt with separately. Thus, a dedicated disparity noise σ_s and outlier rate p_{out}^s are defined for *sky* segments. Compared to the standard disparity noise σ_d the expected noise term σ_s for *sky* is rather small. In contrast, the outlier rate is expected as significantly higher.

3.4.1.3 Gaussian Interval Normalization

Naturally, the domain for disparity measurements $d \in D$ is limited, such that $d_{\min} \leq d \leq d_{\max}$. For our stereo camera setup the domain for valid disparities equals to the interval of $d \in [d_{\min} = 0\,\text{px}, d_{\max} = 128\,\text{px}]$. Therefore, when relying on a Gaussian distribution to model sensor characteristics within the disparity space or comparable limited domains, the resulting probability distribution has to be re-normalized. According to Equation 3.16, this results in a scalar correction of the distribution with the factor A_{range}. To this end, A_{range} is computed by:

$$A_{\text{range}} = \int_{d_{\min}}^{d_{\max}} \frac{1}{\sigma_{c_n}(d_v, v) \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{d - f_n(v)}{\sigma_{c_n}(f_n, v)}\right)^2} \,\mathrm{d}d \tag{3.24}$$

That integral can be computed directly by using a scaled and shifted variant of the error function for a standard distributed Gaussian function with mean 0. For non-standard Gaussian distributions this results in:

$$A_{\text{range}} = \frac{1}{2} \cdot \left[\text{erf}\left(\frac{\mathrm{d_{max}} - \mathrm{f_n}}{\sigma_{\mathrm{c_n}}(\mathrm{f_n}, \mathrm{v}) \cdot \sqrt{2}}\right) - \text{erf}\left(\frac{\mathrm{d_{min}} - \mathrm{f_n}}{\sigma_{\mathrm{c_n}}(\mathrm{f_n}, \mathrm{v}) \cdot \sqrt{2}}\right)\right] \tag{3.25}$$

3.5 Definition of the A Priori Terms

The a priori term $P(L_u)$ from Equation (3.9) incorporates the world model expectation with respect to the resulting three-dimensional reconstruction of the scene. In a certain way this is in analogy to the smoothing priors of modern stereo matching algorithms, such as mentioned in Section 3.3.2, or certain optical flow methods such as the TV-L1 optical flow estimator [97, 161, 236].

In contrast to the data term likelihood $P(Z_u \mid L_u)$, the a priori term $P(L_u)$ does not contain any dependencies to the input data. Instead, it expresses mutual and semantic aspects between the segments of a column, such as those listed in Section 3.2.3. In the following Sections $P(L_u)$ is split up and defined accordingly.

3.5.1 Deduction of the A Priori Terms

The likelihood term $P(L_u)$ rates the probability to encounter a certain constellation of segments s_n with $L_u = \{s_1, \ldots, s_{N_u}\}$. For that purpose, $P(L_u)$ is transformed as follows:

$$\begin{aligned} P(L_u) &= P(s_1, \ldots, s_{N_u}) \\ &= P(s_1) \cdot P(s_2 \mid s_1) \cdot \ldots \cdot P(s_{N_u} \mid s_1, \ldots, s_{N_u - 1}) \end{aligned} \tag{3.26}$$

Proceeding requires to discuss the particular relations in which the segments from L_u participate. In order to keep the modeling complexity tractable, pairwise mutual dependencies between all adjacent segments s_{n-1} and s_n are modeled. Hence, a segment s_n only relates to its adjacent neighboring segments s_{n-1} and s_{n+1}, and does not directly

depend on any other segment within the column labeling L_u. This property allows to simplify the upper terms as follows:

$$\begin{aligned} P(L_u) &= P(s_1) \cdot P(s_2 \mid s_1) \cdot \underbrace{P(s_3 \mid s_1, s_2)}_{P(s_3|s_2)} \cdot \ldots \cdot \underbrace{P(s_{N_u} \mid s_1, \ldots, s_{N_u-1})}_{P(s_{N_u}|s_{N_u-1})} \\ &\sim P(s_1) \cdot \prod_{n=2}^{N_u} P(s_n \mid s_{n-1}) \end{aligned} \tag{3.27}$$

According to this result, $P(L_u)$ breaks into two terms: $P(s_1)$ and the conditional terms $P(s_n \mid s_{n-1})$. The former one states the probability of a particular occurrence of the first segment $s_1 = \{v_1^b, v_1^t, c_1, f_1(v)\}$. The latter part incorporates semantic aspects between adjacent segments up to the last segment s_{N_u}. This term includes such aspects as ordering or gravity regularization.

The unary term $P(s_1)$ is defined by:

$$\begin{aligned} P(s_1) &= P\left(v_1^b, v_1^t, c_1, f_1(v)\right) \\ &= P\left(v_1^b, v_1^t\right) \cdot P\left(c_1, f_1^c(v) \mid v_1^b, v_1^t\right) \\ &= P\left(v_1^b\right) \cdot \underbrace{P\left(v_1^t \mid v_1^b\right)}_{P(v_1^t)} \cdot \underbrace{P\left(c_1 \mid v_1^b, v_1^t\right)}_{P(c_1|v_1^t)} \cdot \underbrace{P\left(f_1(v) \mid v_1^b, v_1^t, c_1\right)}_{P(f_1(v)|c_1)} \\ &\sim P\left(v_1^b\right) \cdot P\left(v_1^t\right) \cdot P\left(c_1 \mid v_1^t\right) \cdot P\left(f_1(v) \mid c_1\right) \end{aligned} \tag{3.28}$$

Note that the last step in order to yield the final result of Equation (3.28) utilized a couple of simplifications that should be explained in a few words. For reducing $P(v_1^t|v_1^b)$ to $P(v_1^t)$, $P\left(c_1 \mid v_1^b, v_1^t\right)$ to $P\left(c_1 \mid v_1^t\right)$ and $P\left(f_1(v) \mid v_1^b, v_1^t, c_1\right)$ to $P\left(f_1(v) \mid c_1\right)$, a few inherent coherences of the optimization problem have been exploited. According to the definition of the labeling (see Equation (3.2) in Section 3.2.1) and due to the fact that s_1 is the first segment, the base point v_1^b has to equal 0. Therefore, the conditional variable v_1^b in $P(v_1^t|v_1^b)$ is ignored. Secondly, $P\left(c_1 \mid v_1^b, v_1^t\right)$ is simplified to $P\left(c_1 \mid v_1^t\right)$ for the same reason. The third term is reduced from $P\left(f_1(v) \mid v_1^b, v_1^t, c_1\right)$ to $P\left(f_1(v) \mid c_1\right)$ due to semantic aspects. The mutual dependency of function $f_1(v)$ to c_1 is obvious, since the type of that function (constant disparity vs. slanted surface) is directly determined by the class type c_1. However, such a connection between $f_1(v)$ and v_1^b or v_1^b does not hold. Therefore, these two conditional parameters are neglected.

Next, the conditional probability $P(s_n \mid s_{n-1})$ from Equation (3.27) is discussed. In this context, s_n represents the upper and s_{n-1} the lower and thus previous segment. According to the definition of s_n and s_{n-1} this term can be rewritten as:

$$P(s_n \mid s_{n-1}) = P\left(v_n^b, v_n^t, c_n, f_n(v) \mid v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right) \tag{3.29}$$

Even though this may appear complex, the probability density allows to be broken

down to a few factors. Hence, the term $P(s_n \mid s_{n-1})$ factorizes as follows:

$$
\begin{aligned}
P(s_n \mid s_{n-1}) &= P\left(v_n^b, v_n^t, c_n, f_n(v) \mid v_{n-1}^b, v_{n-1}^t, c_{n-1}, d_{n-1}\right) \\
&= \underbrace{P\left(v_n^b \mid v_{n-1}^t, v_{n-1}^b, c_{n-1}, f_{n-1}(v)\right)}_{P\left(v_n^b \mid v_{n-1}^t\right) \rightarrow \text{(I)}} \\
&\quad \cdot P\left(v_n^t, c_n, d_n \mid v_n^b, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right) \\
&= P\left(v_n^b \mid v_{n-1}^t\right) \cdot \underbrace{P\left(v_n^t \mid v_n^b, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)}_{P\left(v_n^t \mid v_n^b\right) \rightarrow \text{(II)}} \\
&\quad \cdot P\left(c_n, f_n(v) \mid v_n^b, v_n^t, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right) \\
&= P\left(v_n^b \mid v_{n-1}^t\right) \cdot P\left(v_n^t \mid v_{n-1}^t\right) \\
&\quad \cdot \underbrace{P\left(c_n \mid v_n^b, v_n^t, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)}_{P\left(c_n \mid v_{n-1}^t, c_{n-1}\right) \rightarrow \text{(III)}} \\
&\quad \cdot \underbrace{P\left(f_n(v) \mid v_n^b, v_n^t, c_n, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)}_{P\left(f_n(v) \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right) \rightarrow \text{(IV)}}
\end{aligned}
\tag{3.30}
$$

For this transformation inherent coherences of the optimization problem have been exploited as well. At first, step (I) reduced the term $P\left(v_n^b \mid v_{n-1}^t, v_{n-1}^b, c_{n-1}, f_{n-1}(v)\right)$ to $P\left(v_n^b \mid v_{n-1}^t\right)$. This is justified, since neither the previous base point v_{n-1}^b, nor the class type c_{n-1} and function $f_{n-1}(v)$ relate to the base point of segment s_n in any way. For the remaining variables the ordering condition $v_n^b = v_{n-1}^t + 1$ must apply. Step (II) reduces $P\left(v_n^t \mid v_n^b, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)$ to $P\left(v_n^t \mid v_n^b\right)$ for the same reasons.

Step (III) from $P\left(c_n \mid v_n^b, v_n^t, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)$ to $P\left(c_n \mid v_{n-1}^t, c_{n-1}\right)$ drops all conditional variables that do not influence the likelihood for the class labeling c_n of segment s_n.

Respectively, the same procedure is applied in step (IV) for the resulting function $f_n(v)$, that is taken into account by the term $P\left(f_n(v) \mid v_n^b, v_n^t, c_n, v_{n-1}^b, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)$. Further meaning and semantics of these expressions as well as their definitions are discussed in the next section.

Finally, the conditional probability $P(s_n \mid s_{n-1})$ is written as:

$$
\begin{aligned}
P(s_n \mid s_{n-1}) \sim\, & P\left(v_n^b \mid v_{n-1}^t\right) \cdot P\left(v_n^t \mid v_n^b\right) \\
& \cdot P\left(c_n \mid v_{n-1}^t, c_{n-1}\right) \cdot P\left(f_n(v) \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)
\end{aligned}
\tag{3.31}
$$

3.5.2 Definition of the A Priori Terms

The previous section described how to disassemble the a priori probability $P(L_u)$ in order to yield smaller and more compact terms. Doing so led to the following two expressions:

$$\begin{aligned} P(s_1) &= P\left(v_1^b\right) \cdot P\left(v_1^t\right) \cdot P\left(c_1 \mid v_1^t\right) \cdot P\left(f_1(v) \mid c_1\right) \\ P(s_n \mid s_{n-1}) &= P\left(v_n^b \mid v_{n-1}^t\right) \cdot P\left(v_n^t \mid v_n^b\right) \cdot P\left(c_n \mid v_{n-1}^t, c_{n-1}\right) \\ &\quad \cdot P\left(f_n(v) \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right) \end{aligned} \tag{3.32}$$

To begin with, we determine the resulting terms from $P(s_1)$. Hereby $P\left(v_1^b\right)$ and $P\left(v_1^t\right)$ are of rather trivial nature: According to Equation (3.2) v_1^b must equal 0, and v_1^t is only limited by the height of the image, such that $0 = v_1^b \leq v_1^t < h$. Hence, these terms are modeled by means of the Dirac delta function and a uniform distribution:

$$\begin{aligned} P(v_1^b) &= \begin{cases} 1 & \text{, if } v_1^b = 0 \\ 0 & \text{, otherwise} \end{cases} \\ P(v_1^t) &= h^{-1} \end{aligned} \tag{3.33}$$

Further, $P\left(c_1 \mid v_1^t\right)$ denotes the probability of a certain class c_1 given the top point v_1^t of the first segment s_1. Naturally, since $v_1^b = 0$ and for our purpose the horizon v_{hor} is located at some point within the image ($0 < v_{\text{hor}} < h$) the first segment can not be of class type *sky*. On the other hand, *ground* occurrence is not allowed above the horizon. Accordingly, the term $P\left(c_1 \mid v_1^t\right)$ is defined as:

$$P\left(c_1 \mid v_1^t\right) = \begin{cases} 1 & \text{, if } v_1^t > v_{\text{hor}},\ c_1 = \text{o} \\ 0 & \text{, if } v_1^t > v_{\text{hor}},\ c_1 = \text{g} \\ 0.5 & \text{, if } v_1^t \leq v_{\text{hor}},\ c_1 \in \{\text{g}, \text{o}\} \\ 0 & \text{otherwise} \end{cases} \tag{3.34}$$

The last term $P\left(f_1(v) \mid c_1\right)$ for $P(s_1)$ puts the class type c_1 and the segment approximating function f_1 into context. It is defined as:

$$P\left(f_1(v) \mid c_1\right) = \begin{cases} {}^1\!/_{(d_{\max} - d_{\min})} & \text{, if } c_1 = \text{o} \\ 1 & \text{, if } c_1 = \text{g}, f_1(v) = \alpha \cdot (v_{\text{hor}} - v) \\ 0 & \text{otherwise} \end{cases} \tag{3.35}$$

So far, this concludes the definition of $P(s_1)$. Note that for the last two terms which used the segment type c_1, the label *sky* was not considered explicitly. This is reasoned by the fact, that the first segment is not allowed to be of label *sky*, since the bottom point v_1^b is assumed to be located below the horizon v_{hor}.

In the following the conditional density term $P(s_n \mid s_{n-1})$ is defined. According to Equation (3.32) that term is a product consisting of four coefficients. The first one $P\left(v_n^b \mid v_{n-1}^t\right)$ has the objective to rate v_n^b and thus is similar to $P\left(v_1^b\right)$. The main

difference is that v_n^b is not fixed to row coordinate 0, but is bounded from below by the top point coordinate v_{n-1}^t of the previous segment. That is, because for v_n^b and v_{n-1}^t the ordering condition $v_n^b = v_{n-1}^t + 1$ must apply to suppress invalid segmentation results. Hence, that term is defined as:

$$P\left(v_n^b \mid v_{n-1}^t\right) = \begin{cases} 1 & \text{, if } v_n^b = v_{n-1}^t + 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.36}$$

The second term $P\left(v_n^t \mid v_n^b\right)$ denotes the probability to observe a top point v_n^t of the segment s_n given the base point v_n^b of the same segment. Again v_n^t is limited, this time by the ordering constraint $v_n^b \leq v_n^t < h$. Similar to $P(v_1^t)$ the top point v_n^t is modeled as equally distributed. Thus, the term $P\left(v_n^t \mid v_n^b\right)$ is defined as:

$$P\left(v_n^t \mid v_n^b\right) = \begin{cases} 1/(h - v_n^b) & \text{, if } v_n^t \geq v_n^b \\ 0 & \text{otherwise} \end{cases} \tag{3.37}$$

Thirdly, $P\left(c_n \mid v_{n-1}^t, c_{n-1}\right)$ incorporates transition probabilities between different object types, e.g. in order to denote that it is more likely to observe a transition from object to object instead of object back to ground. Additionally, the term denotes in which row of the image which types of classes are physically possible and thus plays a similar role as $P\left(c_1 | v_1^t\right)$ did for the first segment s_1. Since it is a little bit more complex, the third term $P(c_n \mid v_{n-1}^t, c_{n-1})$ is defined by means of Table 3.1:

c_n	c_{n-1}	Condition	$P(c_n \mid v_{n-1}^t, c_{n-1})$	c_n	c_{n-1}	Condition	$P(c_n \mid v_{n-1}^t, c_{n-1})$
o	o	$v_{n-1}^t < v_{\text{hor}}$	0.7	o	o	$v_{n-1}^t \geq v_{\text{hor}}$	0.5
g	o	$v_{n-1}^t < v_{\text{hor}}$	0.3	g	o	$v_{n-1}^t \geq v_{\text{hor}}$	0
s	o	$v_{n-1}^t < v_{\text{hor}}$	0	s	o	$v_{n-1}^t \geq v_{\text{hor}}$	0.5
o	g	$v_{n-1}^t < v_{\text{hor}}$	0.7	o	g	$v_{n-1}^t \geq v_{\text{hor}}$	
g	g	$v_{n-1}^t < v_{\text{hor}}$	0.3	g	g	$v_{n-1}^t \geq v_{\text{hor}}$	not defined
s	g	$v_{n-1}^t < v_{\text{hor}}$	0	s	g	$v_{n-1}^t \geq v_{\text{hor}}$	
o	s	$v_{n-1}^t < v_{\text{hor}}$		o	s	$v_{n-1}^t \geq v_{\text{hor}}$	1
g	s	$v_{n-1}^t < v_{\text{hor}}$	not defined	g	s	$v_{n-1}^t \geq v_{\text{hor}}$	0
s	s	$v_{n-1}^t < v_{\text{hor}}$		s	s	$v_{n-1}^t \geq v_{\text{hor}}$	0

Table 3.1: Transition probabilities for the term $P(c_n | v_{n-1}^t, c_{n-1})$ for a label type c_n given the preceding type c_{n-1} of s_{n-1} and its corresponding end row coordinate v_{n-1}^t. This term also denotes which kind of transitions are physically allowed and which are not. For instance, the choice of assigning a probability of 0.7 for the transition from object to object was made to express that it is rather likely to have two cascaded *objects* instead of a transition back to *ground*.

The fourth and last term $P\left(f_n(v) \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)$ rates the possibility for a certain function f_n and thus completes Equation (3.32). It allows to model semantic and mutual relation between neighboring segments s_n and s_{n-1}. This includes such world

modeling aspects as mentioned in Section 3.2.3. Which of them applies in particular depends on the actual parameter configuration (i.e. c_n, v_{n-1}^t, c_{n-1} and $f_{n-1}(v)$). Just like $P\left(c_n \mid v_{n-1}^t, c_{n-1}\right)$, the term $P\left(f_n(v) \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1}(v)\right)$ is given by means of a table. For reasons of clarity this table is split in three parts with respect to the class type c_n. At first, the configuration for $c_n = \mathrm{o}$ is given in Table 3.2.

c_n	c_{n-1}	Condition	$P(f_n \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1})$
o	o	$d_n > d_{n-1} + \Delta_d(d_{n-1}, \Delta_Z)$	$\frac{p_{\mathrm{ord}}}{d_{n-1}-\Delta_d}$
o	o	$d_n \leq d_{n-1} - \Delta_d(d_{n-1}, \Delta_Z)$	$\frac{1-p_{\mathrm{ord}}}{d_{\max}-d_{n-1}-\Delta_d}$
o	o	$\lvert d_n - d_{n-1}\rvert < 2 \cdot \Delta_d(d_{n-1}, \Delta_Z)$	0
o	s	$d_n > \epsilon$	$\frac{1/(d_{\max}-d_{\min})}{1-\epsilon/(d_{\max}-d_{\min})}$
o	s	$d_n \leq \epsilon$	0
o	g	$d_n > f_{n-1}(v_{n-1}^t) + \epsilon$	$\frac{p_{\mathrm{grav}}}{d_{\max}-f_{n-1}(v_{n-1}^t)-\epsilon}$
o	g	$d_n < f_{n-1}(v_{n-1}^t) - \epsilon$	$\frac{p_{\mathrm{blg}}}{f_{n-1}(v_{n-1}^t)-\epsilon-d_{\min}}$
o	g	$\lvert d_n - f_{n-1}(v_{n-1}^t)\rvert < 2 \cdot \epsilon$	$\frac{1-p_{\mathrm{grav}}-p_{\mathrm{blg}}}{2\cdot\epsilon}$

Table 3.2: Table for $P(f_n(v) \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1}(v))$ modeling the probability for a function $f_n(v)$ with $c_n = \mathrm{o}$ and a remaining configuration, c_{n-1}, d_{n-1} and v_m^t. The variable d_n is the representative disparity for segment s_n. The first three lines denote the Bayesian information criterion (BIC) (cf. [25, 71, 187]) and the ordering constraint. The two center lines denote that an object has to come forward from *sky*, in order to give reason to be segmented. The last three lines (transitions from *ground* to *object*) denote the gravity constraint. Additionally they render objects with a base point below *ground* as unlikely, which corresponds to the diving constraint.

In this context, p_{ord} corresponds to the ordering regularization and models the probability of two staggered objects to violate the ordering assumption, such that s_n has a larger disparity and thus is closer than s_{n-1}. The second variable p_{grav} models the probability of ground adjacent objects to hover and thus not to touch the ground surface. The segments s_n and s_{n-1} are not allowed to coexist within a certain range $\pm\Delta_Z$. That range is mapped to disparities by $\Delta_d(\mu, \Delta_Z)$. The third and last variable p_{blg} denotes the probability for objects to have a base point below the ground surface. In this notation $f_{n-1}(v_{n-1}^t)$ with $c_{n-1} = \mathrm{g}$ is the end disparity of the preceding ground surface. The parameter ϵ denotes the range in which violations of the gravity and ordering assumption are tolerated. For our purpose, good results have been achieved by choosing $\epsilon = 1.5 \cdot \sigma_d$.

Secondly, Table 3.3 rates the likelihood for the function f_n of a segment s_n that is assigned to *ground* ($c_n = \mathrm{g}$). That scenario is not regularized by any a priori assumption except the condition that road occurrence above *sky* is physically not possible and thus is not allowed for a valid labeling result.

In order to finalize the definition of $P(f_n(v) \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1}(v))$, the fourth and last Table 3.4 rates functions f_n with transitions to the label type $c_n = \mathrm{s}$.

c_n	c_{n-1}	Condition	$P(f_n \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1})$
g	o∨g	$f_n(v) = \alpha \cdot (v_{\text{hor}} - v)$	1
g	s	not allowed	0
g	o∨g	otherwise	0

Table 3.3: This table denotes the probability to have a function f_n that describes a segment s_n labeled as *ground*. Hereby, all parameters of the previous segment s_{n-1} are known. Apparently, this scenario is rather simple and thus has only a few restrictions. The only condition that has to apply is that f_n matches with the ground expectation.

c_n	c_{n-1}	Condition	$P(f_n \mid c_n, v_{n-1}^t, c_{n-1}, f_{n-1})$
s	o	$d_{n-1} < \epsilon$	0
s	o	$d_n = 0$	1
s	g	$f_{n-1}(v_{n-1}^t) = d_n = 0$	1
s	s	not allowed	0
s	o∨g	otherwise	0

Table 3.4: This table denotes the conditions that have to apply for a function f_n describing a *sky* labeled segment s_n. Given that a segment is labeled as *sky*, the approximating function $f_n(d) = d_n$ must equal 0. In addition, certain configurations are not allowed according to our world modeling.

For our results we choose $p_{\text{ord}} = p_{\text{grav}} = 0.1$ and $p_{\text{blg}} = 0.001$, which renders especially the last configuration as quite unlikely. ϵ is chosen as $\epsilon = 3 \cdot \sigma_d$ disparities.

3.6 Solving for L^* by Means of Dynamic Programming

Dynamic Programming (DP) [22] is a solving scheme that has been successfully applied to a vast number of optimization problems. It has the major benefit of yielding the global optimum of a problem non-iteratively in polynomial time. Hence, using DP does not hold the risk of getting stuck in a local minimum.

During the dynamic programming step a cost minimization is performed. These costs are deduced from the probability density functions as defined for the data term $P(D_u|L_u)$ and the smoothness term $P(L_u)$ earlier in this chapter. For this purpose, at first, the duality of probabilities and costs is explained briefly. Secondly, the optimization problem has to meet two essential requirements in order to be able to apply DP.

The actual run-time complexity as well as the memory requirements of the resulting algorithm directly depend on the particular optimization problem. Therefore, thirdly, auxiliary implementation details are outlined in order to compute L^* efficiently.

3.6.1 Numerical Limits and Computability

Section 3.3 focused on showing how the segmentation task can be expressed as a maximum a posteriori estimation problem. This led to the definition of the data terms (see Section 3.4.1) as well as the a priori and model terms (see Section 3.5.2). Accordingly, those interact within a large product of individual probability density distributions.

Such expressions are hard to compute on standard hardware for a variety of reasons. The most significant problem is of numerical nature. With the potential for underflow, standard computer hardware is incapable to represent such small magnitudes as they result from the product sums of the individual probability densities. Another aspect arises from the demand for efficiency in order to obtain a system that is capable to reach real-time performance levels.

Therefore, the following subsections will outline possible options to ensure both: Computability and computational efficiency for the discussed segmentation task.

3.6.1.1 The Energy-Probability Relation

This section focuses on the numerical issues as a result of the large products of the probability densities. As mentioned before, by relying on DP a cost minimization is performed. For the actual solving step it is common practice that this relation between costs and probability densities is made by using the negative natural logarithm of the actual likelihoods [71, 79, 110, 117]. Note that these costs are also often referred to as *log-likelihood* or *energy*.

Since the natural logarithm is a continuous strictly increasing function over the range of the likelihood, the values which maximize this likelihood will also minimize the resulting costs. Consequently, maximizing the posterior probability density $P(L \mid D)$ is equivalent to minimizing its log-likelihood. Following this procedure is very useful to avoid numerical problems resulting from the small magnitudes of the individual probabilities. In addition, minimizing the logarithm allows for a less complex algebra. Note that the log-likelihood is closely related to the information entropy [77] or the Fisher information [40, 140].

For the posterior probability, relying on the negative logarithm has the positive effect that the product of individual probability densities converts into a sum of their corresponding log-likelihoods. Thus, for the product of the posterior probabilities $P(L \mid D) \sim \prod_{u=0}^{w-1} P(D_u \mid L_u) \cdot P(L_u)$ from Equation (3.9) in Section 3.3.2 follows:

$$\begin{aligned} -\log\left(P(L \mid D)\right) &\sim -\log\left(\prod_{u=0}^{w-1} P(D_u \mid L_u) \cdot P(L_u)\right) \\ &= \sum_{u=0}^{w-1} -\log\left(P(D_u \mid L_u) \cdot P(L_u)\right) \\ &= \sum_{u=0}^{w-1} -\log\left(P(D_u \mid L_u)\right) + \sum_{u=0}^{w-1} -\log\left(P(L_u)\right) \end{aligned} \tag{3.38}$$

In analogy, this proceeding is applied to the definition of the conditional probability density $P(D_u \mid L_u)$ (see Equation (3.13) in Section 3.4) and the a priori term $P(L_u)$

(see Equation (3.27) in Section 3.5). Hence, when proceeding to the lowest level, the log-likelihood for the data term $P_{\mathrm{D}}(d_v \mid s_n, v)$ from Equation (3.16) has to be determined, which is the central issue of the following subsection.

3.6.1.2 Approximations for Computability

The computation of the negative log-likelihood for the data term $P_{\mathrm{D}}(d_v \mid s_n, v)$ comes with a few difficulties. This is reasoned by the nature of this distribution as a mixture model. Technically, $P_{\mathrm{D}}(d_v \mid s_n, v)$ consists of a sum of a uniform distribution in order to allow for outliers, and a Gaussian distribution in order to rate the affinity of the disparity measurement d_v to segment s_n. Applying the negative logarithm in order to obtain the actual costs results in:

$$
\begin{aligned}
&-\log\left(P_{\mathrm{D}}\left(d_v \mid s_n, v\right)\right) \qquad (3.39)\\
&= -\log\Bigg(\underbrace{\frac{p_{\mathrm{out}}}{d_{\max} - d_{\min}}}_{P_{\mathrm{uniform}}} + \underbrace{\frac{1 - p_{\mathrm{out}}}{A_{\mathrm{gauss}}} \cdot \frac{1}{\sigma_{c_n}\left(f_n, v\right) \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{d_v - f_n(v)}{\sigma_{c_n}(f_n, v)}\right)^2}}_{P_{\mathrm{Gaussian}}}\Bigg)\\
&= -\log\left(P_{\mathrm{uniform}} + P_{\mathrm{Gaussian}}\right)
\end{aligned}
$$

At this point the expression above can not be simplified further. That is unsatisfying, because that term is very costly to compute. Thus, in order to allow for a more efficient computation, an approximation for the expression from Equation (3.39) is used. Hence, the resulting energy $E_{P_{\mathrm{D}}}$ is defined as:

$$
E_{P_{\mathrm{D}}} = \min\left(-\log\left(P_{\mathrm{uniform}}\right), -\log\left(P_{\mathrm{Gaussian}}\right)\right) \qquad (3.40)
$$

The differences to the non-approximated variant are visualized in Figure 3.4. For this purpose, the exemplary distribution previously depicted in Figure 3.3 is used. The resulting effect is similar to cost limitation or cost saturation known from other methods, such as the truncated quadratic cost functional proposed by Blake et al. [26] or the constant P_2 cost parameter known from semi-global matching [88].

In order to determine the minimum for Equation (3.40), the two terms $-\log\left(P_{\mathrm{uniform}}\right)$ and $-\log\left(P_{\mathrm{Gaussian}}\right)$ have to be computed. This can be done with significantly less effort than the evaluation of the term given in Equation (3.39). For this purpose, the two terms are computed by:

$$
\begin{aligned}
-\log\left(P_{\mathrm{uniform}}\right) &= \log\left(d_{\max} - d_{\min}\right) - \log\left(p_{\mathrm{out}}\right) \qquad (3.41)\\
-\log\left(P_{\mathrm{Gaussian}}\right) &= \log\left(A_{\mathrm{norm}}\right) + \log\left(\sigma_{c_n}\left(f_n, v\right) \cdot \sqrt{2\pi}\right) - \log\left(1 - p_{\mathrm{out}}\right)\\
&\quad + \frac{1}{2 \cdot \sigma_{c_n}^2(f_n, v)} \cdot \left(d_v - f_n(v)\right)^2
\end{aligned}
$$

Note that the only expression that actually depends on the disparity measurement d_v is the last one. It contains the quadratic part $\left(d_v - f_n(v)\right)^2$. All remaining terms are not related to d_v. Instead they depend on the sensor parameter configuration, the image row

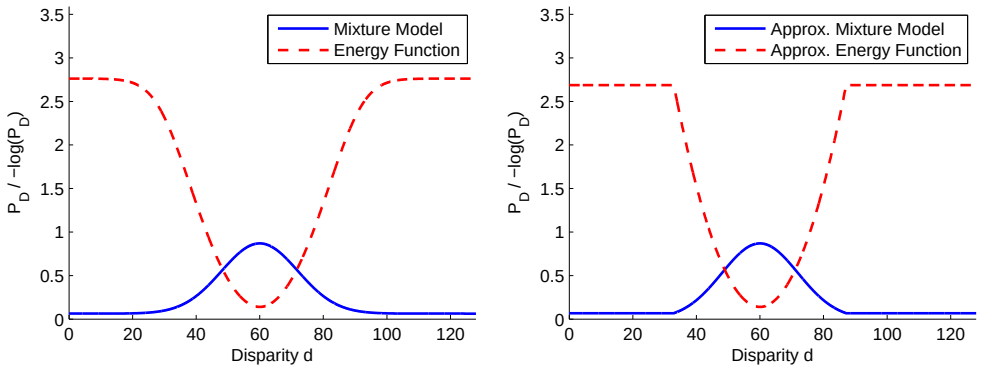


Figure 3.4: The left curve shows the accurate but hard-to-compute variant of the cost function. The right plot shows the optimized version with the shape of a truncated quadratic function for the data term costs.

coordinate v as well as the representative function f_n for segment s_n. Keep in mind that this issue can be exploited later on to increase efficiency during the cost pre-computation step.

3.6.2 Criteria for the Application of Dynamic Programming

As stated earlier, in order to be able to apply DP, the optimization problem has to fulfill two essential criteria. At first, the problem has to be of discrete nature. Secondly, the problem must exhibit optimal substructure. That means it can be expressed recursively as a composition of a set of smaller sub-problems of which each is then either decomposed further or solved optimally. The following sections discuss whether the requirements in order to use DP are met. If required, the necessary steps are sketched in order to make DP applicable. At first, the requirement for a discrete problem is discussed within subsection 3.6.2.1. Then, the property of optimal substructure is discussed and verified accordingly in subsection 3.6.2.2.

Note that the individual columns of the image are solved independently. DP is then used in order to compute the optimal segmentation along every column.

3.6.2.1 Discrete Optimization Problem

The optimization problem has to be of discrete nature in order to be able to apply DP correctly. That property is directly determined by the domain the problem is optimized for. For our purpose, that domain corresponds to the definition of the segments $s_n = \big(v_n^b, v_n^t, c_n, f_n(v)\big)$. Given that all elements of this definition originate from discrete domains, the composite would be of discrete nature as well and the given requirement would be fulfilled. Unfortunately, this is only the case for the image row coordinates v_n^b and v_n^t as well as the class property $c_n \in \{\mathrm{g}, \mathrm{o}, \mathrm{s}\}$ with $v_n^b, v_n^t \in \{0, \ldots, h-1\} \subset \mathbb{N}$. The last member f_n, which originates from the space of valid functions in order to approximate and represent the segment, is not of discrete nature. Even though the set of possible

functions f_n is restricted to linear functions, its actual domain remains continuous.

Note that this issue is very similar to the problem of stereo matching schemes that rely on DP (e.g. [164, 183, 219]). Those optimize for a two-dimensional disparity map with a disparity ideally as a real number $d \in \mathbb{R}$, such that the optimization space is initially not discrete. However, in order to be able to apply DP anyway, those schemes discretize the optimization space to a limited number of (usually equidistant) disparity steps. The actual number of steps directly determines the run-time performance (and memory consumption) of the matching algorithm.

For our purpose, a related principle will be followed and thus turn the problem at hand into a discrete form. However, we will not test for every function f_n explicitly, because the resulting computational complexity would increase significantly.

Instead, for the cost optimization we decide to choose a different approach. To this end, the function f_n is not considered as a direct part of the optimization step, but as a property of the segment s_n and the actual optimization is only performed on the first three variables. As a result, the first property in order to apply DP is implicitly ensured. The fourth variable f_n is determined by performing explicit checks on well selected function candidates. Hence, now the remaining challenge is to determine well suited candidate f_n alternatively. Further details and consequences of this decision are discussed in Section 3.6.3.2 and the subsequent sections.

3.6.2.2 The Property of Optimal Substructure

The second requirement that has to be met states that the optimization problem must exhibit optimal substructure. That is assured if the optimization problem allows to be expressed as a composition of smaller sub-problems. Each of those is then either decomposed further or solved optimally.

In order to show that the discussed optimization problem exhibits optimal substructure, a recursive definition of the optimization problem is given. According to Section 3.6.1.1, it will be formulated as a cost minimization task instead of a maximum a posteriori estimation problem.

For this objective, the following notation is introduced. The variables g_b^t, o_b^t and s_b^t denote the resulting data term costs from assigning measurement data within the range of the base point b and top point t to either *ground*, *object* or *sky*.

Further, let G^t, O^t and S^t denote the minimum aggregate costs in order to yield a segmentation from 0 to t. This segmentation can consist of multiple segments and the type of the last segment corresponds to the particular capital letter. E.g. O^{20} denotes a segmentation where the last segment is of type object with a top point row coordinate 20.

Also, let the functional c denote the model (a priori) costs in order to rate different constellations of adjacent segments. For instance, $c(G^{10}, o_{11}^{20})$ denotes the model costs for the combination of a segmentation ending with type *ground* at top point 10 adjacent to another *object* segment from 11 to 20. Or to give a unary example, $c(o_0^{10})$ denotes the model costs for the first segment o_0^{10} labeled as *object*.

At first, the resulting costs for a segmentation of length one are determined. This can

be done straightforward by evaluating:

$$C^0 = \{G^0, O^0, S^0\} \qquad \{g_0^0 + c(g_0^0), o_0^0 + c(o_0^0), s_0^0 + c(s_0^0)\} \tag{3.42}$$

Consequently, the segmentation from 0 to 1 can either consist of a single segment as well or utilize the previous result C^1 in the process, such that:

$$C^1 = \{G^1, O^1, S^1\} \tag{3.43}$$

$$G^1 = \min \begin{cases} g_0^1 + c(g_0^1) \\ g_1^1 + c(g_1^1, G^0) + G^0 \\ g_1^1 + c(g_1^1, O^0) + O^0 \\ g_1^1 + c(g_1^1, S^0) + S^0 \end{cases} \qquad O^1 = \min \begin{cases} o_0^1 + c(o_0^1) \\ o_1^1 + c(o_1^1, G^0) + G^0 \\ o_1^1 + c(o_1^1, O^0) + O^0 \\ o_1^1 + c(o_1^1, S^0) + S^0 \end{cases}$$

$$S^1 = \min \begin{cases} s_0^1 + c(s_0^1) \\ s_1^1 + c(s_1^1, G^0) + G^0 \\ s_1^1 + c(s_1^1, O^0) + O^0 \\ s_1^1 + c(s_1^1, S^0) + S^0 \end{cases}$$

Continuing with that scheme leads to C^2:

$$C^2 = \{G^2, O^2, S^2\} \tag{3.44}$$

$$G^2 = \min \begin{cases} g_0^2 + c(g_0^2) \\ g_1^2 + c(g_1^2, G^0) + G^0 \\ g_1^2 + c(g_1^2, O^0) + O^0 \\ g_1^2 + c(g_1^2, S^0) + S^0 \\ g_2^2 + c(g_2^2, G^1) + G^1 \\ g_2^2 + c(g_2^2, O^1) + O^1 \\ g_2^2 + c(g_2^2, S^1) + S^1 \end{cases} \qquad O^2 = \min \begin{cases} o_0^2 + c(o_0^2) \\ o_1^2 + c(o_1^2, G^0) + G^0 \\ o_1^2 + c(o_1^2, O^0) + O^0 \\ o_1^2 + c(o_1^2, S^0) + S^0 \\ o_2^2 + c(o_2^2, G^1) + G^1 \\ o_2^2 + c(o_2^2, O^1) + O^1 \\ o_2^2 + c(o_2^2, S^1) + S^1 \end{cases}$$

$$S^2 = \min \begin{cases} s_0^2 + c(s_0^1) \\ s_1^2 + c(s_1^2, G^0) + G^0 \\ s_1^2 + c(s_1^2, O^0) + O^0 \\ s_1^2 + c(s_1^2, S^0) + S^0 \\ s_2^2 + c(s_2^2, G^1) + G^1 \\ s_2^2 + c(s_2^2, O^1) + O^1 \\ s_2^2 + c(s_2^2, S^1) + S^1 \end{cases}$$

With C^2 the recursive nature of the followed procedure becomes more obvious, and

hence leads to the generic definition of C^n:

$$C^n = \{G^n, O^n, S^n\} \tag{3.45}$$

$$G^n = \min \begin{cases} g_0^n + c(g_0^n) \\ g_1^n + c(g_1^n, G^0) + G^0 \\ g_1^n + c(g_1^n, O^0) + O^0 \\ g_1^n + c(g_1^n, S^0) + S^0 \\ \vdots \\ g_n^n + c(g_n^n, G^{n-1}) + G^{n-1} \\ g_n^n + c(g_n^n, O^{n-1}) + O^{n-1} \\ g_n^n + c(g_n^n, S^{n-1}) + S^{n-1} \end{cases}$$

The remaining terms O^n and S^n follow according to the definition of G^n. Finally, in order to find the optimum segmentation, C^{h-1} has to be computed. The minimum costs of $C^{h-1}_{\min} = \min(G^{h-1}, O^{h-1}, S^{h-1})$ mark the beginning of the optimal path for the backtracking process to start.

3.6.3 Implementation Details

The following sections aim to give insight into some essentials of the implementation details. They sketch how to design the cost and index tables for the DP step and outline possible optimizations and approximations in order to improve the run-time performance.

3.6.3.1 Computation of the Cost Tables

The computation of the cost table is the key element of the dynamic programming part. The size of the cost table directly relates to the search space of the particular optimization problem. For our purpose, the table dimensions depend on the image height (due to the possible positions for cuts between segments) and the number of possible labels. Thus, a cost table of size $h \times 3$ is computed and evaluated for every column of the image. Due to the mutual independence of the columns, all columns allow to be solved independently (and thus also in parallel).

To increase the efficiency of the backtracking step, an additional index table is built. Its size is identical to the cost table. The indices correspond to image row coordinates and label types. They link different entries of the cost table and thus chain a possible optimal path of segments. In order to find the best column segmentation, the cost and index tables are computed completely. The optimum path is traversed reversely from the top of the cost table to its bottom within the backtracking process. Thereby the optimum column segmentation is extracted.

The working principle for the cost table is illustrated for an exemplary column with a height of nine pixels in Figure 3.5. Each table entry consists of two parts: the minimum costs that could be determined for that cell (marked in color) and a pair of indices which relate to the previous segment. The first index denotes the preceding label type (column

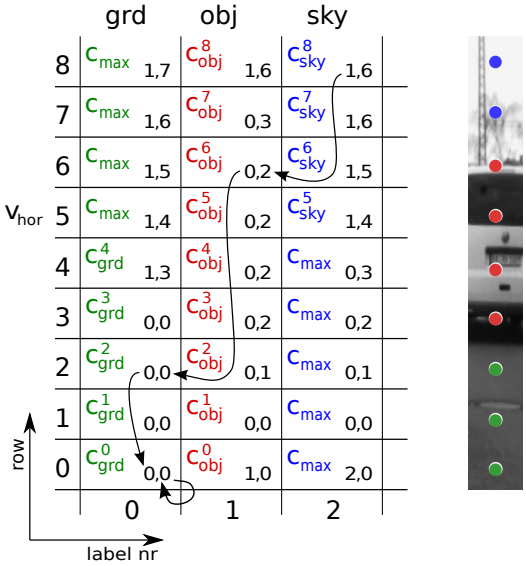


Figure 3.5: Illustration of an exemplary cost and index table for an image column with 9 (virtual) pixels in height. The entries mark the resulting minimum costs that are determined for each cell. Each pair of indices encodes the predecessor that led to these costs. They also reveal the class type and top point coordinate of that predecessor. The chain with the smallest end-point costs is traversed (backtracking step) until it ends with a self-loop in the bottom row.

of the cost table) and the second index marks the row of the corresponding top point coordinate v^t. In order to perform the backtracking step, the minimum costs within the upper row are determined. In this example, the minimum costs are reached by c^8_{sky}. Hence, the first segment extracted is labeled as *sky*. The corresponding column and row index reveal the label type and top coordinate v^t of the previous segment, the one that led to the current costs. In the given example it relates to an object segment starting at row 6. This repeats until the chain ends with a self-loop at the bottom row. Note that some fields of the table are marked with c_{max} which accounts for maximum (or infinite) costs. This effect results from the a priori modeling (e.g. no street occurrence above the horizon).

3.6.3.2 Discussion of the Run-Time Performance

Naturally, when using DP the largest amount of computation time is required for filling the cost table(s). While the actual design for these tables is highlighted in 3.6.3.1, this subsection focuses on the required computational effort for their computation. For this purpose, a non-optimized straightforward implementation is discussed with respect to run-time complexity. Furthermore, an approach is presented with the objective to make the cost table computation more efficient.

Note that all Stixels are given a fixed width of w_s px. As a result of that parameter, only

Algorithm 3.1 Straightforward approach for the computation of the cost table

```
D:=getCurrentDisparityImage()       //access via [col][row]
for u:=0 to w-1 stepsize w_s        //image column coordinates
    for vT:=0 to h-1                //image row coordinates
        for vB:=0 to vT             //image row coordinates
            for all fn              //check against allowed functions
                cG_data:=0
                cO_data:=0
                cS_data:=0
                //compute data terms costs
                for v:=vB to vT     //go vB to vT
                    d:=D[u,v]       //traverse all disparities
                    cG_data+=getDataCostsGround( vB, vT, fn, d )
                    cO_data+=getDataCostsObject( vB, vT, fn, d )
                    cS_data+=getDataCostsSky( vB, vT, fn, d )
                next v
                //compute priors costs
                cG_model:=getModelCostsGround()
                cO_model:=getModelCostsObject()
                cS_model:=getModelCostsSky()
                //update cost tables
                updateCostTables()
            next fn
        next vB
    next vT
next u
```

every $w_s th$ column has to be computed. Therefore, neighboring measurement columns are fused row-wise in groups of w_s. This is achieved efficiently in a pre-processing step by using a horizontal median. This filter offers noise reduction and increases robustness by suppressing outliers reliably. Further, this operation is edge-preserving and thus only outputs values that have been an actual measurement.

According to the reasoning in Section 3.3.2, the different columns are considered as independent. This property offers the opportunity to compute the individual columns in parallel, which of course improves the run-time performance significantly.

Irrespective of particular details, computing the cost tables for a straightforward implementation results in five nested loops. The first one is used in order to traverse all columns $0 \leq u < w$ for which the Stixel representation is supposed to be extracted.

Then two further loops follow, one to traverse all top points $0 \leq v^t < h$ and one to check for all possible base points $0 \leq v^b \leq v^t$ of the current segment. Within these loops, possible functions f_n for the labels $c_n \in \{\mathrm{g}, \mathrm{o}, \mathrm{s}\}$ are checked in order to rate the segment $s_n = \left\{v_n^b, v_n^t, c_n, f_n\right\}$ in combination with its possible predecessor segments s_{n-1} such that $v_{n-1}^t = v_n^b - 1$.

Depending on the individual solution, checking for different functions f_n can lead to a fourth loop, e.g. when the domain for the tested functions f_n is discretized. For details

Algorithm 3.2 Optimized version for computing the cost tables

```
D:=getCurrentDisparityImage()       //access via [col][row]
for u:=0 to w-1 stepsize w_s        //image column coordinates
    for vT:=0 to h-1                //image row coordinates
        for vB:=0 to vT             //image row coordinates
            fnGrd:=getHypothesisForFnGrd()
            fnObj:=getHypothesisForFnObj( vB, vT )
            fnSky:=getHypothesisForFnSky()
            //compute data terms costs
            cG_data:=getCostForGroundFromLUT( vB, vT, fnGrd )
            cO_data:=getCostForObjectFromLUT( vB, vT, fnObj )
            cS_data:=getCostForSkyFromLUT( vB, vT, fnSky )
            //compute priors costs
            cG_model:=getModelCostsGround()
            cO_model:=getModelCostsObject()
            cS_model:=getModelCostsSky()
            //update cost tables
            updateCostTables()
        next vB
    next vT
next u
```

refer to the related discussion in Section 3.6.2.1. Finally, all disparities d_v within the range of the currently considered segment s_n are rated in respect of the representative function f_n. This step is accounted for by a fifth loop. The described procedure is exemplified by Algorithm 3.1.

In terms of the Landau notation [193], the resulting complexity is roughly estimated as $\mathcal{O}\left(w/w_s \times h \times h/2 \times |\mathbb{C}| \times |f_n| \times h/4\right)$ which equals $\mathcal{O}\left(3/8 \times w/w_s \times h^3 \times |f_n|\right)$.

For the experiments presented later in this section, stereo image pairs of size $1024 \times 440\,\text{px}$ are used and the Stixel width is chosen as $w_s = 5\,\text{px}$. Assuming that the object functions f_n are sampled to 128 disparity steps and the classes *object*, *ground* and *sky* have to be checked for every pixel in vertical direction, this setup results in running through the most inner loop approximately 10^{12} times. There is no doubt that for currently used consumer hardware this is too much to handle when aiming for a real-time capable solution.

For this reason, the attention is focused on how to compute the cost tables more efficiently. Irrespective of the final solution, the three outer loops for iterating the columns u, the top and base points v^t and v^b cannot be left out. However, this is different for the inner loop. Algorithm 3.2 describes a procedure where those two loops are spared.

The key idea for this to work is to check the segment s_n only for selected candidate functions f_n and to maintain the corresponding costs for the three different classes in a pre-computed form. This approach allows to yield a run-time complexity of $\mathcal{O}\left(3/2 \times w/w_s \times h^2\right)$ which, for the configuration noted above, adds up to passing the innermost loop approximately 10^7 times. Even though this is considerably less, it should

Algorithm 3.3 Pre-computation for object costs

```
D_u:=getCurrentDisparityCol()       //access via [row]
Costs:=getCostLUT()                 //access via [fn][row]
tmpSum[max_fn+1]                    //temporal cost storage,
                                    //initialized to 0
for v:=0 to h-1                     //cycle row coordinates
    d:=D_u[v]                       //get current disparity
    for fn:=0 to max_fn             //cycle fn hypothesis
        if(d is valid)
            tmpSum[fn]+=getCost(fn, v, disp)
        endif
        Costs[fn][v]=tmpSum[fn]
    next fn
next v
```

be kept in mind that the pre-processing requires a certain computation time as well.

More details on the pre-computation steps are given in the following subsections.

3.6.3.3 Pre-Computation of the Data Terms

The key challenge for the data term cost computation is to determine if a disparity measurement d_v has to be considered as an outlier. According to the sensor measurement model from Equation (3.16), this decision depends on the particular choice for a representing function f_n. The resulting costs either rise in a quadratic fashion or remain saturated on a maximum level (see Figure 3.4).

For the label types *ground* as well as *sky* this is not an issue, since both labels have a unique and explicit model expectation. For *ground* that is the expected gradient function given by $f_n^g(v) = \alpha \cdot (v_{\text{hor}} - v)$, and *sky* regions are checked against $f_n^s(v) = 0$. In contrast, objects can reside at every depth within the working disparity range $[d_{\min}, d_{\max}]$.

Unfortunately, the cost computation for every individual disparity measurement d_v is expensive, for which the strong non-linear characteristic of that cost function leaves barely room for optimization. Hence, computing the cost within the innermost loop of the cost table computation is not an option.

Therefore, the goal is to determine the resulting data costs for a segment configuration $s_n = \{v_n^b, v_n^t, \text{o}, f_n\}$ with the least necessary effort. This is achieved by means of a look-up table (LUT) that is precomputed in advance to the actual cost table computation. That LUT allows to directly determine the resulting object data cost for that segment s_n. Instead of pre-computing the cost for every possible base point v_n^b and top point v_n^t, the costs are stored using integral tables in a partially finished form. Hence, it is accessed via the row coordinates v_n^b and v_n^t as well as f_n. An exemplary pseudo-code for the computation of such a LUT is depicted by Algorithm 3.3.

Accordingly, the costs for an *object* segment configuration $s_n = \{v_n^b, v_n^t, \text{o}, f_n\}$ is determined by evaluating $c_{\text{o}}(v_n^b, v_n^t, f_n) = \text{Costs}[f_n][v_n^t] - \text{Costs}[f_n][v_n^b - 1]$. The run-time complexity for the sketched LUT pre-computation corresponds to $\mathcal{O}\left({}^{w}/_{w_s} \times h \times |f_n|\right)$,

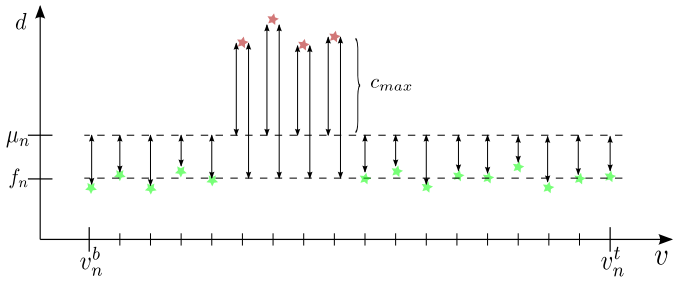


Figure 3.6: Exemplary set of disparity measurements within v_n^b and v_n^t. Inliers are marked green, outliers are red. This illustration shows two aspects. Firstly, due to the truncated cost function, the resulting costs for each outlier are identical. Secondly, in case of having outliers, using the mean for estimating f_n results in an inappropriately high penalty for each inlier. Hence, using the mean does not guarantee to yield the minimum data costs.

which is tolerable given the direct comparison to complexity of the straightforward approach. Nevertheless, depending on the hardware platform, the memory requirements for the LUTs might not be negligible (e.g. when thinking of embedded hardware solutions). The required LUT size equals to $h \times |f_n|$ for every column that is segmented.

Pre-computing the data terms for *ground* and *sky* follows accordingly, except that for each LUT a single one-dimensional array of size h is sufficient, since both labels suffice being checked against their known model.

3.6.3.4 Increasing Robustness by Robust Mean Estimation

A remaining and central question regards the choice for the particular candidate function f_n that is used to obtain the data term costs for the segment s_n. Assuming, the used measurement model would consist of a Gaussian distribution only, the optimal candidate would be the mean disparity value of all disparities within the range of segment s_n. This is due to the fact that the mean would yield the minimum variance. The mean allows to be computed efficiently by means of two sums of the following form:

$$\begin{aligned} \text{sum}[i] &= \sum_{v=0}^{h-1} \exists(d_v) \cdot d_v \\ \text{valid}[i] &= \sum_{v=0}^{h-1} \exists(d_v) \end{aligned} \tag{3.46}$$

Using these sums, the mean disparity within the range of v^b to v^t and thus the candidate function for f_n is determined by:

$$f_n := \text{mean}(v^b, v^t) = \frac{\text{sum}[v^t] - \text{sum}[v^b - 1]}{\text{valid}[v^t] - \text{valid}[v^b - 1]} \tag{3.47}$$

Anyhow, for our purpose, the proposed measurement model is a mixture model, such that the mean does not essentially correspond to the optimum estimation. This circumstance is visualized in Figure 3.6. Yet, the mean typically is a good initial value and works perfectly for non-disturbed data.

The problem is that both outliers and inliers contribute to the mean value estimation equally. This issue can be accounted for by means of an iterated robust mean estimation. To this end, the choice for a particular method is not fundamental (see [235, 7] for possible implementations). However, it is advisable to rely on a scheme that allows for a robust mean estimation in a single pass and does not require complex re-organization of the disparity data such as ordering.

For this purpose, good results are obtained by using the following scheme:

$$f_n = \frac{\sum_{v=v_n^b}^{v_n^t} \omega_v \cdot d_v}{\sum_{v=v_n^b}^{v_n^t} \omega_v} \qquad (3.48)$$
$$\omega_v = \frac{\exists(d_v)}{1 + |d_v - \mu|}$$

The corresponding weights ω_v are computed from the mean disparity $\mu = \text{mean}(v^b, v^t)$, using the previously described computation method. A weight is computed individually for every disparity measurement. They are inversely proportional to the distance to the mean value μ. Note that this procedure increases the run-time complexity of the cost table computation from $\mathcal{O}\left(3/2 \times w/w_s \times h^2\right)$ to $\mathcal{O}\left(3/8 \times w/w_s \times h^3\right)$ significantly.

Anyhow, instead of testing against multiple discretized hypothesis functions f_n the proposed method yields a very precise and robust candidate function f_n for the cost computation of the data term of segment s_n.

3.7 Results

In the following section qualitative results for the computation of the multi-layered Stixel World are presented. For this purpose we will focus on the domain of traffic scenarios or other environments related to vehicles.

First, the testing platform is specified briefly and the basic data of the camera system and the processing hardware are given. Secondly, different scenarios are depicted that exhibit certain characteristics. Those are highlighted and discussed in the process. Thirdly, limitations of the proposed approach with respect to the models are shown. Finally, the performance of the algorithm under adverse weather and poor lighting conditions is discussed.

3.7.1 The Testing Vehicle and Hardware Setup

For the experiments a stereo camera system with a resolution of 1024×440 px, a lens with a field of view of 42°, a focal length of $f_u, f_v \approx 1250$ px and a base line of 22 cm is used. The camera system is mounted to the testing vehicle behind the windshield at a height

of $h_{\text{cam}} = 1.17\,\text{m}$. It is slightly tilted downwards with an angle of $\alpha_{\text{cam}} = 0.063\,\text{rad}$. The installation setup is depicted in Figure 3.7.

Figure 3.7: Installation of the stereo camera system in the testing vehicle. The camera setup has a base line of 22 cm. Each camera has a resolution of $1024 \times 440\,\text{px}$. The captured image format is gray-value with a depth of 12 bit per pixel.

For the semi-global matching stereo we rely on an implementation that runs on dedicated low-power FPGA hardware at a rate of 25 Hz [73]. The output range for valid disparities starts from $d_{\text{min}} = 0$ and goes up to $d_{\text{max}} = 128$.

For both label types *object* and *ground*, a disparity measurement uncertainty $\sigma_d^{\text{g,o}} = 0.5\,\text{px}$ as well as an outlier rate of $p_{\text{out}}^{\text{g,o}} = 0.15$ is assumed. For the label *sky* $\sigma_d^{\text{s}} = 0.2\,\text{px}$ and $p_{\text{out}}^{\text{s}} = 0.4$ are chosen. To properly cope with invalid disparities, we empirically set $p_{\not\exists}^{\text{o}} = 0.3$, $p_{\not\exists}^{\text{s}} = 0.36$ and $p_{\not\exists}^{\text{g}} = 0.34$ as well as $P(\exists(d_v) = 0) = 0.25$ and $P(c_n) = 0.33\ \forall\, c \in \mathbb{C}$ (for details see 3.4.1).

The outlier rate for *sky* is set strikingly high. This is due to the fact that those regions typically exhibit very hard-to-match texture information and thus firmly support the occurrence of outliers. The given parameter choice is considered a fail-safe configuration to work reliably for all typical traffic scenarios, weather and lighting conditions. If the environmental conditions are sufficient, these parameters can be tightened severely. However, this issue is not supposed to be made a central subject of this work.

All processing steps of the segmentation algorithm are done on the CPU (Core-i7 980X, 6×3.4 Ghz, 6 GB of RAM). This way, all pre-computation for a stereo image pair of size $1024 \times 440\,\text{px}$ and a Stixel width of 5 px is done within 15 ms. The solving step via dynamic programming runs within 60 ms.

Due to the smoothing characteristic of SGM scaling down the image in height by a factor of two comes without noteworthy impact to the reconstruction quality, but allows to reduce the computation time significantly. Thus, solving is done in real-time within 15 ms. Nevertheless, the time required for the pre-computation remains unchanged, because all look-up tables are still computed at full resolution in order to minimize the potential loss of accuracy as a result of scaling. Nevertheless, in case that maximum performance is an issue, computing the LUTs from the scaled images would result in a linear decrease of their computation time depending on the used scaling factor.

3.7.2 Results for Different Scenarios

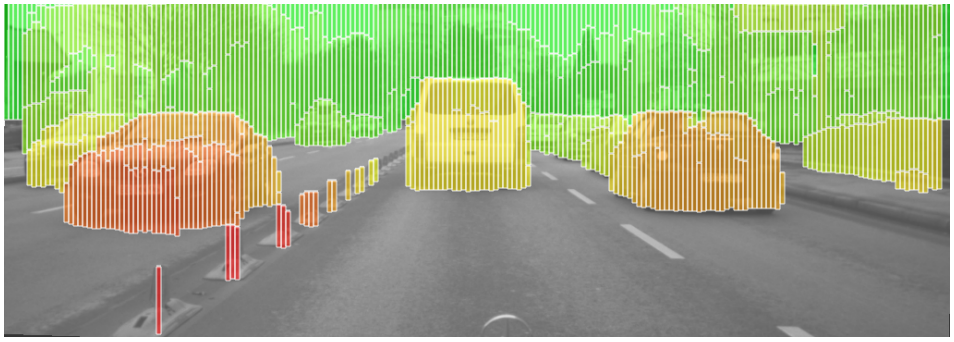
For a comprehensive testing, typical situations from different scenarios of open road traffic have been chosen. This includes urban environments, rural roads as well as highway scenarios. Those setups exhibit various lighting and weather conditions. Further, exemplary situations and constellations are depicted that show the limits of the suggested model types. For instance, certain types of objects or surfaces are not supported natively, such as closed environments with ceilings. The actual labeling quality is inspected with respect to the different classes *ground*, *object* and *sky*. In addition, the proposed algorithm is used to process 3D measurement data obtained from using a Velodyne HDL-64E S2 LIDAR [85]. For this purpose the most significant characteristics of this sensor and their influence to the algorithm are discussed briefly.

For all visualizations a unified color scheme is used. Stixels are drawn with colors from red over yellow to green. Red means the object is very close, green is used for objects far away. For the labeling results we use a tripartite color scheme with green for *ground* surface, red for *objects* and blue for *sky* segments.

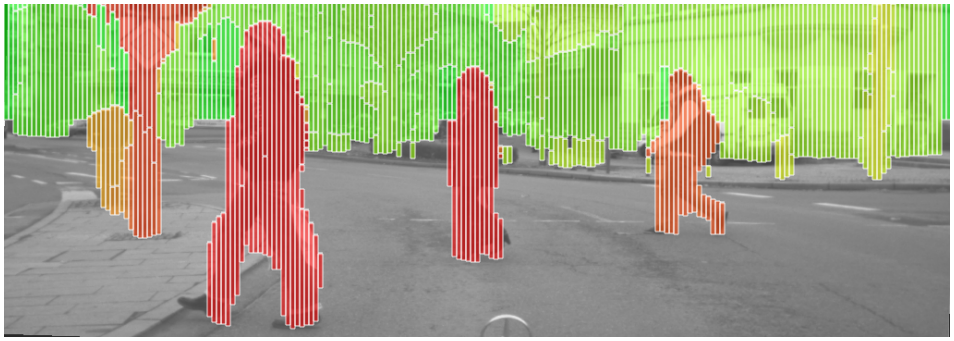
At first, qualitative results for typical urban scenarios are depicted in Figure 3.8. Figure 3.8a shows a traffic situation with multiple cars and objects at different depths, obstacles on the side walk and well-structured background. A pedestrian crossing with objects of rather complex shape and silhouette are given in Figure 3.8b. The third urban example depicted in Figure 3.8c contains a considerably crowded environment with a line of cars located at multiple depths and a bicycle on the sidewalk as well as a complex combination of rather thin objects in the background.

Rural road and highway scenarios are assembled in Figure 3.9. At first, Figure 3.9a shows a construction site in a highway environment. The scene features far sight and obstacles of multiple shape, size and distance. On the left side, a small wall separates our lane from the opposing driving direction. The area behind that barrier is detected as ground surface correctly. The second example, Figure 3.9b, was captured in a tunnel. It features poor lighting as well as weak texture information (mainly due to long exposure times) and strong reflectance on the road surface. Thus, the depth reconstruction of this scenario is very challenging for the stereo matching scheme. Thirdly, Figure 3.9c shows an exemplary rural road scenario. It features far sight and multiple object. Note that our used surface model is not capable to represent the decline of the ground surface to our right. This is due to the violation of the assumption of having a planar ground surface. This effect is discussed separately later in this section.

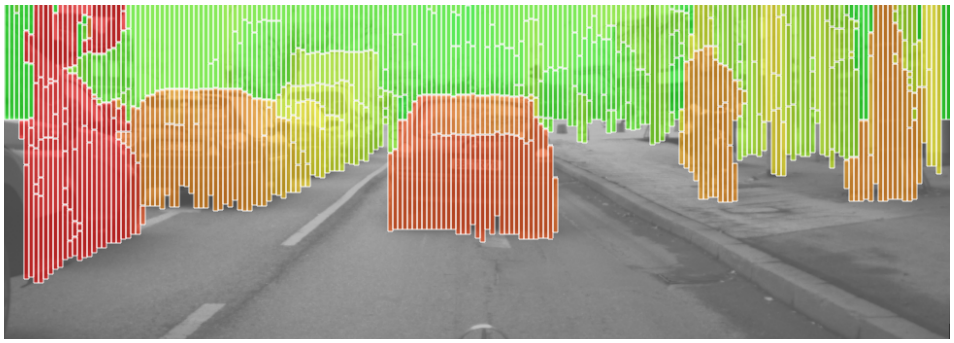
Next, Figure 3.10 relates to typical problems that result from adverse weather conditions. For this purpose a highway scenario has been chosen. The images suffer from strong rain, mist, severe reflectance on the road, low-textured surfaces as well as water droplets on the windshield. For a better visual impression, the raw gray-value image, the disparity result from stereo matching as well as the resulting Stixel representation are shown. The turbulences within the depth map lead to considerably strong noise and outliers that occur randomly as connected patches. Accordingly, these inflict the occurrence of phantom Stixels. However, the random characteristic of this behavior can be exploited by evaluating temporal coherences. For instance, this is done within the



(a) Urban scenario with multiple cars up to a distance of 80 m. The small obstacles are pylons to separate the two opposing driving lanes. They have a height of about 35 cm. The background is cluttered and contains concrete walls, trees, houses, traffic signs as well as fragments of sky.

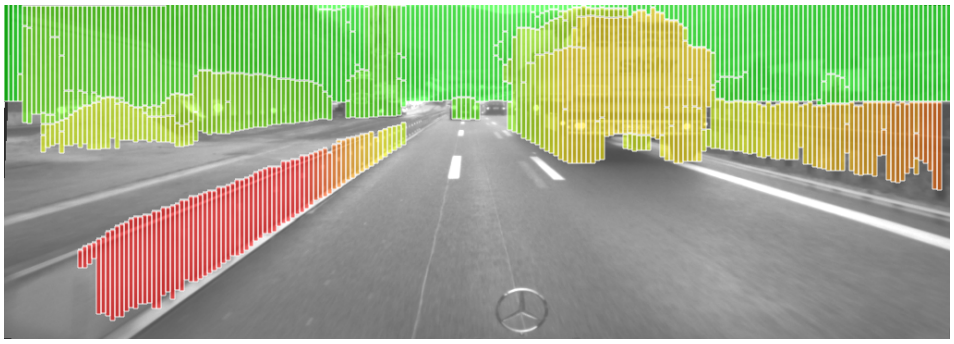


(b) Pedestrian crossing with objects located at multiple depths. Note the quality of the segmentation with respect to the silhouettes of the pedestrians. This scenario exhibits multiple violations of the gravity constraint (e.g. the legs of the people) and the ordering constraint (e.g. the traffic light, signs and trees).

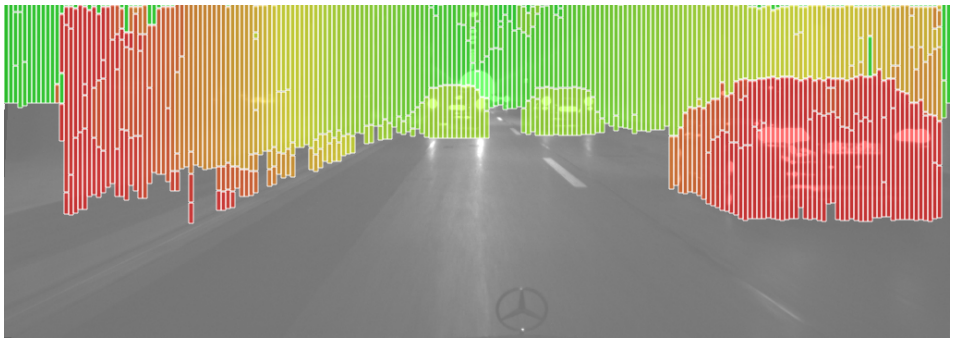


(c) Considerably complex scenario that contains a leading vehicle, a line of piled cars on the left and a cyclist with cluttered background (trees in front of a house) on the right. Note how the algorithm is capable to segment the outlines of the individual cars precisely.

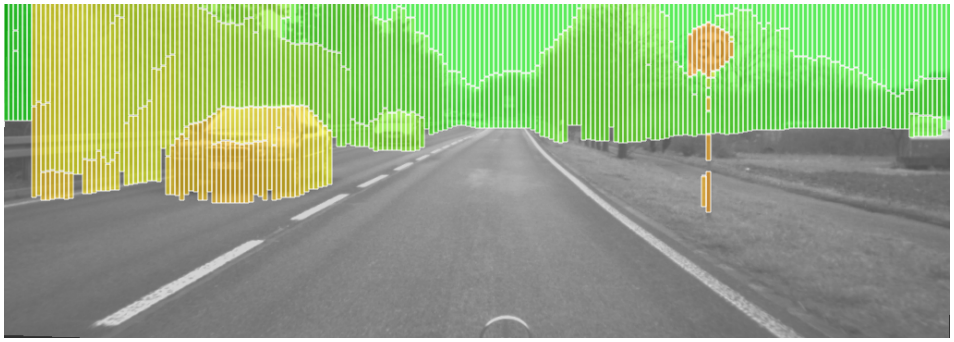
Figure 3.8: Visualization of the segmentation for three different urban scenarios. The color scheme visualizes distance with red for close and green for far away.



(a) Complex highway scenario with far sight and multiple rows of objects. The barrier on the left has a height of 60 cm and is segmented up to a distance of 35 m. The leading vehicle has a distance of 75 m. The quality of the stereo data even allows to segment the fourth covered wheel of the truck. The actual label types for this scenario are visualized in Figure 3.12b.



(b) Highway tunnel scenario with three leading vehicles and concrete walls on both sides of the road. This scene exhibits challenging conditions due to low light and strong road reflectance. On the left side there is a curb with a height of approximately 30 cm.



(c) Rural road environment. The crashing barrier merges with the bushes on the left. Two cars are segmented along with a traffic sign. The metal pipe holding the traffic sign is not reconstructed properly by the stereo matching. The field on the right side has a strong vertical slope which is not accounted for by the used ground expectation model.

Figure 3.9: Visualization of different scenarios depicting a highway, a tunnel and a rural road. The examples exhibit both: Far sight as well as objects at close range. The coloring scheme denotes the distance to the objects.

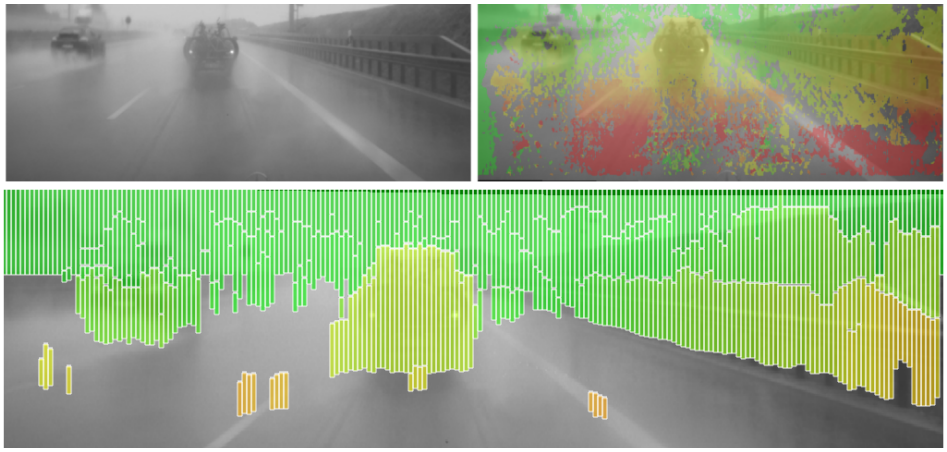


Figure 3.10: Highway scenario with adverse weather conditions: Strong rain, spray of water, mist, strong reflectance on the road, low-textured surfaces as well as water droplets on the windshield. The stereo depth map is very noisy and cluttered which leads to phantoms that occur randomly. Despite these circumstances, the two leading vehicles and the crash barrier on the right are segmented quite well. The phantoms are suppressed by using a tracking scheme as introduced in the following chapter.

Stixel tracking scheme as proposed within the next chapter that focuses on the motion estimation for the objects.

The next two figures show borderline cases or violations of the used models for *ground* and *object*. At first, Figure 3.11a shows a streetcar stop with a very high curb in order to allow comfortable boarding for the passengers. The elevation of this surface does not conform to the ground model expectation computed from the camera geometry. Accordingly, the data term for *ground* yields considerably high cost penalties for every pixel that would be assigned to that label. Instead, the sidewalk is approximated by means of multiple staggered object segments. That is reasoned by the fact that the model for large objects does not match well itself, because the surface is still oriented horizontally and not vertically. Consequently, the resulting objects are segmented rather often in order to yield data term errors as small as possible. A similar behavior occurs on strong vertical ascent of the ground surface or faulty tilt angles of the camera orientation. In order to be able to account for such cases, the ground model is extended accordingly in Section 3.8.1.

Secondly, Figure 3.11b depicts an underground parking lot. This scene exhibits several challenges for the three-dimensional reconstruction, such as the presence of a ceiling, poor lighting conditions as well as reflective surfaces. The ceiling is represented by means of multiple object segments, just like the elevated sidewalk. Even though, the ceiling could be considered by a dedicated geometry model, we regard this characteristic as not significant for our purpose. Additionally, doing so would increase the possible degree of freedom of the system. This bears the risk of ambiguous behavior.

However, yet another quality of the proposed scheme becomes obvious in this example. It is mandatory for every pixel to be assigned to only one of the available classes. As a consequence, even under such challenging conditions, good results are achieved in terms of the visual range.

At last, the class label results are shown. Until now, the given illustrations have only visualized the resulting segmentation into staggered vertical segments, but did not account for the actual type of the label. Therefore, Figure 3.12 displays the corresponding classes for the segmentation results shown in Figure 3.9a and Figure 3.9c.

3.8 Extensions

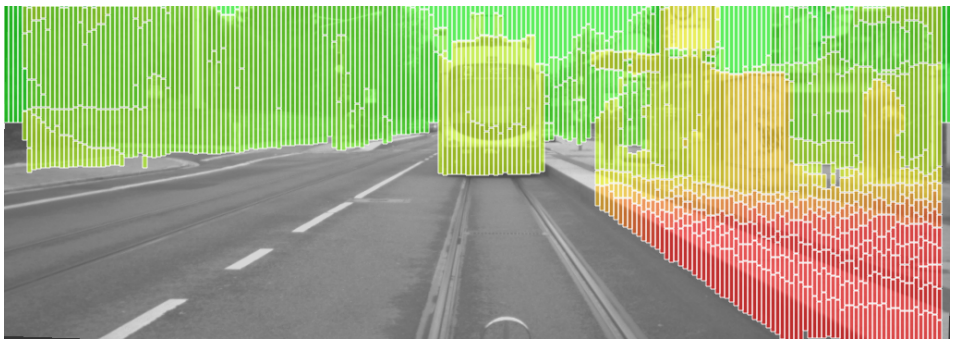
The previous section made a few limitations of the used models apparent. For instance, it was shown that modeling the ground surface strictly by using a single hypothesis function solely does not suffice for all situations. For this reason, the algorithm is extended with respect to two aspects. At first, we discuss how to incorporate an estimation for the ground surface height. That procedure allows to correctly represent elevated surfaces, such as sidewalks and curbs. Secondly, the suggested scheme is supported by incorporating a vertical road course profile which encodes the height of the road surface in front of the vehicle. This profile is obtained from an external road course estimation. Exemplary results are given for both.

Another aspect that we intend to highlight is the choice of the sensor type. So far, the proposed approach was utilized to work with stereo data. However, it is not limited to that, as it can also be applied to data obtained from other sensors as well. This will be exemplified by applying the algorithm to 3D measurement data obtained from a 360 degree LIDAR. Moreover, it can even be used for sensor fusion in order to process different types of input data from multiple sources in a unified scheme. This idea is discussed with the objective to use 3D stereo data and optical flow information for the segmentation process.

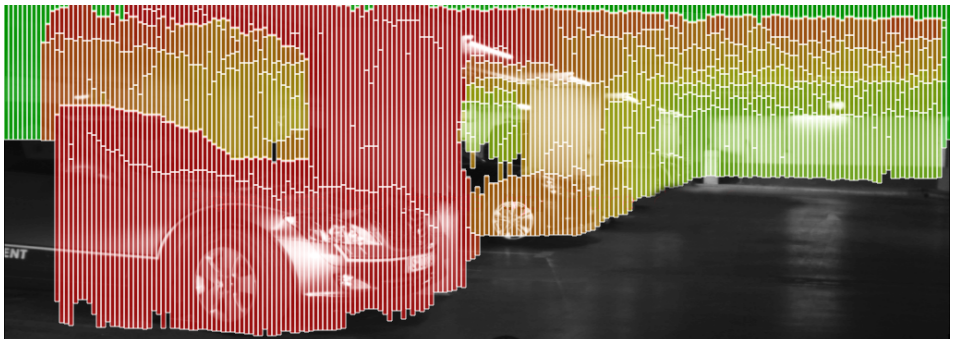
3.8.1 Ground Level Estimation

The given example with the elevated ground surface in Figure 3.11a makes it obvious that the used ground model by default does not suffice for all situations. The actual problem is that the sidewalk has a different height compared to the expected ground surface model. This results in a discrepancy between the disparity measurements and the expected function f_n^g. That, on the other hand, leads to considerably high costs for the data term with regard to the label *ground* and thus yields an over-segmentation by means of multiple small objects instead of a single ground segment. This issue is visualized in Figure 3.13.

In order to take that effect into account, the deviation between the initial ground expectation $f_n^g(v)$ to the measured data is estimated. This is identical to accounting for an incorrect height of the camera rig. The model violation is considered as an offset in disparities that is expressed by means of the function $\Delta_g(v)$. This allows to compensate

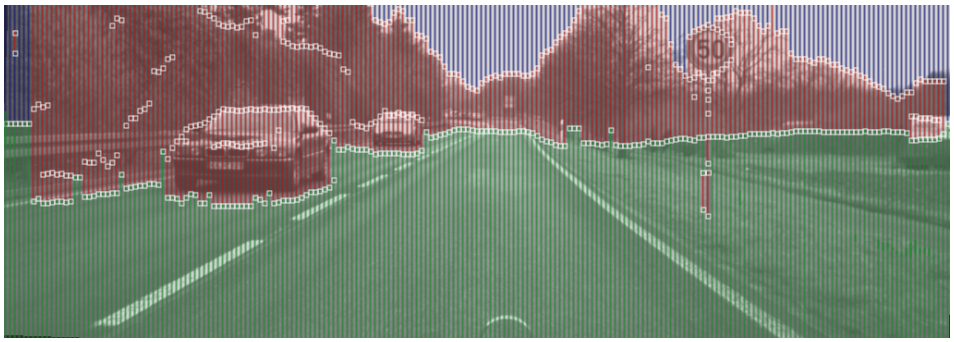


(a) The sidewalk of the tram station has an elevation of approximately 25 cm compared to the road surface and the expected surface model. This deviation results in comparably large penalties for the ground model data term. Anyhow, the object model does not match either. Thus, the whole sidewalk is segmented as a cascade of multiple small obstacles in order to keep the costs for data term small. For this problem Section 3.8.1 proposes a solution by estimating the elevation of the ground surface.

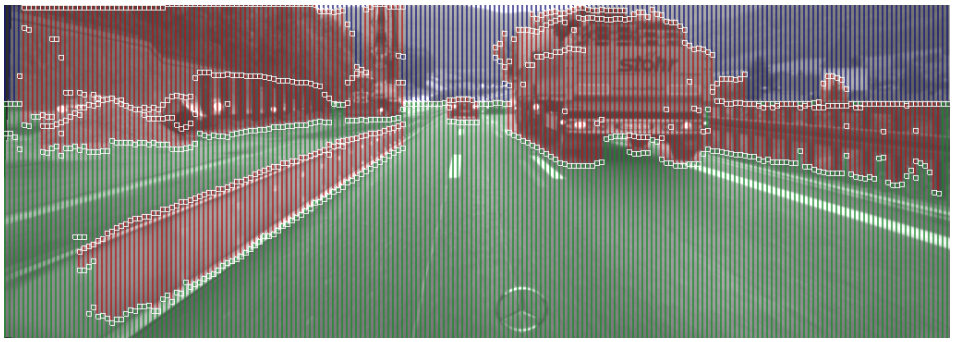


(b) An underground parking lot is shown. This scenery exhibits several challenges, such as very poor lighting as well as shiny and reflective surfaces. Also, the scene has a ceiling which is a class type that is not accounted for specifically. In terms of energy minimization the algorithm approximates the ceiling with multiple small object segments, similar to the example above.

Figure 3.11: The proposed scheme is not capable to reconstruct all possible constellations for *ground* and *objects* properly which is exemplified by means of the given illustrations. This is reasoned by insufficient or incomplete models. Particular limitations are dealt with, such as the problem of elevated ground surfaces. Others, such as the over-segmentation of the ceiling, are not crucial for our purpose.



(a) Labeling results for the rural road example. The row of trees is segmented against *sky* quite accurately. The street surface rises slowly and thus has a slight vertical ascent. Though, this causes the distant trees to intersect the road surface a tiny bit.



(b) Labeling result for the highway scenario. The distant background is segmented as *sky* and the area behind the small crash barrier is assigned to the label type *ground*. The small barrier, the left side of the scene, the vehicles, and even tree behind the fence are labeled to type *object*. The closest part of the barrier is not segmented properly, because it is not visible in the right camera image.

Figure 3.12: Labeling for the segmentation result from Figure 3.9a and Figure 3.9c. Green areas represent *ground*, red stands for *objects* and blue corresponds to *sky* labeled regions. The white rectangles visualize the positions of horizontal cuts between neighboring segments.

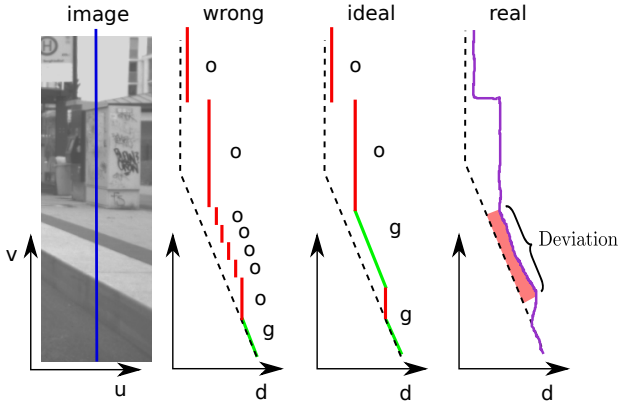


Figure 3.13: The elevation of the ground surface leads to a divergence (marked red) between the expected ground surface and the actual ground measurements. This effect is taken into account additionally by estimating the height of the ground surface for that segment.

for the deviation within the Gaussian part of the data term, such that:

$$P^{g}_{\text{Gaussian}}\left(d_v \mid s_n, v\right) \quad = \quad \frac{1}{\sigma_{\text{g}}\left(f_n, v\right) \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{\left(d_v - f^{g}_{n}(v)\right) - \Delta_{\text{g}}(v)}{\sigma_{\text{g}}(f_n, v)}\right)^2} \tag{3.49}$$

The recomputed result for the street car stop scenario is given in Figure 3.14. The curb is either detected as two adjacent ground segments or as a transition from ground to object and then back to ground for the sidewalk. Which of both applies depends on the local quality of the disparity data. Adjusting the parameters gives room to tune the performance of the algorithm thus allows to detect the curb as a continuously perpendicular object. On the other hand, increasing the sensitivity for the sensor also increases the chance for phantoms in case of errors within the stereo matching. In order to detect curbs properly, more sophisticated schemes have been presented that explicitly exploit the height discrimination of the two-dimensional road and sidewalk surfaces [12, 165, 192].

The second example in Figure 3.15 depicts the labeling result for the urban scenario given in Figure 3.8a. Note the continuous segmentation of the sidewalk on the right hand side. That sidewalk has a height of approximately 20 cm. Except for some minor noise in the data, the remaining ground surface is only segmented for the transition to obstacles such as the cars or the small pylons.

3.8.2 Incorporating 3D Road Course Estimation

Even though the elevation for ground segments is estimated individually, the ground surface itself is approximated using piecewise planar and differently leveled horizontal surfaces.

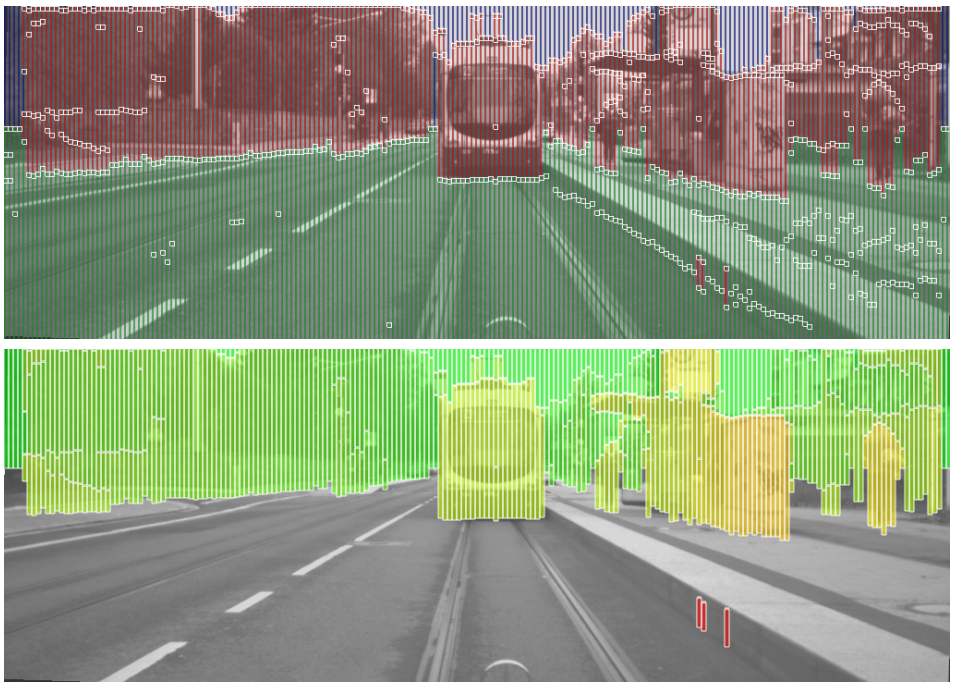


Figure 3.14: The sidewalk of the tram station has a height of approximately 25 cm in comparison to the road surface. The upper image shows the corresponding labels of the segmentation. The curb is continuously segmented from the bottom of the image up to the street car. Thereby it is partially detected as an *obstacle*, depending on the local quality of the stereo reconstruction. This behavior can be adapted by changing the parametrization of the sensor. On the other hand, a higher sensitivity also increases the chance for phantoms in case of strong noise or errors within the stereo depth map.

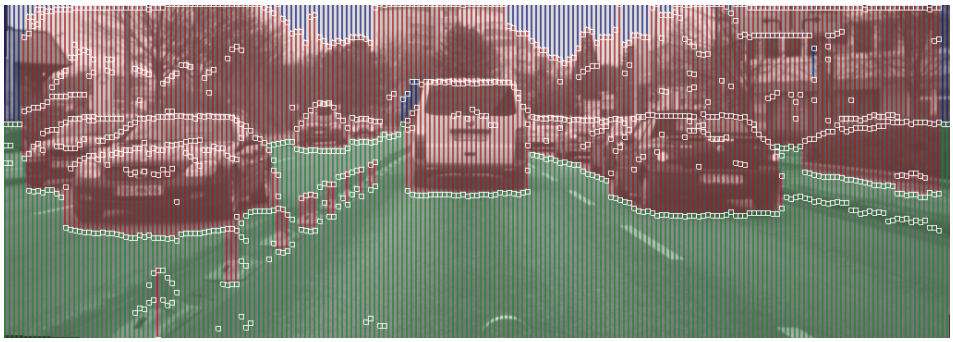


Figure 3.15: This illustrations shows the labeling result for the urban scenario from Figure 3.8a. Using ground level estimation, the approach is capable to detect and segment even slight elevations of the ground surface. This can exemplary be seen on the right side of the image with the sidewalk.

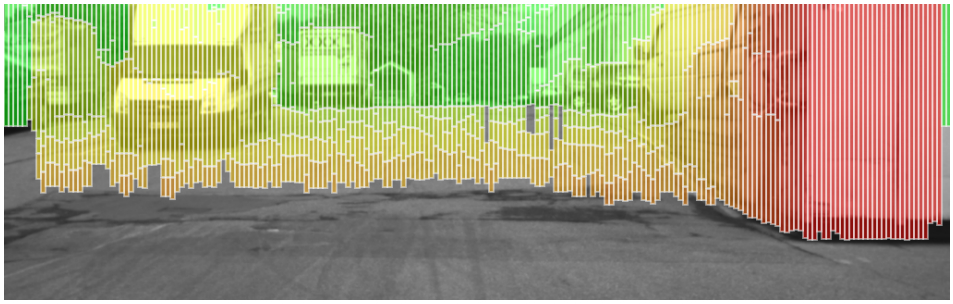


Figure 3.16: The road profile performs a strong ascent right in front of the vehicle. Thus, using piecewise horizontal surfaces to model the road does not suffice to reconstruct the scene properly. As a result, the road is reconstructed using small vertical object segments.

Yet, for a variety of different situations, such as for road ascents at hills or slopes at pits, the road follows a curvature and therefore the assumption of a flat ground surface is not always fulfilled. The resulting deviation between the model and the actual data bears the risk of causing an improper segmentation of the scene. The magnitude of this effect depends on the severity of the model violation and the chosen sensor parameters. An example for such behavior is depicted in Figure 3.16. Due to a strong road ascent of about 1.5 m in height within a 10 m ground segment the reconstruction yields an over-segmentation of the ground surface by means of multiple vertical object segments.

Now, this problem can be encountered in two different ways. The first strategy is to extend the segmentation in order to estimate the gradient of the road course as part of the segmentation process. Even though this approach has the potential to yield good results, the required computational complexity would increase significantly.

Alternatively, the segmentation scheme is assisted by an external road course estimation technique. Concerning this matter, literature provides numerous strategies to estimate the three-dimensional road course in front of the vehicle (e.g. [136, 206, 220]). For our purpose, we choose a variant of the method proposed by Wedel et al. [222] that allows to model the vertical gradient of the ground surface using Bézier curvatures (B-splines). The proceeding is similar to the road level estimation in Section 3.8.1, but allows for a ground surface with a non-planar progression.

The output of the proposed method is interpreted as the ground expectation f_n^g. For that purpose, the obtained spline is projected into disparity space. This proceeding is visualized in Figure 3.17. The final result for the given scenario is depicted in Figure 3.18.

The sketched approach does not contradict with estimating the ground surface elevation, such as for sidewalks. However, the basic assumption is that those surfaces must follow the curvature of the road.

Further note that irrespective of the used road course estimation technique this proceeding underlies certain physical limitations. For instance, even though the road course may be estimated correctly, upon a high gradient of the road surface, the two models in terms of ground and object become very similar and thus are increasingly hard to

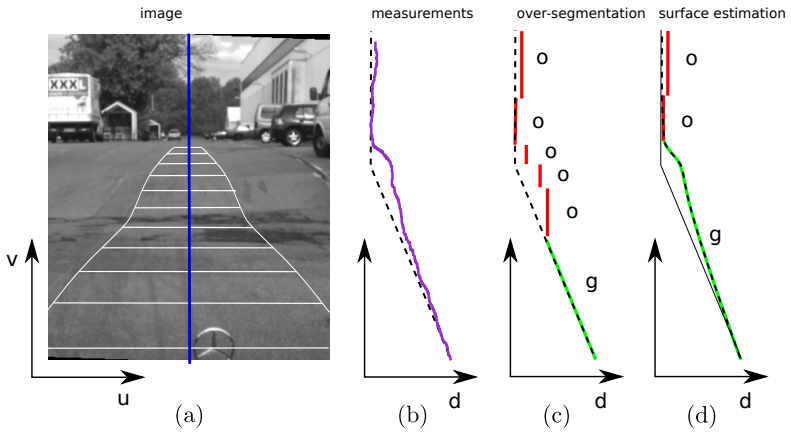


Figure 3.17: The situation with a vertical gradient of the road course is shown in (a). The disparity input for the marked column and the expected ground profile (dashed line) are visualized in (b). Accordingly, this proceeding yields an over-segmentation, such as visualized in (c). To account for this problem, in (d), the estimated spline road-model is mapped to a disparity profile. This profile allows to take the road course into account in a correct way.

distinguish. Furthermore, larger cambers of the road pose an additional challenge. In this case, the view on the falling slope of the camber is often obstructed by the camber itself and thus can not be reconstructed properly.

3.8.3 Processing 3D LIDAR Data

The primary focus of this work is on image-based processing and thus the use of stereo camera systems. However, there are plenty of other sensor types that allow to obtain three-dimensional measurement data, such as time-of-flight cameras or LIDAR.

For this purpose, we discuss how to process 3D point data obtained by using the Velodyne HDL-64E S2 LIDAR [85]. This LIDAR generates 360° point scans of the environment and is well known from its application in the DARPA Urban Challenge contest [141]. It is mounted to the roof of the test vehicle and calibrated relatively to the left camera (see [171] for details). The Velodyne uses 64 vertically aligned laser beams that revolve in 360 degree. The slowest possible spin-rate for the turret is 5 Hz (15 Hz is its maximum), which allows for a maximum horizontal sampling of 10 vertical scans per degree. In comparison, the used stereo camera system yields a resolution of 22 px, so in reference to the camera, the LIDAR reaches about 50% of its horizontal resolution. The most significant feature of this sensor is a measurement accuracy within a few centimeters irrespective of the actual distance to the measured object.

By default, the proposed segmentation scheme is not limited to a certain type of sensor. However, within Section 3.4.1 the importance of the choice for an optimization space with respect to the used 3D sensor was discussed. For segmenting the measurement data obtained with the LIDAR choosing a different coordinate system for optimization

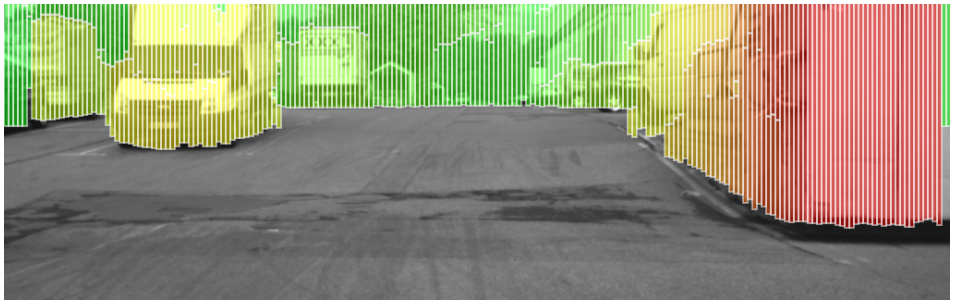


Figure 3.18: Within 10 m the road has an ascent of about 1.5 m in height. Instead of relying on the flat-world assumption, the usage of suitable road course estimators allows to precisely reconstruct even strong gradients of the road course properly (c.f. Figure 3.16).

could prove beneficial. For instance, the use of a cylindrical coordinate space defined by means of two rotational angles in [deg] and distance in [m] might turn out to be a good choice. Therefore, solving the optimization task properly requires to adapt the measurement model accordingly. That on the other hand includes modifications to the definition of the data term and thus involves changes for the data cost pre-computation as well.

With this background and as a proof of concept, a slightly different path is chosen. For our objective, the LIDAR is converted into a virtual stereo sensor. To this end, all 3D point measurements that lie within the field of view of the camera are projected into a virtual depth map. Then, this image is used for the segmentation process. This procedure allows to process the LIDAR data without fundamental or exhaustive changes to the discussed algorithm.

Accordingly, the stereo sensor parametrization (for details see Section 3.4.1) is adapted, such that the measurement characteristic of the Velodyne LIDAR is imitated. The disparity standard deviation $\sigma_d = 0.01$ is chosen considerably small. The term σ_Z used for the model-based normalization (see Equation (3.21) in Section 3.4.1.2) is modified, such that it also takes the standard deviation of the LIDAR into account. Therefore, it computes as

$$\sigma_Z = \frac{f_n^2}{f_u \cdot b} \cdot \sqrt{\Delta_Z^2 + \sigma_L^2} \tag{3.50}$$

Hereby $\sigma_L = 0.1\,\text{m}$ is the assumed standard deviation for the LIDAR measurements and Δ_Z remains unchanged as the expected model violation of the constant depth assumption for objects. The remaining sensor parameters are set accordingly. The outlier rates are set to $p_{\text{out}} = 0.05$ for all models and the class assignment parameter for missing disparities $p_\sharp^c$ is set to neutral with $p_\sharp^g = 1/3$. The latter is required, since the LIDAR outputs spares point clouds. Thus, when using the LIDAR, not having a valid disparity measurement for a pixel is rather the use-case than the exception.

An exemplary result of this approach is shown in Figure 3.19. The upper part of the illustration shows the virtual depth map result. The point measurements have been

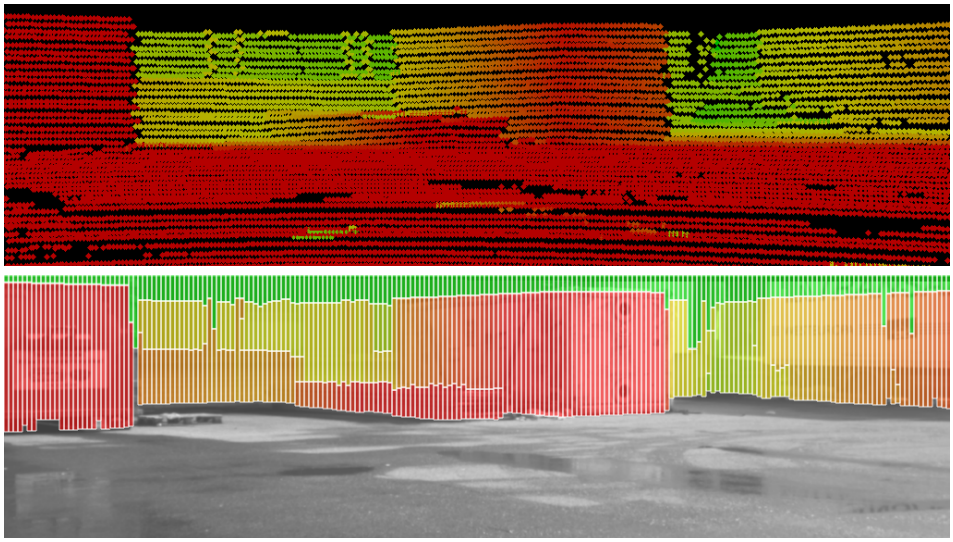


Figure 3.19: The Stixel World computed from sparse 3D point cloud measurements that have been obtained with a Velodyne HDL-64E S2. The upper image shows the projection of the independent LIDAR points (thickened to 5 px for a better visualization) into the left camera. The lower image shows the Stixel segmentation result. In order to be able to work with the LIDAR point measurements, only minor changes to the parametrization had to be applied.

thickened for a better visualization. The lower part shows the corresponding object reconstruction as overlay to the image of the left camera.

Alternatively to using sparse point data, interpolation can be used. Dolson et al. [52] presented a method that allows to obtain dense depth maps from the Velodyne LIDAR even under strong motion of the measurement vehicle.

3.8.4 Incorporating Dense Optical Flow

The scope of the previous sections solely focused on processing 3D data that was obtained from some kind of distance sensor, such as a stereo camera or LIDAR. However, the approach provides much more opportunities. For example, it can be used to combine measurement data taken from multiple sensors. Beyond that, it even allows to combine measurement data of different types. Accordingly, the proposed scheme inherently gives the opportunity to perform multi-sensor data fusion.

In the following, this idea is exemplified by fusing both dense stereo and dense optical flow. The stereo depth map is computed using semi-global matching [73] and the optical flow images are calculated by using the method proposed by Müller et al. [161]. That scheme is based on the variational approach (TV-L1) suggested by Zach et al. [236] and is particularly optimized to cope with the high demands of automotive environments. An exemplary optical flow image is depicted in Figure 3.20. It corresponds to the urban scenario illustrated previously in Figure 3.8a.

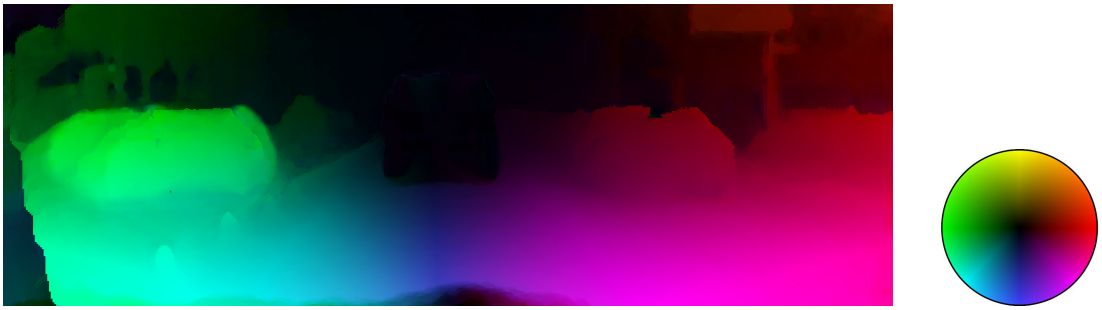


Figure 3.20: A dense optical flow image for the urban scenario from Figure 3.8a is shown. The color denotes the direction of the optical flow vectors while the brightness encodes their length. The leading vehicle is almost completely black. This is correct, since it moves at constant distance and therefore does not change its location or scale within the image.

In order to obtain a unified scheme that allows to incorporate multiple input cues for the segmentation process, the measurement input D is extended to $Z = \{D, U\}$. The variable D corresponds to the stereo depth map computed for the left and right rectified camera images I_k^L and I_k^R, just as before. The measurement input U corresponds to the optical flow image computed for the left camera between the consecutive images I_{k-1}^L and I_k^L. Thus, $Z_u \in Z$ is the corresponding vector that consists of the measurements for the column u. An element $z_v = (d_v, du_v, dv_v) \in Z_u$ is a triplet that consists of the disparity measurement d_v as well as the optical flow component vectors du_v and dv_v for the horizontal and vertical displacements.

The application of the Bayesian rule in Equation (3.5) yields:

$$P(L \mid Z) \sim P(Z \mid L) \cdot P(L) \tag{3.51}$$

It is self-evident that using multiple sources for input is without impact to the prior term $P(L)$, because the world model expectation does not depend on the used input cues. However, the conditional probability $P(Z \mid L)$ has to be extended. In guidance of the argumentation from Section 3.3.2 that discussed the mutual independence of the individual input measurements and the different segmentations of the individual columns, the term $P(Z \mid L)$ can be expressed as:

$$P(Z \mid L) = \prod_{u=0}^{w} P(Z_u \mid L_u) \tag{3.52}$$

$$P(Z_u \mid L_u) = \prod_{n=1}^{N_u} \prod_{v=v_n^b}^{v_n^t} P(z_v \mid s_n, v)$$

In order to evaluate the conditional term $P(z_v \mid s_n, v)$, the affinity of the measurement input z_v at row v for the segment s_n is determined. This term is separated such that a

product of two densities is extracted:

$$P\left(z_v \mid s_n, v\right) = P\left(d_v \mid s_n, v\right) \cdot P\left(du_v, dv_v \mid s_n, v, d_v\right) \tag{3.53}$$

At this point, $P\left(d_v \mid s_n, v\right)$ is identical to the stereo sensor model as discussed in Section 3.4.1. The second term $P\left(du_v, dv_v \mid s_n, v, d_v\right)$ denotes the probability for a pair of optical flow vectors (du_v, dv_v) given the segment s_n in row v with a disparity d_v.

For the definition of that term two things are required. At first, the corresponding sensor model has to be specified that characterizes the used optical flow algorithm in terms of noise and outlier behavior.

Secondly, the corresponding data models for the optical flow occurrences with respect to the different classes *ground*, *object* and *sky* have to be determined. This is challenging, because the occurrence of optical flow is a direct result of the ego-motion and the motion of the observed objects (see [37, 201, 6]). For instance, static ground segments are expected as a gradient directed into the focus of expansion (FOE). The location of the FOE and the actual expected lengths of the optical flow vectors depend on the ego-motion of the vehicle. On the other hand, a particular model for the flow occurrence of objects is hard to determine, since those can move independently from our own vehicle. The assumption that objects move rigidly leads to a uniform occurrence of the horizontal optical flow components du_v. However, for non-rigid objects (such as pedestrians) that model assumption is not met, and objects that reside within the FOE cannot be seen within the optical flow image at all.

At last, *sky* segments are expected to have a homogeneous flow field. Those are expected at an infinite distance, such that their corresponding flow occurrence is only related to the changes of the yaw, pitch and roll angle of the camera system.

In addition, the term $P\left(du_v, dv_v \mid s_n, v, d_v\right)$ has d_v as conditional variable. This allows to consider such fundamental assumptions, such that close objects usually have large flow vectors. If connections of such nature using the disparity d_v are not exploited, the term can be simplified to $P\left(du_v, dv_v \mid s_n, v\right)$.

Alternatively, the term $P\left(z_v \mid s_n, v\right)$ from Equation 3.53 is separated as:

$$P\left(z_v \mid s_n, v\right) = P\left(d_v \mid s_n, v, du_v, dv_v\right) \cdot P\left(du_v, dv_v \mid s_n, v\right) \tag{3.54}$$

Unless the disparity measurements d_v are expected as mutually independent from the optical flow components (du_v, dv_v), that procedure does not allow to directly use the previously defined stereo sensor model. Accordingly, both terms have to be specified.

Also note that in order to use further input cues, the cost pre-computation and the computation of the cost tables have to be adapted accordingly. Another important matter is to ensure that dynamic programming for the optimization step is still applicable (see Section 3.6.2).

4 Stixel Motion Estimation

In order to employ the Stixel World for a wide variety of applications such as in driver assistant tasks and safety systems, an important aspect regarding the represented objects is to have knowledge about their exact motion state. For this reason, this chapter discusses how to obtain an independent motion state estimate for every Stixel. This is achieved by tracking Stixels over time using Kalman filter systems [225]. Those Stixels that have been enriched with velocity information are referred to as *dynamic Stixels*.

The following elaboration is based upon our previous work *"Efficient Representation of Traffic Scenes by Means of Dynamic Stixels"* [169]. In consideration of our fields of application, objects are expected to move earthbound on the ground surface. For this reason, the proposed motion state estimation scheme focuses on the first Stixel along every column of the image. The tracking is done for every Stixel independently.

For this task, different prerequisites are necessary, which are covered within the next sections. Accordingly, we will discuss the design of the Kalman filter system and describe how to obtain the required measurements for the filter update step. For that purpose, multiple approaches to estimate the two-dimensional displacement of Stixels between consecutive images are discussed along with their assets and drawbacks. Finally, experimental results are presented. A detailed analysis regarding the quality and performance of the tracking scheme is found separately in Chapter 5 that focuses on an in-depth evaluation.

4.1 The Filter System

This subsection describes the motion state estimator for the object tracking. For this purpose, we rely on an extended Kalman filter system that estimates the location and the motion state for every Stixel independently. The described filter system is a variant of the point feature tracking approach called *"6D-Vision"* presented by Franke et al. [66, 176].

In the following we will sketch the used ego-motion estimation, discuss the 6D-Vision principle, describe the used filter system for motion estimation and how to obtain the Stixel measurements for the filter update step. Further, different ways to improve the tracking performance using multi-filter systems are discussed.

4.1.1 Ego-motion Estimation

In order to estimate the absolute motion state of other objects, the motion of the ego-vehicle has to be known. Certainly, this information can be taken from inertial sensors. In terms of this work, the use of an image-based ego-motion estimation is preferred, because it outperforms available standard inertial sensors in terms of precision. For this

objective, the literature provides several methods [75, 98, 112, 142]. Among those, the stereo-based method described by Badino et al. [7] is our method of choice.

The basic idea of this approach is to track static image points over time and to use their depth to estimate the motion state of the ego-vehicle with full six degrees of freedom. The optimal three-dimensional translation and rotation between consecutive images is determined in a least square error minimization approach. In order to reduce drift errors, multi-frame estimations of the resulting poses are performed.

4.1.2 The 6D-Vision Principle

Franke et al. presented a method called 6D-Vision [66, 176] that allows for the simultaneous estimation of the three-dimensional position and motion state for a large number of two-dimensional image point features. These features are tracked over time by means of Kalman filters [225]. This procedure results in a rich 6D representation that combines position and velocity information within the state vector $\underline{X} = (X, Y, Z, \dot{X}, \dot{Y}, \dot{Z})^T$. Hereby, X is the lateral, Y the vertical, and Z the longitudinal distance component with respect to the ego-vehicle. The estimation works for every point independently. The underlying motion model is assumed to have a constant velocity with $\dot{v} = a = 0$.

For our purpose, we follow the same principle. However, we do not use point features, but Stixels to estimate the motion states of other road users. Therefore, the tracking approach focuses on the first object to encounter along every column, since those objects are considered the most relevant. Further, objects are expected to move earthbound or at least in parallel to the ground surface. For this reason, the possible degree of freedom of the motion state is restricted with respect to the (upwards pointing) y-axis, such that $\dot{Y} = 0$. That means the Stixels are not expected to move in vertical direction.

As a result, the state vector is reduced to 4D with $\underline{X} = (\underline{x}^T, \underline{v}^T)^T = (X, Z, \dot{X}, \dot{Z})^T$. Hence, only the two-dimensional position and velocity have to be estimated. Note that for environments with severe gradients of the ground surface that approximation results in small errors of the motion state. This is due to not considering the vertical motion component of the objects, which potentially leads to an underestimation of the absolute velocity. However, this error is regarded negligible for our purpose.

4.1.3 The Kalman Filter Design for Dynamic Stixels

In the following section the design of the used Kalman filter system is discussed. Initially, this requires a motion model for the ego-vehicle in order to compensate for the ego motion and to obtain the absolute motion estimate for the tracked objects. For this purpose, a constant velocity v_c and a constant yaw rate $\dot{\psi}_c$ for a given time interval $\Delta t = t_k - t_{k-1}$ between the time steps $k-1$ and k is assumed. Those parameters are determined by using the ego-motion method mentioned above. The movement of the ego vehicle is described in the right-handed coordinate system of the car by:

$$\Delta \underline{x}_c = \int_0^{\Delta t} \underline{v}_c(\tau)\, d\tau = \frac{v_c}{\dot{\psi}_c} \begin{pmatrix} 1 - \cos \dot{\psi}_c \Delta t \\ \sin \dot{\psi}_c \Delta t \end{pmatrix} \tag{4.1}$$

In this coordinate system, every Stixel has a position $\underline{x}_{k-1} = (X, Z)_{k-1}^{T}$ with the corresponding velocity $\underline{v}_{k-1} = (\dot{X}, \dot{Z})_{k-1}^{T}$. Within an elapsed time interval Δt, each Stixel moves to a new corresponding position $\underline{x}_k$. This connection is given by the motion model for the tracked objects. It is defined as:

$$\underline{x}_k = R_y(\psi_c)\left(\underline{x}_{k-1} + \underline{v}_{k-1}\Delta t - \Delta \underline{x}_c\right) \tag{4.2}$$

The term R_y denotes the 2×2 rotational matrix around the y-axis. Next, the system model and the measurement model of the extended Kalman filter are specified. In order to predict the new state $\hat{\underline{X}}_k = (\hat{\underline{x}}^T, \hat{\underline{v}}^T)_k^T = (\hat{X}, \hat{Z}, \hat{\dot{X}}, \hat{\dot{Z}})_k^T$ for a Stixel, the system model is applied to its previous estimate state $\underline{X}_{k-1}$. The discrete system model is defined as:

$$\hat{\underline{X}}_k = A_k \underline{X}_{k-1} + \underline{B}_k v_c + \underline{\omega}_{k-1} \tag{4.3}$$

In this notation, the matrix A_k corresponds to the state transition matrix. The vector $\underline{B}_k$ denotes the control input into the system. The noise term $\underline{\omega}_k$ is assumed to be Gaussian white noise with covariance matrix Q. The state transition model A_k as well as the input vector $\underline{B}_k$ are computed by:

$$A_k = \begin{pmatrix} R_y(\psi_c) & \Delta t R_y(\psi_c) \\ 0_{2\times 2} & R_y(\psi_c) \end{pmatrix} \tag{4.4}$$

$$\underline{B}_k = \frac{1}{\dot{\psi}_c} \begin{pmatrix} 1 - \cos\dot{\psi}_c \Delta t \\ -\sin\dot{\psi}_c \Delta t \\ 0 \\ 0 \end{pmatrix}$$

The tracking algorithm operates on rectified images. This proceeding allows to rely on the pin-hole camera model. Thus, the following non-linear measurement equation is used:

$$\underline{m} = \begin{pmatrix} u \\ v \\ d \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} X f_u \\ Y f_v \\ b f_u \end{pmatrix} + \underline{\gamma} \tag{4.5}$$

with the focal lengths f_u and f_v and the baseline b of the stereo camera system. The noise vector $\underline{\gamma}$ is assumed to be Gaussian white noise with a covariance matrix R. This non-linear projection from the system state to the coordinate space of the measurements makes it mandatory to use an extended Kalman filter. The resulting measurement-state relation for the Kalman filter is obtained by the Jacobian approximation H of Equation (4.5). Further details to this proceeding can be found in [66, 159] or [177].

4.1.4 Obtaining the Stixel Measurements

The measurement update for each Stixel is done in two steps: the update step for the Kalman filter system and the update for the height of the Stixel. For this purpose, the set of new measurements at time step k consists of three parts: an image coordinate

consisting of a column and a row position $(u, v)_k^{\text{meas}}$, the corresponding disparity measurement $d_k^{\text{meas}}$and a height observation h_k^{meas}. Note that the height is not a direct part of the filter system.

The area covered by the Stixel at the previous time step is denoted by Ω_{k-1}. It is linked to the previously measured image position $(u, v)_{k-1}^{\text{meas}}$. It is important to note that this is an actually tracked coordinate and not a filtered one. The new image coordinate measurements $(u, v)_k^{\text{meas}}$ are determined by computing the transition from Ω_{k-1} to Ω_k. Hence, a method to measure the two-dimensional displacement between two consecutive images I_{k-1} and I_k is required. For this purpose, the choice for a particular scheme is not substantial. Possible methods are discussed along with their advantages and disadvantages in Section 4.2. Their particular influence to the performance of the estimator is evaluated within Section 5.3.

In the following, $\Delta(u, v)$ denotes the corresponding translation for the Stixel area Ω_{k-1} to Ω_k. Consequently, the new image coordinate measurements $(u, v)_k^{\text{meas}}$ are computed by:

$$(u, v)_k^{\text{meas}} = (u, v)_{k-1}^{\text{meas}} + \Delta(u, v) \tag{4.6}$$

Secondly, the disparity measurement d_k^{meas} is extracted from the current stereo depth map D_k by using the obtained area Ω_k as region of interest (ROI). This is achieved by means of the function $\mathrm{d}(D_k, \Omega_k)$, such that:

$$d_k^{\text{meas}} = \mathrm{d}\left(D_k, \Omega_k\right) \tag{4.7}$$

In a straightforward approach, that function could simply compute the average disparity of all depth measurements within Ω_k. However, since D_k is a dense disparity map, it is reasonable to make use of the available amount of data and to rely on a scheme that in addition to an increased accuracy of the disparity measurement offers robustness to outliers. For our implementation, this is achieved by using a histogram-based registration scheme. All disparity measurements $d_{u,v} \in D_k(\Omega_k)$ are registered in a histogram while taking into account their assumed stereo uncertainty σ_d. For that step the disparity measurements are modeled to be Gaussian distributed. Then, the most frequent value is extracted as the new disparity measurement d_k^{meas}. The histograms are using eight bins per pixel to allow sub-pixel accuracy.

Note that at this point a minor aspect is disregarded during the computation of the displacement $\Delta(u, v)$ and the extraction of the new depth measurement d_k^{meas}. This concerns possible scaling effects when deducing Ω_k from Ω_{k-1}, which is either a result of our own motion or the longitudinal movement $\dot{Z}$ of the Stixel itself. Due to the rather small changes between consecutive images, those effects are regarded as insignificant for the disparity measurement computation. In either way, the area Ω_k is always centered on the new tracked image position of the Stixel.

However, if it is desired to consider changes in scale, the method to determine the displacement $\Delta(u, v)$ discussed in Section 4.2.6 can be applied, since it allows to estimate that degree of freedom as well.

The third measurement in order to update the Stixel is the corresponding height measurement h_k^{meas}. Since objects are assumed to be rigid, the height of each Stixel within world coordinates should remain constant. Nevertheless, when approaching an object, the quality of the height measurement for a Stixel representing that object tends to increase. Thus, additionally updating the height of that Stixel is advisable. Since dynamic Stixels are not re-computed for every frame explicitly, this height measurement has to be determined in a different way.

For this purpose, after estimating the new state $\underline{x}_k$, the closest static Stixel is determined and its height is adopted. That approach is straightforward, yields good results and minimizes the computational overhead, since no additional height computation is required. The height information is updated by means of a low pass filter:

$$\begin{aligned} h_k^{\text{meas}} &= h_{\text{static}} \\ h_k &= \alpha \cdot h_{k-1} + (1-\alpha) \cdot h_k^{\text{meas}} \end{aligned} \tag{4.8}$$

For this approach, $\alpha \approx 0.95$ is chosen as a fixed and rather insensitive parameter.

4.1.5 The Multi-Filter System

Traffic environments typically exhibit a very large variety of independently moving objects. The dynamics of these scenarios ranges from urban districts with slow, but multi-directional traffic to highway scenarios with very fast, but one-way traffic. Within all these scenarios, the motion state of the ego-vehicle is not necessarily related to the motion state of the other objects. That issue makes it very challenging to build a system that is capable to estimate the motion state for other objects reliably and under all conditions.

Normally, upon initialization of a new dynamic Stixel, the motion state of the tracked object is yet unknown. Hence, the goal is to have the filter to adapt as fast as possible. On the other hand, the filter system must not be disturbed easily upon noisy measurements, such that it actually has a true filtering effect.

A common strategy to tackle that problem is to rely on multi-filter systems. These use a certain number of filters to track one and the same object. Typically, the difference lies within the model parameters used for the individual filters. For instance, they can differ with respect to the agility of the filter system in order to adapt to changes of the observed motion state differently (e.g. see [66]). Alternatively, they are used to consider different types of motion or motion models, such as moving straight versus turning maneuvers (e.g. see [19]).

For our purpose, we rely on a multi-filter systems that uses two filters, a dynamic one that is optimized to detect fast movement and a delayed one to work well for static objects. Thus, the used filters differ in regard to the parametrization of the modeled system variances.

Optionally, instead of using a single dynamic filter, additional pre-initialized filters with certain hypothetical initial velocities can be used. Those approaches aim for the effect that the pre-initialization coincides closely with the actual motion state of the object. However, due to the wide variety that different types of traffic scenes exhibit, that approach has not worked well for our purpose.

Another common and straightforward idea to optimize the transient response of the filter is to exploit already existing Stixels for a neighborhood initialization. To do so, qualified neighbors have to fulfill certain fitness criteria, such as a minimum age and a maximum residual error of the filter. When no qualified neighbors are available, the tracker falls back into the normal mode of operation.

From the set of available Kalman-filters the currently active one is determined by the smallest normalized innovation squared (NIS) [13, 66]. The NIS corresponds to the Mahalanobis distance between the predicted and the actual measurements using the inverse covariance matrix $\sum_k^{-1}$ of the innovation (residuals). Thus, it is defined as

$$\varepsilon_{\text{NIS}} = (\underline{m}_k - H\hat{\underline{x}}_k)^T \cdot \sum\nolimits_k^{-1} \cdot (\underline{m}_k - H\hat{\underline{x}}_k) \ , \tag{4.9}$$

where $\underline{m}_k = (u, v, d)_k^{\text{meas}}$ is the actual measurement and $\hat{\underline{x}}_k$ is the predicted state that is projected to the measurement space by the Jacobian approximate H of the measurement model (see Equation(4.5)). Finally, in order to optimize the computational effort, the multi-filter system is reduced to a single filter after a certain number of update steps. For this purpose, the filter with the smallest NIS is chosen to remain.

Alternatively to deactivating filters, interacting multiple models (IMM) can be used, such as introduced by Blom et al. [28] or Bar-Shalom et al. [13, 145]. The idea of IMM is straightforward. All filters are maintained during the tracking process, and the one with the highest likelihood is the one used to proclaim its estimate state for the tracked object. For instance, IMM found application in the work of Kaempchen et al. [109] or Barth et al. [17, 19]. A combination of both NIS and IMM was shown by Lategahn et al. [124].

4.2 Measuring 2D Displacements Between Consecutive Time Steps

The most essential part for the tracking process is to compute the translation of the Stixel area Ω_{k-1} to Ω_k, which is a typical two-dimensional optical flow correspondence problem. Within the scope of that task, several optical flow estimators have been tested. This includes the signature-based method proposed by Stein et al. [196], the TV-L1 dense optical flow as proposed by Müller et al. [161], the well known KLT-Tracker presented first by Lucas et al. [137] and an extension of that approach that we refer to as *patch-KLT*.

4.2.1 Choosing an Optical Flow Scheme

For our objective, these mentioned optical flow estimators can be classified with respect to three essential criteria. At first, some of them are feature-based (and thus sparse) while others allow for a dense computation, such that almost every pixel has a corresponding optical flow estimate. Naturally, having more optical flow measurements increases the chance to be able to compute Ω_k from Ω_{k-1} more reliably. However, the performance of optical flow methods differs severely when utilized for such challenging and complex

scenarios as traffic environments. Those usually exhibit a very high variety of lighting conditions and partially require to estimate relatively large optical flow vectors.

Thus secondly and more important for our application is the support of the used optical flow estimator to allow for pre-warping. That means that the actual optical flow scheme can be pre-initialized on a feature-based level. If supported, that is a very useful feature, because it gives the opportunity to use the motion filter prediction to aid the optical flow computation. As a result, the performance with regard to large optical flow vectors can be increased significantly.

Thirdly, the used methods strongly vary in terms of the computational effort that is required to determine the 2D motion field. Some of the proposed methods are available on low-power FPGA hardware, some occupy the CPU and others rely on a GPU implementation.

The individual methods and their specific characteristics for the computation of Ω_k are described in the following.

4.2.2 Computing the 2D Displacement Using Census Optical Flow

Stein et al. [196] presented a method that allows to compute optical flow using the Census transform [231] as matching criteria. The proposed scheme computes sparse and pixel discrete optical flow vectors. It is available in real-time by using a low-power FPGA hardware implementation. The authors rely on a hash-table registration scheme for the patch descriptors which allows to decouple the computation time from the maximum supported vector length. As a result, the most significant asset of that approach is the support for arbitrarily large flow vectors. The Census itself is known to be quite robust to changes of the brightness between consecutive frames [89, 241].

In order to compute the new location of the area Ω_k, the optical flow measurements within Ω_{k-1} are accumulated within histograms. These do not need to offer sub-pixel resolution, because the proposed scheme itself computes only pixel-discrete optical flow vectors. Optionally, when all measurements are registered, the histograms are smoothed using a Gaussian kernel mask. The most frequent value is extracted and a parabolic refinement is used to obtain sub-pixel accuracy such that

$$\begin{aligned} px_{\text{corr}} &= \frac{h_R - h_L}{2 \cdot (2 \cdot h_C - h_L - h_R)} \\ idx_C &= idx_C + px_{\text{corr}} \end{aligned} \tag{4.10}$$

In this notation h_C denotes the most frequent value located at the index idx_C within the histogram. The terms h_L and h_R are the corresponding neighbors of h_C. The term px_{corr} is the sub-pixel correction with $px_{\text{corr}} \in [-0.5, 0.5]$.

4.2.3 Computing the 2D Displacement Using Dense TV-L1 Optical Flow

Typically, large displacement vectors and changes of brightness (globally and locally) pose a big challenge for approaches based on total variation error minimization [197]. In order

to compensate for that, Müller et al. [161] have proposed a method that puts dedicated focus on the application in open road scenarios. Therefore, it allows to incorporate additional knowledge about the three-dimensional scenario. Their scheme is a variant of the work proposed by Zach et al. [236] which on the other hand is based on the variational approach proposed by Horn and Schunk [97].

In order to stabilize their method, the authors incorporate depth from stereo and odometry information in order to compute an expected optical flow field for static image points. Further, moving objects are considered by means of the previously discussed Census optical flow variant. The obtained information is used as an additional regularization prior during the TV-L1 optical flow computation. Their provided CUDA implementation is real-time capable for high-end GPUs. For further detail we refer to the corresponding work of the authors [161]. An example of their approach was shown in Figure 3.20.

In order to use that method to compute the new Stixel area Ω_k, the same approach as described in subsection 4.2.2 is used. A slight difference is that the TV-L1 approach is able to yield sub-pixel accuracy. That is taken into account by using histograms with a higher resolution. Good results have been achieved by using 8 bins per pixel. Optionally, the obtained precision allows to waive the parabolic fitting step.

4.2.4 Computing the 2D Displacement Using the KLT Method

A well-known method to compute tracks for sparse point features has been proposed by Lucas et al. [137]. That scheme relies on the differential Lucas-Kanade method. The actual displacement is computed by solving the optical flow equations for all the pixels in the neighborhood of the center point. This is achieved by means of a least squares error minimization. With the objective to gain robustness to global illumination changes, the scheme was extended with respect to the matching criteria, such as the zero-mean sum of squared differences (ZSSD) or the zero-mean sum of absolute differences (ZSAD) [184, 89]. Large displacements are achieved by computing the algorithm at multiple scales of the input images.

In order to use the KLT method to compute the displacement $\Delta(u, v)$ for a given Stixel, the goal is to spread reliable point features across its area Ω_{k-1} for tracking. For this purpose, this area is first sub-divided into M adjacent regions. Then, possible features within these regions are selected using the Harris corner detector [84]. The quality of each feature is determined by computing the eigenvalue for its neighborhood patch [213, 214]. Then, a certain number N/M with the highest eigenvalues is selected from each interval for the tracking step.

The major benefit of the KLT method is to be able to pre-initialize the optical flow vectors independently for every dynamic Stixel. That initialization is computed from the prediction of the Kalman filter. For every feature point $(u, v)_n$ with $1 \leq n \leq N$ the corresponding predicted image coordinate is computed as:

$$\underline{\hat{m}}_n = (\hat{u}, \hat{v})_n^T = (u, v)_n + \underbrace{\left(H\underline{\hat{x}}_{k,n} - H\underline{x}_{k-1,n}\right)|_{u,v}}_{\hat{\Delta}(u,v)_n} \tag{4.11}$$

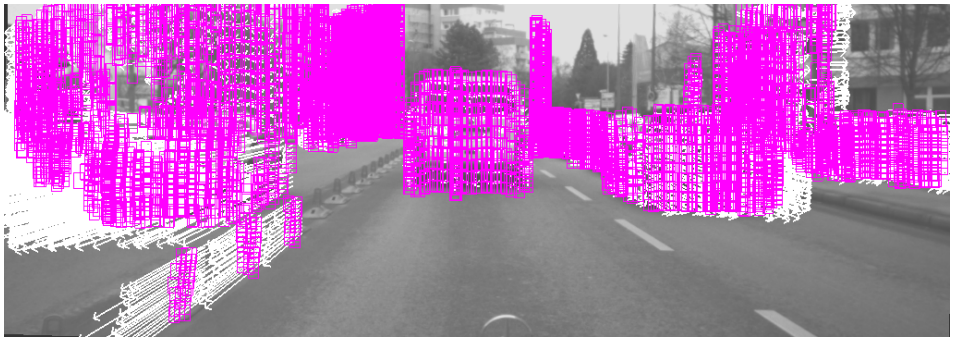


Figure 4.1: The pink rectangles denote the $w \times h$ sized patches that are used for computing the two-dimensional displacement into the next time step. The white arrows show the estimated direction into which each particular feature was tracked.

The term $\underline{x}_{k-1,n}$ is the last Stixel location estimate and $\hat{\underline{x}}_{k,n}$ is the corresponding prediction for the current time step. These coordinates are extended for the world height h_n of the particular point feature. This is because $\underline{x}_{k-1}$ just denotes the two-dimensional XZ-location of the Stixel's base point. The matrix H is the Jacobian approximation of the measurement equation. The resulting term is projected to image coordinates (u, v) which drops the unused disparity component.

The proposed scheme is real-time capable on both the CPU and any mainstream GPU using a CUDA implementation. For our experiments, every Stixel is sub-divided into regions of 25 px in height. Up to $M = 3$ features are tracked per region, such that an exemplary object with a height of 150 px is tracked with up to $N = 18$ features. The KLT-feature size is 5×5. The tracks are regenerated for every new input image.

4.2.5 Computing the 2D Displacement Using the Patch-KLT Method

At last, the patch-KLT method is discussed. The basic motivation for that scheme is the request to make a more thorough use of the actual texture information within the area of the Stixels. The proposed method is an extension of the previously discussed KLT feature tracker with the objective to use real $w \times h$ sized areas as features.

Since the underlying Stixel assumption to have objects with rigid and flat shapes is not met by all types of objects, using the Stixel as a single patch to track does not yield satisfying results. For this reason, the Stixel is sub-divided vertically into equally distributed and partially overlapping patches that are individually tracked as a complete patch between the consecutive frames. The working principle of the patch-KLT is visualized in Figure 4.1.

Additionally, since working with larger patches also leads to a few special characteristics, three further aspects are considered. Firstly, in order to gain robustness, possible scaling effects are taken into account. Thus, if an object approaches, the corresponding patches grow according to the changes of depth of that object and vice versa. The scale estimation is a subcategory of the affine transformations as introduced by Tomasi et

al. [214] or by Birchfeld et al. [24]. However, in contrast to that work, we do not estimate distortion or similar perspective effects.

Secondly, a pixel-based voting scheme is introduced. That procedure has the objective to concentrate on foreground pixels. This is achieved by balancing the influence of the different disparity measurements depending on how good they coincide with the depth of the Stixel. For instance, this method is well suited to reduce the influence of grass beneath guard rails. Usually, those rails exhibit very high ambiguity due to repetitive patterns along the road. This characteristic makes them very hard to track. Vegetation on the other hand is well-textured and good to track but does not belong to the foreground, consequently has a different depth and thus a different optical flow than the actual tracked object.

Thirdly, for the matching criteria we rely on the well-known zero-mean sum of squared differences (ZSSD) [89, 184]. This has the objective to be more insensitive to changes of illumination across the area of the patch.

For improving the performance with respect to large displacement vectors, the approximate location of the patches within the current frame is initialized using the Kalman filter prediction of the corresponding Stixels. For this purpose, the method described in Equation (4.11) is used.

Technical details on the patch-KLT procedure are given in the following Section 4.2.6. An exhaustive survey on KLT flow estimators can be found in [143].

For our experiments, the KLT-patches are chosen to have a width of 25 px (which is significantly larger than the 5 px width of the Stixel) and the height is set to 15 px with a vertical overlap of 20%. The proposed method is computationally very expensive. Even though we rely on a considerably fast CUDA implementation, the scheme requires an average of 150 ms to compute on high-end graphics hardware. Thus, it is not fast enough to be tested online in the vehicle. However, given its theoretical and technical advance to the other optical flow schemes, this method is expected to obtain very good results when used for offline processing.

4.2.6 Working Principle of Patch KLT

In the following, the previously introduced variation of the Kanade-Lucas-Tomasi tracker is described in technical detail. It estimates the two-dimensional displacement as well as the change of scale for $w \times h$ sized patches. Also, it uses a weight mask obtained from the underlying disparities to emphasis the tracking on foreground pixels.

At first, the mathematical background with regard to the relation of scale and depth is discussed. Assume to have a Stixel at time step $k-1$ with the center world coordinates in the camera coordinate system $X_{k-1}, Y_{k-1}, Z_{k-1}$ and the width and height W_{k-1}, H_{k-1}. The Stixel is idealized as a fronto-parallel area, and therefore has the same depth in reference to the camera for all underlying pixels.

Its center is projected to image coordinates (u_{k-1}, v_{k-1}) by using the projection equations for the pinhole camera geometry denoted in Equation (4.5). The components of

the image coordinates compute as follows:

$$u_{k-1} = \frac{X_{k-1}}{Z_{k-1}} \cdot f_u + u_p \tag{4.12}$$
$$v_{k-1} = \frac{Y_{k-1}}{Z_{k-1}} \cdot f_v + v_p$$

The variables u_p and v_p correspond to the principal points of the camera. The dimensions of the Stixel in the image are given by:

$$w_{k-1} = \frac{W_{k-1}}{Z_{k-1}} \cdot f_u \tag{4.13}$$
$$h_{k-1} = \frac{H_{k-1}}{Z_{k-1}} \cdot f_v$$

Since the represented objects are assumed to be rigid, the world dimensions of the Stixels do not change over time. Consequently, that characteristic also implies

$$W_{k-1} = W_k \tag{4.14}$$
$$H_{k-1} = H_k \ ,$$

such that the dimensions of the Stixel within in the image at time step k are given by:

$$w_k = \frac{W_{k-1}}{Z_k} \cdot f_u \tag{4.15}$$
$$h_k = \frac{H_{k-1}}{Z_k} \cdot f_v$$

As a result, the change of depth is indirectly proportional to the change of the dimensions in the image. That relation is given by the scale factor s_{k-1}. That term computes as:

$$\frac{Z_{k-1}}{Z_k} = \frac{w_k}{w_{k-1}} = \frac{h_k}{h_{k-1}} = s_{k-1} = 1 + \Delta s_{k-1} \tag{4.16}$$

Firstly, the KLT-based tracker with scale estimation for $w \times h$ sized patches is described. The per-pixel weighting and the zero-mean aspect are incorporated afterwards.

A wide-spread approach to solve optical flow estimation problems is to proclaim the constant brightness assumption [97, 213, 236]. This assumption states that the same three-dimensional world point at different time steps $k-1$ and k and different locations within the camera's coordinate system may lead to a projection into the image at different positions but with the same intensity value for the affected pixels. This principle is applied to the area of the whole patch with u_c, v_c as the coordinates of the feature's center. The consideration of every pixel $u \in \left[-\frac{w}{2}, \frac{w}{2}\right], v \in \left[-\frac{h}{2}, \frac{h}{2}\right]$ within that patch leads to the following over-determined system of equations:

$$I_{k-1}(u_c + u, v_c + v) = I_k(u_c + s_{k-1} \cdot u + \Delta u, v_c + s_{k-1} \cdot v + \Delta v) \quad \forall u, v \tag{4.17}$$

The terms Δu and Δv represent the horizontal and vertical displacement components of the considered patch. To maintain a better readability, in the following we use s for s_{k-1} and Δs for Δs_{k-1}. Hence, by solving the given equation system the unknown parameters $(\Delta u, \Delta v, \Delta s)$ are determined. Unfortunately, the right side contains the unknowns as a functional dependency in order to access the gray value of the image. To resolve that dependency, a first order Taylor expansion is applied to that term, which results in:

$$\begin{aligned} & I_k\left(u_c + (1+\Delta s)\cdot u + \Delta u, v_c + (1+\Delta s)\cdot v + \Delta v\right) \\ \approx & I_k\left(u_c + u, v_c + v\right) + \nabla I_{k_{u_c+u,v_c+v}} \Delta s \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \nabla I_{k_{u_c+u,v_c+v}} \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} \\ = & I_k\left(u_c + u, v_c + v\right) + \frac{\partial I_k}{\partial u}\left[\Delta s \cdot u + \Delta u\right] + \frac{\partial I_k}{\partial v}\left[\Delta s \cdot v + \Delta v\right] \end{aligned} \tag{4.18}$$

At this point, a closed form solution of that problem cannot be found directly for two basic reasons. Firstly, the Taylor expansion performs a linearization, which is only an assumption of the linear characteristic to apply for the given image pair. Secondly, the constant brightness assumption is a rough approximation as well. That is due to various physical reasons, such as sensor noise, certain lens effects like vignetting, changes of the environmental lighting conditions or alike. However, that problem is encountered by stating it as an error minimization problem. For this purpose, the sum of squared differences between the pixel intensities of I_k and I_{k-1} is minimized. This is achieved by solving:

$$\min \sum_u \sum_v \left(I_k(u_c + (1+\Delta s)\cdot u + \Delta u, v_c + (1+\Delta s)\cdot v + \Delta v) - I_{k-1}(u_c + u, v_c + v)\right)^2 \tag{4.19}$$

With the first order Taylor expansion given in Equation (4.18), the solution is found by solving the least squares problem. The resulting equation system can be expressed as:

$$Ax = b \tag{4.20}$$

$$\begin{bmatrix} \frac{\partial I_k}{\partial u} & \frac{\partial I_k}{\partial v} & \frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{pmatrix} \Delta u \\ \Delta v \\ \Delta s \end{pmatrix} = \begin{bmatrix} I_k(u_c + u, v_c + v) - I_{k-1}(u_c + u, v_c + v) \\ \vdots \end{bmatrix}$$

The optimum solution for that problem is found by means of the Gauss–Markov theorem. For this purpose, the following expression is computed:

$$x = \left(A^\top A\right)^{-1} A^\top b \tag{4.21}$$

The first part is defined as:

$$A^\top A = \tag{4.22}$$
$$\begin{bmatrix} \sum \frac{\partial I_k}{\partial u}\frac{\partial I_k}{\partial u} & \sum \frac{\partial I_k}{\partial u}\frac{\partial I_k}{\partial v} & \sum \frac{\partial I_k}{\partial u}\left(\frac{\partial I_k}{\partial u}x + \frac{\partial I_k}{\partial v}y\right) \\ \sum \frac{\partial I_k}{\partial v}\frac{\partial I_k}{\partial u} & \sum \frac{\partial I_k}{\partial v}\frac{\partial I_k}{\partial v} & \sum \frac{\partial I_k}{\partial v}\left(\frac{\partial I_k}{\partial u}x + \frac{\partial I_k}{\partial v}y\right) \\ \sum \frac{\partial I_k}{\partial u}\left(\frac{\partial I_k}{\partial u}xu + \frac{\partial I_k}{\partial v}v\right) & \sum \frac{\partial I_k}{\partial v}\left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right) & \sum \left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right)^2 \end{bmatrix}$$

and the second part as:

$$A^\top b = \begin{bmatrix} \sum \frac{\partial I_k}{\partial u}\left(I_k(u_c+u, v_c+v) - I_{k-1}(u_c+u, v_c+v)\right) \\ \sum \frac{\partial I_k}{\partial v}\left(I_k(u_c+u, v_c+v) - I_{k-1}(u_c+u, v_c+v)\right) \\ \sum \left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right)\left(I_k(u_c+u, v_c+v) - I_{k-1}(u_c+u, v_c+v)\right) \end{bmatrix} \tag{4.23}$$

Next, the individual weighting of the pixels is considered. Therefore, the last disparity measurement d_{k-1}^{meas} of a Stixel is used as reference for all assigned patches. An individual weight $w(u, v)$ is computed for every pixel (u, v) within a patch depending on the similarity of d_{k-1}^{meas} to the disparity $D_{k-1}(u, v)$. Since every pixel can have a different disparity, the corresponding weights may differ as well. The weight mask can be computed in advance to the optical flow computation. For computing the weights we obtained good results using:

$$w\left(u, v\right) = \frac{1}{\left|D\left(u, v\right) - d_{k-1}^{\text{meas}}\right| + 1} \tag{4.24}$$

In the following, each equation of the least squares system is multiplied with the corresponding weight factor $w = w(u, v)$ for the pixel (u, v). The resulting components are:

$$A^\top b = \tag{4.25}$$
$$\begin{bmatrix} \sum w \cdot \frac{\partial I_k}{\partial u}\frac{\partial I_k}{\partial u} & \sum w \cdot \frac{\partial I_k}{\partial u}\frac{\partial I_k}{\partial v} & \sum w \cdot \frac{\partial I_k}{\partial u}\left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right) \\ \sum w \cdot \frac{\partial I_k}{\partial v}\frac{\partial I_k}{\partial u} & \sum w \cdot \frac{\partial I_k}{\partial v}\frac{\partial I_k}{\partial v} & \sum w \cdot \frac{\partial I_k}{\partial v}\left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right) \\ \sum w \cdot \frac{\partial I_k}{\partial u}\left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right) & \sum w \cdot \frac{\partial I_k}{\partial v}\left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right) & \sum w \cdot \left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right)^2 \end{bmatrix}$$

and

$$A^\top b = \tag{4.26}$$
$$\begin{bmatrix} \sum w \cdot \frac{\partial I_k}{\partial u} \cdot \left(I_k(u_c+u, v_c+v) - I_{k-1}(u_c+u, v_c+v)\right) \\ \sum w \cdot \frac{\partial I_k}{\partial v} \cdot \left(I_k(u_c+u, v_c+v) - I_{k-1}(u_c+u, v_c+v)\right) \\ \sum w \cdot \left(\frac{\partial I_k}{\partial u}u + \frac{\partial I_k}{\partial v}v\right) \cdot \left(I_k(u_c+u, v_c+v) - I_{k-1}(u_c+u, v_c+v)\right) \end{bmatrix}$$

To gain robustness against illumination changes, it is advisable to replace the direct access to the gray values by the difference between the gray value and the mean gray value calculated over the feature patch. So instead of the difference

$$\Delta I(u, v) = I_k(u_c+u, v_c+v) - I_{k-1}(u_c+u, v_c+v) \tag{4.27}$$

that was used up to this point, the following difference is used:

$$\Delta I(u,v) = \left(I_k(u_c+u, v_c+v) - \bar{I}_k\right) - \left(I_{t-1}(u_c+u, v_c+v) - \bar{I}_{k-1}\right) \tag{4.28}$$

Hereby the terms $\bar{I}_k$ and $\bar{I}_{k-1}$ denote the mean gray values. Those are computed by using the weighted mean of the pixel intensities, such that:

$$\begin{aligned} \bar{I}_k &= \frac{\sum w(u,v) \cdot I_k(u_c+u, v_c+v)}{\sum w(x,y)} \\ \bar{I}_{k-1} &= \frac{\sum w(u,v) \cdot I_{k-1}(u_c+u, v_c+v)}{\sum w(u,v)} \end{aligned} \tag{4.29}$$

The term $A^T A$ remains unchanged and $A^\top b$ becomes:

$$A^\top b = \begin{bmatrix} \sum w(u,v) \cdot \frac{\partial I_k}{\partial u} \Delta I(u,v) \\ \sum w(u,v) \cdot \frac{\partial I_k}{\partial v} \Delta I(u,v) \\ \sum w(u,v) \left(\frac{\partial I_k}{\partial u} x + \frac{\partial I_k}{\partial v} y\right) \Delta I(u,v) \end{bmatrix} \tag{4.30}$$

As a result of the linearization, the proposed method requires iterations in order to compute the displacement and scale for the patch. To this end, it is advisable to use a pyramid with different scales of the camera input images in order to allow for larger displacements [137]. For our tests, we have used up to three pyramid levels, each with a scale down factor of two in each dimension. Additionally, using different scales can help to improve the convergence speed.

4.3 Tracking Results

In the following section we present experimental results for the Stixel motion estimation as discussed in the previous sections. For this purpose, all sequences have been captured with our test vehicle. Since the described approach is considered supplemental to our work presented previously, the result section for the motion estimation is kept rather short. Additional results can also be found in the original publication for the dynamic Stixel World [169]. Even though the initial paper did rely on the Stixel World as proposed in [9], the tracking scheme does not imply a specific Stixel computation principle. An in-depth analysis with regard to the tracking performance of the different flow schemes is done in the evaluation in Chapter 5.

For the visualization of direction and speed we rely on the color scheme that is depicted in Figure 4.2. Each color type represents a different moving direction within the XZ-plane and the particular saturation denotes the absolute velocity for every Stixel. White in that context means that the Stixel is estimated as not moving and maximum saturation is reached with $30\,\mathrm{km/h}$ or respectively $\sim 8.3\,\mathrm{m/s}$. In addition, the moving direction and speed are also encoded by arrows drawn on the ground plane. Longer arrows represent higher velocities. The arrows point to that spot where the corresponding Stixel is expected to be within the next half second under the assumption of constant speed and linear motion.

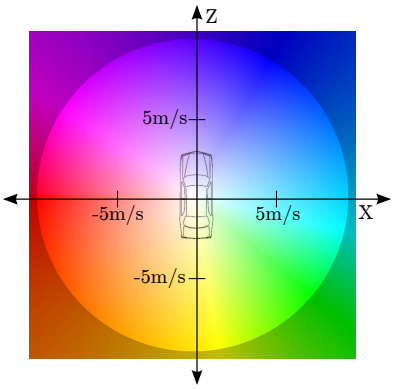


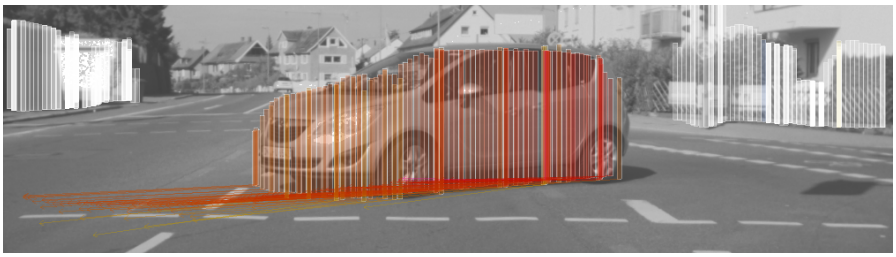
Figure 4.2: This color wheel is used for encoding the motion of Stixels. The color type denotes the moving direction. The absolute velocity is given by the saturation of the color. White in this context means that the Stixel is standing still.

For urban districts, Figure 4.3 illustrates several typical situations to encounter. The examples show other traffic as captured from our own moving vehicle. Those are detected to move in all kinds of different directions. A brief description for each scenario is given beneath the corresponding figure.

The next example is given in Figure 4.4. It shows a car that performs a sharp turning maneuver with 180 degrees right in front of our own vehicle. Using the motion coloring scheme, this figure vividly illustrates the different stages of that maneuver. The car enters the scene from left to right, turns into our direction and finally exits our field of view to the left.

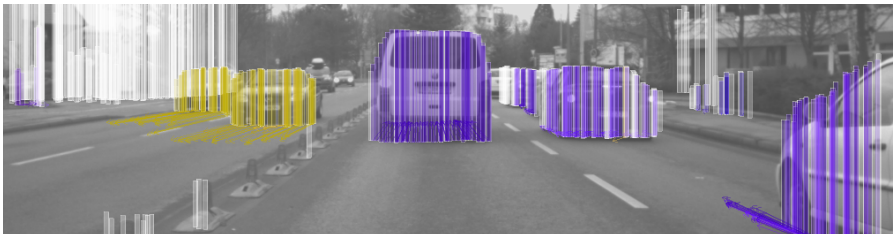
Furthermore, Figure 4.5 depicts scenarios that pose a great challenge for the motion state estimation. A common problem is that the frame rate is not sufficient to reliably resolve the optical flow correspondences for repetitive patterns as a result of the driven speeds. Consequently, the quality of the optical flow estimate suffers severely and hence the quality of the estimated motion state does too. Other structures and areas of the image are barely textured at all which makes the optical flow computation potentially ambiguous. This effect is amplified by weak scene illumination levels. Those lead to low contrasts and motion blur due to long exposure times. In literature, these effects are commonly referred to as the aperture problem or as the blank wall problem [39, 139, 212]. Except for the Census optical flow scheme, the described effects increase the computational effort strikingly. This is due to the gradient-descent based methods that require significantly more iterations for convergence (cf. Section 4.2). As soon as the demanding computation results in additionally skipped (and thus non-computed) frames, the quality of the tracking process declines rapidly.

At last, an example for non-rigid motion is provided. That issue is exemplified in Figure 4.5c by means of pedestrians that walk across the road at an intersection. The coarse motion states for the center of the persons are estimated correctly. But as expected, the overall quality suffers due to the inherent model violations, when the rigid Stixel principle is applied to the flexible body shape. For that objective, either point-based

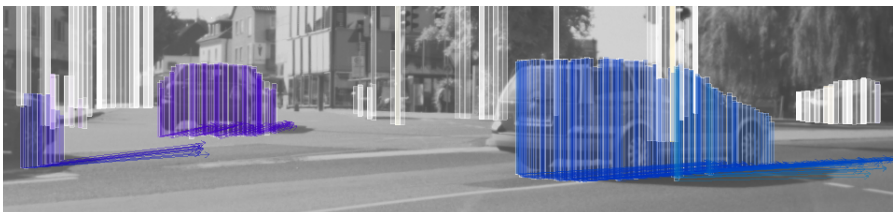


(a) A car is taking a turn at an intersection while passing from right to left. Its rotation can be seen as the arrows on the rear still point to the left while the arrows attached to the front already point further into our opposing lane.

(b) Two cars and a scooter are taking a left turn to our right. The vehicle in the middle just emerged from behind the traffic light and the sign. Dynamic Stixels are visualized after 3 update steps of their Kalman filter system.



(c) A four lane scenario is shown with moving vehicles on every one of them. The background on the left is detected as static. However, the bushes on the right side are estimated as slightly moving.



(d) An intersection is shown with multiple cars in its center coming from our left. The first one of them drives straight, the other two take a left turn.

Figure 4.3: Typical urban traffic scenarios with moving cars are depicted. Each scenario is given with a short description below. The coloring visualizes the estimated motion state as a result of dynamic Stixel tracking. The color scheme for the visualization is explained in Figure 4.2.

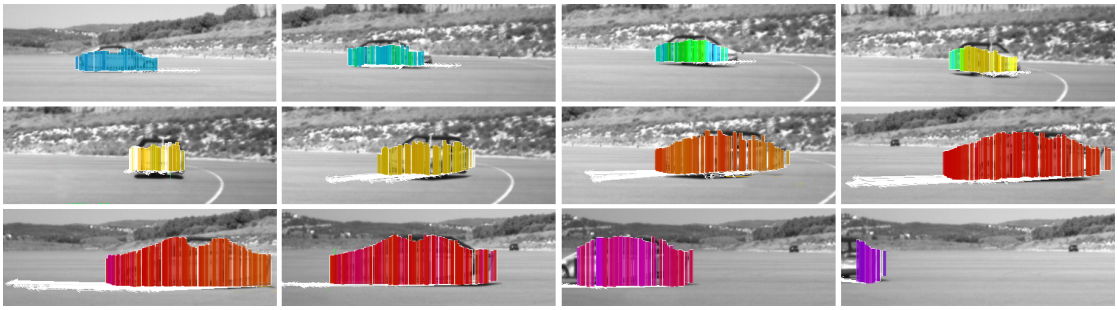


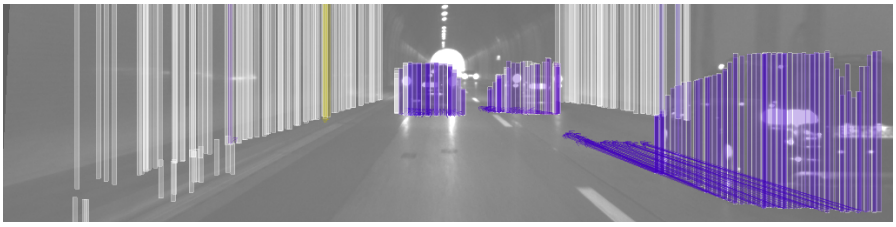
Figure 4.4: A car is shown making a sharp U-turn. The different stages of that maneuver are denoted by the coloring and the arrows that point into the respective moving direction.

or feature-based tracking schemes [66, 159, 177] or approaches dedicated for tracking pedestrians, such as skeleton- or joint-based pedestrian trackers [64, 91, 152, 173, 200], are advisable.

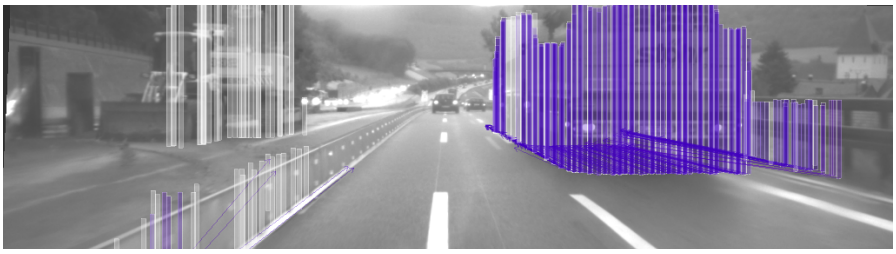
4.4 Extension for the Tracking Scheme

As things stand at present, tracking Stixels over time allows to obtain good and reliable results for the motion states of other objects.

However, as a consequence of the individual non-restricted motion, dynamic Stixels are no longer bound to equidistant columns. That is a drawback of the proposed approach. Especially for subsequent applications, this would be an ideal characteristic to have, because dropping the property to have a regular grid makes things a little bit more complicated. As a further consequence, the total number of dynamic Stixels is not fixed. They begin to overlap partially or can belong to objects at different depths. Converged Stixels are detected as twins and merged according to their normalized innovation squared (NIS) and the variances of their Kalman filters, such that their total maximum number does not increase exorbitantly. Their actual number depends on the parametrization. For a common scenario, an entire scene is described with up to two times as much dynamic as static Stixels.



(a) The highway tunnel scenario exhibits very low light and contrasts. As a result, the perception of the scene infrastructure suffers from weak contrasts and motion blur. In addition, the untextured walls show highlights and reflections, which is challenging for both the stereo reconstruction and the optical flow computation. However, the cars are tracked reliably and the walls remain mostly solid.



(b) Repetitive texture patterns in combination with high driving speeds makes the crash barrier and guard rail hard to track. Concerning this matter, two two extremes are shown. Firstly, due to an unsuccessful optical flow computation on the small crash barrier, the Kalman filters cannot stabilize. This causes a repetitive reset of the tracks. Secondly, the rail to our right appears to be driving with us. That effect is amplified by the truck, as it accelerates the Stixels to its own speed. Then, occasionally, a Stixel slips from the truck onto the rail where its tracking is continued. Again, this process is supported by the repetitive pattern of the rail.

(c) This scenario illustrates the limitation of the Stixel tracking principle with respect to objects that exhibit non-rigid motion, such as pedestrians. The centers of the bodies are tracked quite well, however, the legs and feet are not. For this purpose, a more detailed motion model for pedestrians would be required. For instance, point-based tracking, such as scene flow or 6D-Vision, or schemes that explicitly rely on the shape of human bodies would help to track pedestrians more reliably.

Figure 4.5: The given figures show three challenging scenarios for the tracking process. A common problem is an insufficient frame rate for resolving the optical flow correspondence problem for repetitive patterns. Other structures are not even textured at all. Long exposure times amplify this problem additionally. For non-rigid objects the Stixel-inherent model assumptions are not applicable. This is exemplified by estimating the motion state of walking pedestrians.

5 Evaluation

The Stixel World is a highly flexible representation. Consequently, it is in our interest to use it as input for further processing tasks either as direct measurement input or in terms of attention control. When thinking about safety related applications, such as advanced driver assistance systems, it is a necessity to provide statements regarding reliability and robustness. Unfortunately, the requirement patterns of different applications are often diverse and sometimes even contradictory. For instance, certain applications must not miss a single object while others cannot afford to have phantom detections.

Apparently, making a solid statement is not straightforward and for certain circumstances nearly impossible. However, especially when dealing with advanced safety and assistance systems for automotive application, awareness of such properties becomes a mandatory aspect.

For this reason, the following chapter focuses on a quantitative evaluation of the proposed Stixel World as well as the tracking schemes for motion state estimation. To this end, we proceed in three steps. At first, the measurement precision of the Stixel World is analyzed. For this objective, a high-performance LIDAR for real-world ground truthing is used. That part is based on our previous work presented in [171].

Secondly, reliability concerns of the static Stixel World are discussed. This includes statistical aspects as well as robustness and steadiness measures. For this task, the evaluation relies on large sequence database sets that are used for optimizing the performance of the proposed algorithms.

Thirdly, dedicated effort is put into evaluating the motion estimate for other objects. For this purpose, the different techniques proposed for computing the displacement of Stixels between two consecutive images are tested and discussed. To ensure a comprehensive examination, multiple data sources are considered for input. This also includes ground truth data that is obtained from autonomously driving robotic vehicles with an accurately known position and motion state. In order to obtain meaningful and realistic results, we will also compare the tracking algorithm against other sensors, such as RADAR or the inertial sensors of other vehicles within real-world traffic scenarios.

5.1 Analyzing the Measurement Precision

Naturally, applications that depend on the Stixel representation as input for further processing require reliability measures regarding the given data. In this context, the actual measurement precision plays an essential role.

Within the scope of our previous work "*Ground Truth Evaluation of the Stixel Representation Using Laser Scanners*" [171] we evaluated the distance accuracy that is achieved

by using the Stixel representation. Even though this work is based on the original Stixel World, compared to the techniques proposed in this thesis, the actual steps that deliver the final distance measurement are closely related. Hence, we argue that the evaluation holds its principles of general validity even when the computation scheme is substituted with the newly proposed one.

The achieved quality of the Stixel World directly depends on the quality of the original input stereo data. By relying on semi-global matching [88] for the FPGA implementation [73], one of the top-performing algorithms found in the Middlebury database [183] was chosen. Even though this database is a good platform to compare and rank stereo matching algorithms under controlled and artificial environmental conditions, detailed surveys on the accuracy and reliability of stereo algorithms in real-world and automotive scenarios are still an open topic [101, 155, 154, 198].

Understandably, it is desired to use reference sensors for that task and to automate this process without the need of human interference, such that the verification can be performed on large data sets that cover various real-world scenarios.

5.1.1 Test Setup for Evaluation Using a LIDAR

With the objective to rate the precision of the Stixel World, the use of 3D laser scanners as reference sensors appears to be an evident choice. For this reason, we rely on the Velodyne LIDAR HDL-64E S2 [85]. Note that in Section 3.8.3 the same sensor was also used as data source to directly compute the Stixel World. The stereo vision system used for the evaluation consists of two $1024 \times 334\,\mathrm{px}$ cameras with a $45°$ field of view and a base line of $25\,\mathrm{cm}$.

The stereo rig is attached to the front windshield close to the rear view mirror while the LIDAR is mounted to the roof rack. The corresponding sensor setup for this objective is shown in Figure 5.1.

For comparing the Stixel results with the obtained three-dimensional ground truth data we calibrated the left camera relatively to the LIDAR. This calibration consists of six parameters, namely a three-dimensional translation and rotation, which are determined using the method presented by Lepetit et al. [130]. To this end, a small number of point measurements from the LIDAR and manually selected two-dimensional point correspondences within the left camera image are used. Then, the suggested method allows for a closed-form solution of the perspective-n-point problem, which yields the desired calibration parameters. Alternatively, the parameters can be estimated by means of an iterated closest points (ICP) approach [75, 180].

Given the intrinsic camera parameters, the relative orientation to the LIDAR and the base line of the stereo camera system allows to compute virtual ground truth depth maps from the three-dimensional LIDAR point clouds. Such an image is depicted along with the corresponding SGM depth map in Figure 5.2. These images denote the input data used for the evaluation.

Note that the physically unavoidable deviation of the installation positions between the LIDAR and the left stereo camera leads to overlaps of objects from different depths when projecting three-dimensional LIDAR points into the image domain. This is a negative

Figure 5.1: Photograph of the used test vehicle. It shows the installation of the stereo camera system behind the windshield and the Velodyne LIDAR HDL-64E S2 mounted to the roof rack. This LIDAR unit is used for benchmarking the measurement accuracy that is achieved by using Stixels.

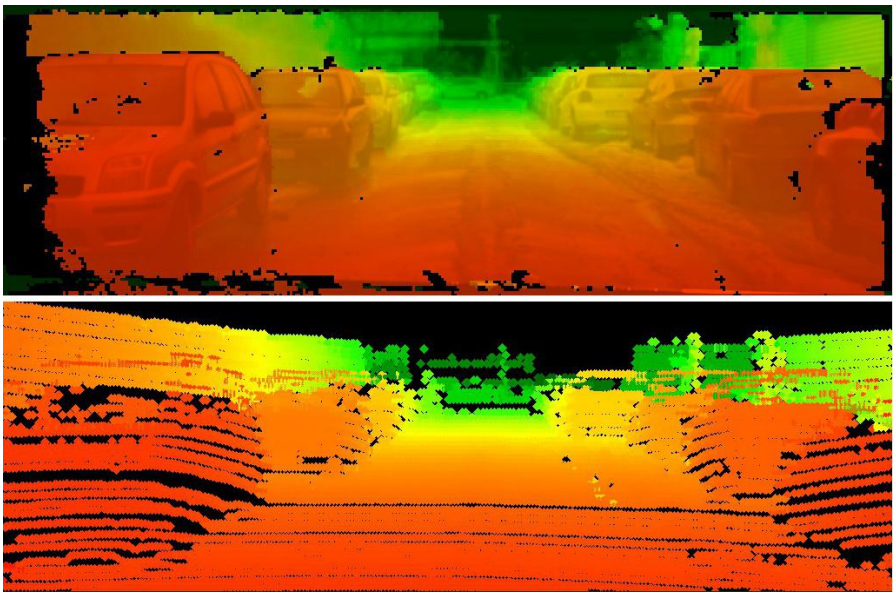


Figure 5.2: Visual comparison of the stereo reconstruction using SGM (top) against a non-artificial ground truth depth map (bottom) obtained from the LIDAR Velodyne HDL-64E S2. For certain types of objects these two sensors behave quite differently. For instance, this can be seen in the windows of the cars.

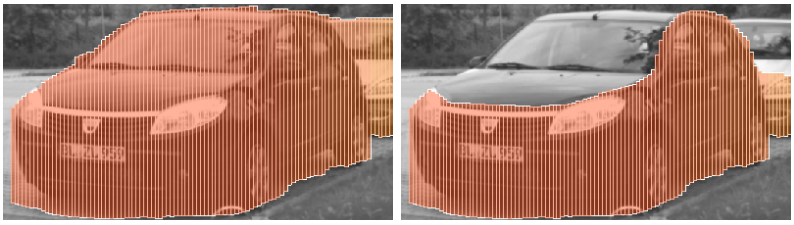


Figure 5.3: Two different ways to represent the same vehicle. On the left side, the complete object is approximated. That view typically corresponds to the human way of perception. In contrast, the right side puts much more emphasis on the ideal Stixel *object* model and the constant disparity assumption.

effect, because the primary concern is to record three-dimensional points that are visible for the LIDAR as well as for the camera and to use those for verification. By choosing a mounting position as close as possible to the left camera, the resulting error cannot be eliminated but kept small.

For avoiding ungrateful issues due to synchronization between the camera system and the LIDAR [52, 186], we decided to capture the sequence using a stop-motion procedure [34]. For further details on the sensors, the testing setup and data preparation please refer to the corresponding publication [171].

5.1.2 Quality Rating and Sensor Characteristics

Instead of falling back to a per-pixel comparison, we chose to determine the measurement quality directly at the Stixel level. To this end, the Stixel distance Z_S is compared to the distance Z_L obtained from the LIDAR. Z_L is simply calculated from those LIDAR points that are projected into the Stixel area within the image. For this purpose, these points are sorted, and Z_L is computed as the mean of their inner 70%. Due to a significantly higher accuracy of the LIDAR in comparison to the stereo sensor, this approach is regarded adequate in terms of precision.

It is important to note that the LIDAR and stereo sensor behave quite differently for specific object classes. A good example for that are reflective or (semi)transparent object parts, such as windows, mirrors or water on the road. The LIDAR either looks right through those objects or follows the reflected ray, while the stereo reconstruction is usually smooth (primarily due to SGM) but often still does not yield a result even close to the expected value. This effect is clearly visible in the virtual depth map that is illustrated in Figure 5.2. For instance, look at the windows of the parked cars. Consequently, these areas should not be used for a ground truth evaluation.

Another concern results from the way how the precision for each Stixel is determined. According to the used model constraints, objects are assumed to have a perpendicular pose. Thus, a constant disparity across the Stixel area is imposed by penalizing deviations within the segmentation process. However, with an increasing height, many slanted objects fail to fulfill that condition. For instance, Figure 5.3 illustrates that effect using the front view of the engine hood of a vehicle. Depending on the strictness with respect

Figure 5.4: For the evaluation the container right to the parked car was used (red box). That object is well-textured, exhibits no protruding parts and satisfies the Stixel *object* model constraints almost perfectly.

to model deviations, the object is either segmented as a whole or is separated at the transition from the front bumper to the engine hood.

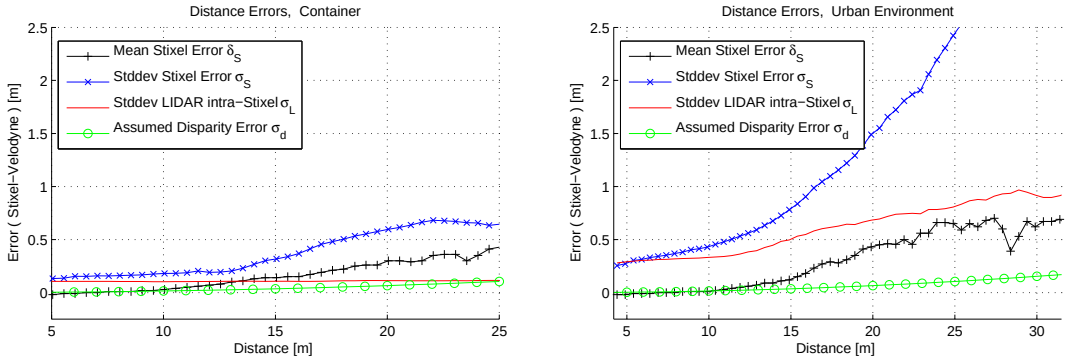
To deal with these particularities properly, the evaluation is split into two parts: At first, the evaluation is performed using a sequence that contains pre-selected perpendicular objects. For this purpose, a well-textured container is slowly approached by the test vehicle. That container has no reflective or transparent parts and thus complies with the imposed rectangularity constraint almost perfectly. A snapshot of the corresponding sequence can be seen in Figure 5.4.

The second part of the evaluation is done within a real urban traffic environment while moving through a narrow street. The image used for illustration in Figure 5.2 is part of that sequence. In this scenario, parked cars reside on both sides of the road. In order to avoid negative influences from the individual sensor characteristics mentioned above, the evaluation of LIDAR disparities is limited to a height of 80 cm. Apart from that, this sequence has a high severity for two further reasons. Firstly, most of the objects are oriented along the road in a very acute angle to our vision system. That arrangement makes the depth measurement quite error-prone. Secondly, those objects exhibit very shiny and reflective surfaces. To make things worse, the acute angles amplify the latter effect additionally. Both factors result in a challenging scene for accuracy investigations of the depth estimate. Both sequences consist of more than 300 frames and contribute about 10000 Stixels each. Note that parts of these test sequences have been made publicly available. For further details refer to [86, 154].

5.1.3 LIDAR Results

Three different types of measures were determined for the evaluation. These include the mean difference δ_S of the Stixel distance Z_S compared to the obtained LIDAR distance Z_L, the standard deviation σ_S of that error and the intra-Stixel standard deviation σ_L of the individual LIDAR distance measurements.

The standard deviation of the SGM disparities have not been considered though. As a result of the SGM smoothness constraints, those are correlated and thus are misleadingly small. Literature concerning stereo evaluation methods often claims a disparity accuracy of up to $\sigma_d = 0.25\,\text{px}$ for reasonably textured objects [36, 76, 89, 184, 215]. Even though this might be true for the pure matching part of the used stereo algorithm, the actual disparity uncertainty as a result from the error aggregation within the whole vision system



(a) Resulting error curves for the evaluation using the container. The LIDAR uncertainty σ_L is constant. Expectedly, the Stixel standard deviation σ_S rises quadratically. The mean error δ_S is increasing quadratically as well.

(b) The curves visualize the obtained errors of the depth estimates for the real-world sequence. Compared to Figure 5.5a, the obtained errors are significantly higher. Surprisingly, the LIDAR standard deviation σ_L is not constant.

Figure 5.5: The left Figure (a) denotes the result for the container scenario, the right Figure (b) shows the results for the narrow road. The mean Stixel error δ_S is drawn in black, the standard deviation σ_S of that error is blue, and the standard deviation of the intra-Stixel LIDAR measurements σ_L is given in red. The corresponding depth-dependent stereo error, when a assuming a disparity standard deviation of $\sigma_d = 0.25\,\text{px}$, is shown in green for reference.

is certainly higher. Unfortunately, this interaction is not resolvable with simple methods. However, in order to aid the interpretation of the given results, the corresponding error curve when assuming $\sigma_d = 0.25\,\text{px}$ as applicable is drawn as well.

At first, the container scenario is discussed. The corresponding error plot is depicted in Figure 5.5a. It is striking (but according to the technical specifications of the Velodyne not surprising) that the standard deviation of the intra-Stixel LIDAR measurements is constant. However, with $\sigma_L \approx 0.1\,\text{m}$ it is certainly higher than expected and we reason this aspect by a weak calibration of the single laser beams.[1]

Further, it is apparent that the mean error δ_S increases with a square dependency. Up to 17 m it is below 0.2 m and rises to 0.45 m at a distance of 25 m. Apparently, Stixels are always estimated as too far away. In that magnitude, this effect is only to explain as an aggregate of inaccuracies in the considerably complex processing chain. That includes the stereo camera calibration, the relative calibration between the stereo rig and the LIDAR, the LIDAR calibration itself or even the rectification step. We will discuss this issue later in this section.

Further, up to a distance of 15 m, the standard deviation of the Stixels mean error σ_S lies within the 3-sigma band of σ_L, but increases rather strongly from there. At a distance of 25 m a standard deviation σ_S of approximately 0.7 m is obtained, which is considered a natural behavior. At this point it is important to note that the curve σ_L for the LIDAR measurements serves as a lower bound for the Stixel standard deviation

[1] As a result of this evaluation, the Velodyne LIDAR was returned to the manufacturer for re-calibration.

σ_S. This characteristic is reasoned by the working principle of the evaluation. Using the LIDAR as reference implies that the obtained results cannot have a higher accuracy than the LIDAR itself. This is because $\sigma_S^2 = \sigma_L^2 + \hat{\sigma}_S^2$ applies, where $\hat{\sigma}_S$ denotes the actual theoretical but unknown uncertainty of the Stixel measurements.

Secondly, the evaluation for the non-constrained scenario with the narrow urban environment is shown in Figure 5.5b. In contrast to the container scenario, it shows that the standard deviation σ_L of the intra-Stixel LIDAR measurements is not constant anymore. By ranging from 0.3 m at 10 m to 1 m at a distance of 30 m it is significantly higher than in the plot shown in Figure 5.5a. In that form, that was not to be expected. However, the reason for that is not solely the LIDAR itself, but also our claim to have a constant distance across each Stixel. Surely, the reflectivity of the objects plays its part as well.

Summarizing, the same conclusion regarding the overestimation of the mean Stixel distances δ_S is drawn. Up to 15 m the mean error δ_S is below 0.3 m and does not exceed 0.7 m at 30 m range. This approximately matches the curve shown in Figure 5.5a. However, the standard deviation σ_S increases significantly stronger with the distance. Up to 15 m it is similar to the LIDAR with $\sigma_S < 1.5 \cdot \sigma_L$, but already exceeds the $3 \cdot \sigma_L$ range at a distance of 26 m with 2.7 m standard deviation.

5.1.4 Correction of Systematic Calibration Errors

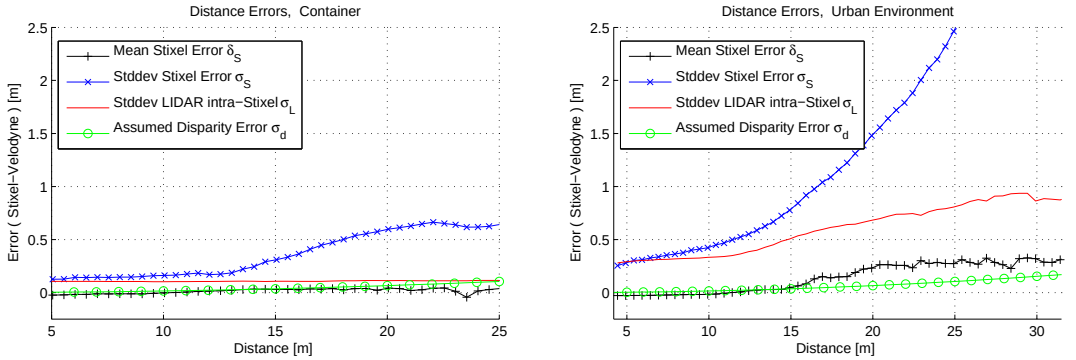
The given plots indicate an interaction of multiple error sources. Even though this is to be expected given the complexity of such vision systems, these errors appear to contain a certain systematic component. This mainly concerns the mean error δ_S. Under the assumption of a correct calibration of both the LIDAR and the stereo camera system, it should have an approximate zero-mean error characteristic. Instead, in both scenarios, the Stixels are estimated as too far away to a degree that is beyond expectation.

Certainly, it is not applicable to have a LIDAR running as a permanent ground truth reference. Thus, the following strategy is used to improve the system performance. The container sequence and its corresponding error statistic obtained with the LIDAR serve as training input. Then, the acquired knowledge is applied to the urban street scenario, and the evaluation procedure is started again.

Within both scenarios, the mean Stixel error δ_S quite precisely follows a quadratic function of the form

$$f_e(z) = az^2 + b \tag{5.1}$$

For the container sequence that is $a = 0.000886$ and $b = -0.0433$. Typically, such quadratic error dependency in Cartesian space results from a disparity offset, which in return can be explained by a squint angle offset in the calibration of the stereo rig. According to camera setup the factor 0.000886 equals a 0.26 px disparity offset. That, respectively, equals a squint angle error of 0.012 deg. The constant term b is considered a remaining error in the longitudinal calibration between the stereo camera system and the LIDAR. Though, it is considered when mapping from image to world coordinates and vice versa.



(a) Error curves for the container scenario after applying the squint angle corrections to the stereo calibration. That error characteristic was learned on the same sequence. Consequently, the mean distance error of the Stixels has been eliminated almost perfectly.

(b) The curves visualize the distance errors for the road sequence after applying the squint angle correction obtained from the container scenario. Roughly, the error is reduced by a factor of two. Naturally, the standard deviation σ_L is not affected by that adjustment.

Figure 5.6: The given plots show the results of the evaluation with a corrected squint angle of the stereo rig. The remaining mean error δ_S of the Stixel distance measurements for the road scenario is reduced significantly. Naturally, the noise characteristics remain unchanged.

When this correction is applied to the calibration of the container sequence, the error curves depicted in Figure 5.6a are obtained. Of course, it is not surprising that the mean Stixel error δ_S coincides the x-axis and thus reflects the expected zero-mean error characteristic. Naturally, this procedure does not affect the disparity noise and therefore does not help to reduce the standard deviation σ_S of the Stixels distances.

Next, the same method is applied to the scene with the parked cars using the statistics obtained from the container sequence. The resulting error curves are given in Figure 5.6b. Even though the result is not as striking as for container scenario, that procedure allows to reduce the error δ_S by a factor of two. Naturally, an elimination of this error was not to expect, since the error statistics for both sequences are clearly different.

The remaining deviation is reasoned with the higher complexity of the real-world scenario. Despite all this, we consider a mean error δ_S of less then 0.4 m at 30 m distance under the given environmental conditions an excellent result. For comparison, this is quite close to the often claimed per pixel disparity standard deviation error of $\sigma_d = 0.25$ px.

In total, this evaluation using the LIDAR made a handful of insights apparent. Besides showing clearly how crucial it is to have a flawless sensor calibration, applying this method also allowed to identify and correct even slight calibration errors within the magnitude of 0.01 deg, as for instance within the squint angle calibration of the stereo camera system.

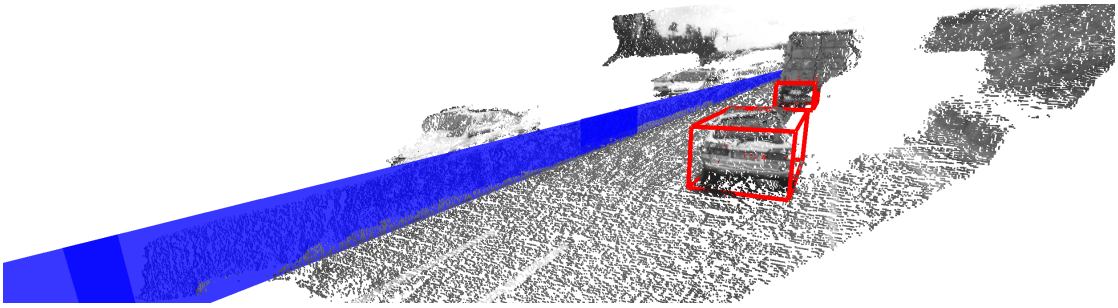


Figure 5.7: Virtual view of the ground truth editor. Solid obstacles are inserted using geometric primitives like planes and boxes (blue). Dynamic scene content is included by using the tracking scheme suggested by Barth et al. (red frames).

5.2 Evaluation of Robustness

Within the scope of this section, the static Stixel World is benchmarked with regard to visual detection range and robustness.

To this end, for the first aspect, a virtual 3D editor is used to create artificial ground truth data. For this purpose, static scene content from recorded sequences is projected into a virtual 3D view. Within that view, scene geometry is defined using basic geometric shapes. For this step, dynamic scene content is taken into account by using the boxed-based tracking scheme proposed by Barth et al. [16, 19]. A snapshot of the editor view is given in Figure 5.7. Even though this procedure allows to extract the motion state of the tracked object, at his point, the obtained data is only used for creating scene ground truth data in order to test the visual range of the proposed object detection technique.

Then, for the latter aspect, robustness tests are performed to evaluate the algorithm's behavior regarding phantom objects. By using a large sequence data base, multiple weather scenarios such as day and night sequences are taken into account.

To begin with, this section starts with a basic analysis of the segmentation statistics for different scenario types.

5.2.1 Statistics for the Static Stixel World

Different typical road scenarios have been considered for collecting basic statistical data of the proposed segmentation technique. This includes an urban environment, a rural road scenario, a forest scenario and a highway sequence. All of these sequences consist of an approximately eight minute drive and thus contribute about $12,000$ frames each. This certainly might not be enough to be statistically meaningful, but should be sufficient for getting a first rough impression of the algorithms behavior for different scenario types.

Note that throughout this evaluation chapter the parameter set for the static Stixel computation is not altered, neither for adapting to different weather or other environmental conditions nor for the different test setups and evaluation aspects. The chosen system parameters are given in Table 5.1.

w_s	σ_d	σ_d^{sky}	σ_Z	σ_{h_cam}	$\sigma_{\alpha_\mathrm{cam}}$
5 px	0.75 px	0.1 px	0.3 m	0.05 m	0.05 rad

$p_\mathrm{out}^{grd,\,obj}$	p_out^{sky}	p_ord	p_grav	p_blg
0.1	0.4	0.1	0.1	0.001

Table 5.1: Assignments of the computational parameters. The given configuration is used throughout all evaluations made within this chapter. For further technical details on these parameters refer to Section 3.4.1 and Section 3.5.2.

At first, the number of segments per column labeled to *object* or *sky* is inspected. According to the choice of the Stixel width w_s, each five consecutive columns are vertically segmented. For an 1024×440 px input image this results in a segmentation of 205 columns in total. However, note that due to certain effects, e.g. occlusion as a result of the left-right displacement of the stereo cameras, not all columns may provide valid depth information.

The summarizing result of this test is depicted in Table 5.2 and the progression itself is illustrated in Figure 5.8. Interestingly, with regard to the average number of *object* and *sky* segments, the different scenarios turn out to be very similar. This is a noteworthy aspect, since these scenarios are topologically clearly different. The standard deviation of the number of segments is the highest within the urban district and forest environment. However, due to the rather complex structure of these scenarios, this characteristic is expected.

	Urban	Rural	Forest	Highway
Number of segments $\{\mathrm{o}, \mathrm{s}\}$ per column	2.83	2.61	2.72	2.78
Standard deviation of seg. per col.	0.49	0.32	0.46	0.21
Avg. standard dev. of the disparities	2.10	0.64	0.88	2.31

Table 5.2: This table shows the average number of *object* and *sky* segments used per column. Apparently, it is quite similar for the different scenarios. All in all, the standard deviation for that measure differs only slightly. However, this is different for the standard deviation of the single disparity measurements within these segments.

Further, the average standard deviation of the disparity measurements within these segments was considered. Naturally, since upright and piecewise planar surfaces are assumed, this value is expected to be close to zero. Thus, it is striking that for urban and highway environments this statistical measure is significantly higher than for the rural road or the forest scenario. However, that behavior has a quite simple reason too and results from approaching other objects closely. That, on the other hand, leads to an inverse proportional increase of the corresponding disparities and thus increase of the standard deviation itself.

Therefore, while for forest and rural road environments other objects are typically

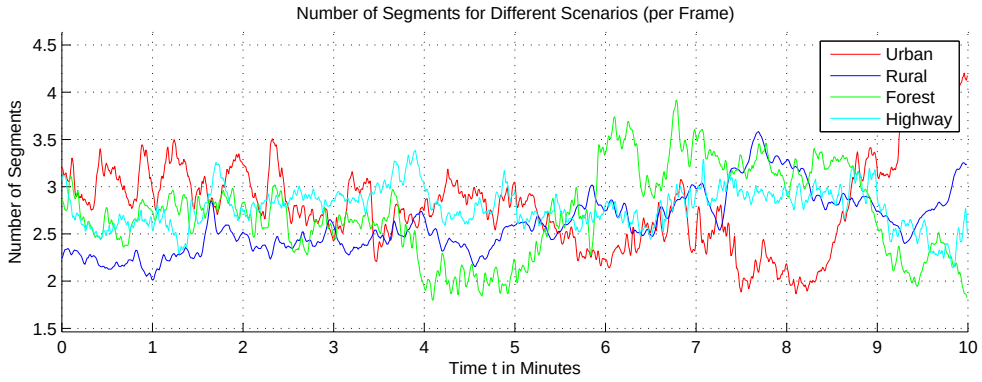


Figure 5.8: Average number of segments (*object* and *sky*) per column. Apparently, irrespective of the actual content, the different scenarios behave quite similar.

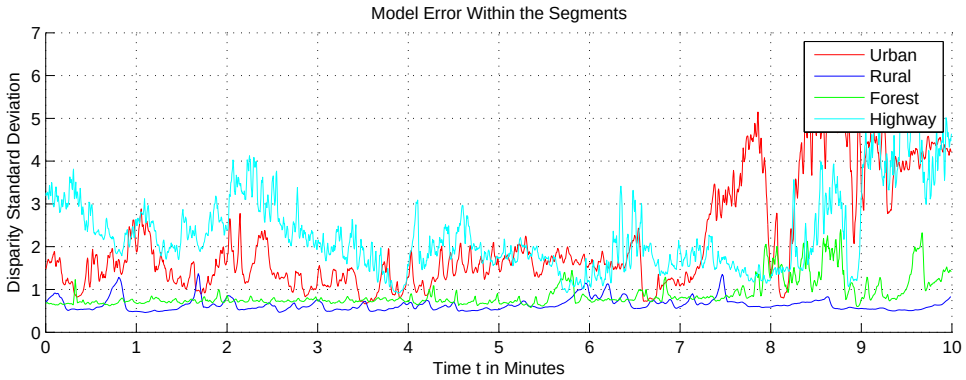


Figure 5.9: Model error expressed by the average standard deviation of the disparity measurements within the segments. Interestingly, the spikes of the dark blue graph mostly correspond to approaching vehicles within the opposing lane.

not approached closely, that aspect is completely different for the urban or the highway scenario. For instance, within the urban environment vehicles parking next to the road or vehicles waiting in line at a red traffic light are objects commonly driven by closely. The same applies for the highway scenario, where other traffic is approached during overtaking, and certain objects (e.g. guard rails or delineator posts) come very close as well. The corresponding graphs that illustrate the progression of the average standard deviation of the disparity values for the discussed scenarios is depicted in Figure 5.9.

5.2.2 Comparison of Visual Detection Range

An important characteristic for object detection systems is the achieved performance in terms of visual detection range. Hence, that issue is discussed in the following. To help with a qualitative comparison, the obtained results are also compared to our previous and well established Stixel computation scheme presented in [9]. The evaluation is performed on a sequence data base that provides both good and adverse weather conditions.

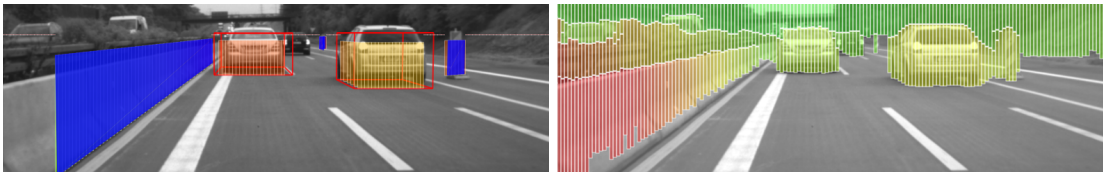


Figure 5.10: A 3D editor is used to create ground truth scene data by hand. The right image shows the corresponding output. The blue walls describe static scene infrastructure, the red boxes are used for object tracking to consider moving objects efficiently. The right image shows the stereo-based Stixel result.

For this objective, appropriate ground truth data is required. Naturally, this is a challenging task for multiple reasons. Firstly, developing systems for real-world application is connected with a certain requirement to test within the same environment the system is built for. Naturally, as just discussed within the LIDAR-based evaluation section, obtaining ground truth data for real-world scenarios is complex and time-consuming.

Secondly, environmental ground truth is strongly dependent on the definition of an obstacle, which on the other hand depends on the targeted application. For example, for certain tasks (e.g. consider lane keeping assistance) it is crucial to detect obstacles as small as curbs, while other tasks (e.g. emergency braking) have clearly different requirements.

For the evaluation, our decision is to characterize such scene content as an object that either poses an imminent threat to the driver and persons involved, or that would cause severe vehicle damage on impact. Since the proposed Stixel representation describes objects using vertically layered models, small obstacles (such as curbs) are not in our focus and are thus not taken into account.

For each sequence within the data base the corresponding ground truth data is created by hand. For this step, the static scene content within each sequence is mapped into a unified virtual view. Then, that view is used to manually create obstacles using simple geometric primitives. Dynamic scene content is incorporated by object tracking. For this purpose, the vehicles are initially selected by hand and then tracked automatically throughout the sequence using the vehicle tracking approach proposed by Barth et al. [16, 19]. Pedestrians, in that context, are modeled as static obstacles. That approximation is considered to be sufficiently accurate for the given task and the chosen scenarios. For each frame of the sequence the modeled objects are used to extract a ground truth Stixel representation. An illustration of that process is given in Figure 5.10.

Finally, a particular ground truth Stixel is considered as detected if a corresponding Stixel is computed from the input images (true positive). Therefore, both Stixels have to be within a depth-range of $\pm 1\,\mathrm{m}$ or $\pm 3\,\mathrm{px}$ disparities. Otherwise, the object is considered as missed (true negative). For both weather types 10 sequences with 250 frames each were used. The results are depicted in Figure 5.11. In this illustration, the goal is to have a detection rate close to one.

Apparently, for both weather types, the approaches are equal at close distances. However, with an increasing visual range, the unified (and hypothesis-based) scheme shows significantly better results than the depth map based approach. For good weather, the

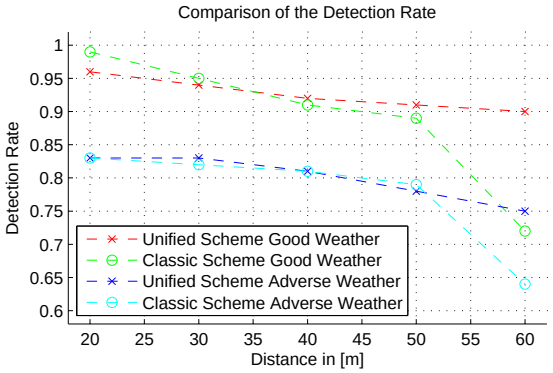


Figure 5.11: This figure compares the two Stixel computation approaches in terms of visual range. This comparison is split into sequences associated to good and bad weather. Both methods are equal at close distances. However, with an increasing range, the unified scheme shows significantly better results than the occupancy grid based scheme.

detection rate is within the range of good 90 percent. Under poor weather conditions, however, it drops to about 83 to 75 percent depending on the distance of the objects to the stereo camera rig.

5.2.3 Occurrence of False Positives

Compared to the classic approach, using the unified scheme allows to improve the visual range significantly. In fact, that increase of performance is reasoned with a higher sensitivity of the proposed method compared to the classic scheme. However, a higher sensitivity is likely to lead to false positives, so called phantom object detections.

Consequently, to draw a meaningful conclusion, it is necessary to take that matter into account as well. A false positive is defined as an object detection that cannot be associated to an actual object in the real world. Typically, those occur on sensor failures, atypical events (such as wipers crossing the windshield), or adverse weather and lighting conditions. A few examples for such scenarios are depicted in Figure 5.12.

According to our personal, human impression, in their present state both approaches already are quite fail-safe and robust. Therefore, only using those 20 sequences as within the previous section will not be sufficient for obtaining meaningful and reliable statistics. Furthermore, especially for the occurrence of false positives, a more precise differentiation with regard to the considered scenarios is desirable. Thus, the evaluation will cover the following aspects: *Daytime*, *Night*, *Rain*, *Heavy Rain* and *Snow*.

As a consequence of the large amount of data, it is neither practical nor expedient to create ground truth data manually. Instead, a different strategy is chosen. Since no collision occurred while recording the sequences, it is most likely that the space in front of the vehicle is free of obstacles. The vehicle's driven path through the three-dimensional scene is reconstructed by looking ahead the odometry information (velocity and yaw rate)

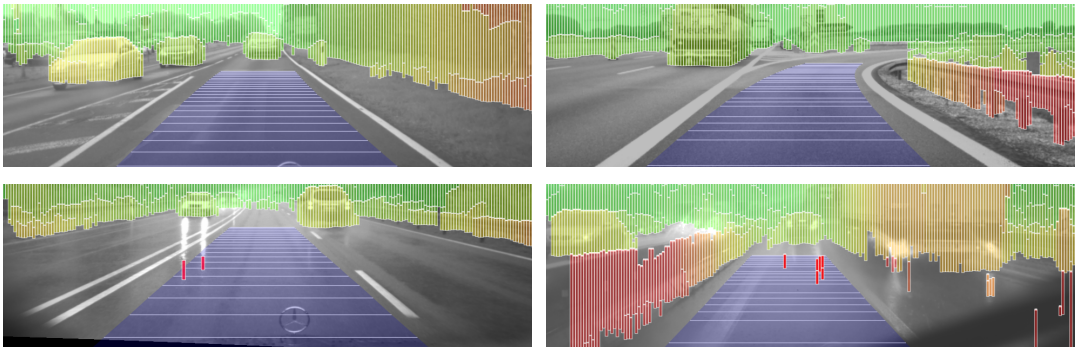


Figure 5.12: A virtual driving path (dark blue area) is computed from the vehicle's odometry information to automatically detect false positives (phantom Stixels). Object detections within the driving corridor are considered as false detection (marked in red) if they are not supported by the RADAR sensor.

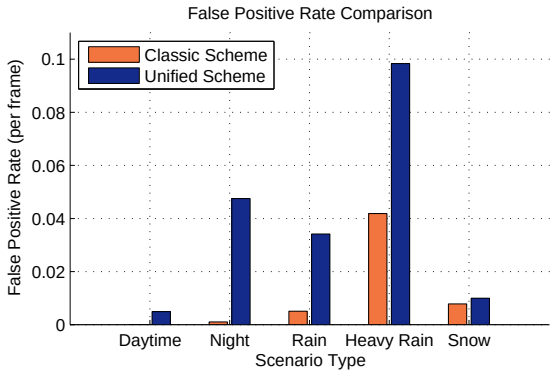
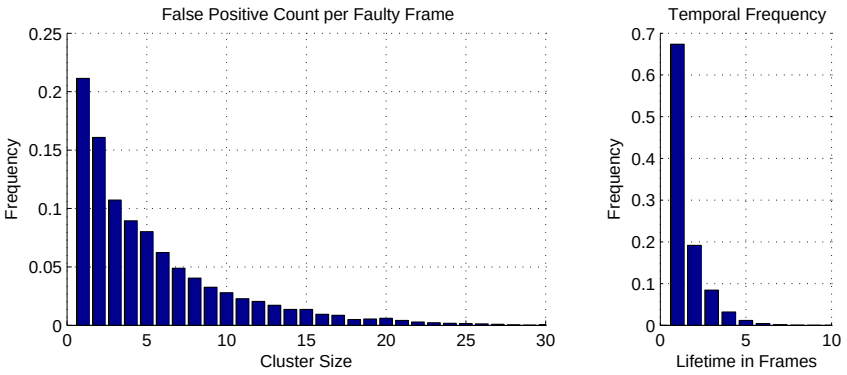


Figure 5.13: This plot shows the false positive rate in comparison for both Stixel computation schemes. That measure computes from the ratio of the number of frames with false detections to the total number of frames. Apparently, the unified approach is more susceptible to that matter.

from the recorded sequence. In case of having a leading vehicle, the actual free space is determined using an independent RADAR sensor (Continental ARS300 long range RADAR [188]). In this process, the integrity of the RADAR is not questioned. For a better understanding, exemplary virtual driving paths are visualized in Figure 5.12.

All Stixel observations that fall into the driven path are considered to be false detections. Surely, by just taking that area into account, large amounts of information are discarded. However, following that strategy allows to process many sequences without the need for human inspection or interaction. In return, that gives the opportunity to evaluate very large sequence data bases with low effort. Hence, for the evaluation, each scenario type was considered with about 200 sequences consisting of approximately 250 frames each.

Finally, the false positive rate is computed. It corresponds to the ratio of the number



(a) (b)

Figure 5.14: Evaluation of false positives characteristics. About 75 % of them consist of five Stixels or less. More than 95 % of all false positives have a lifetime of less than four frames. A few typical examples are depicted in Figure 5.12.

of frames exhibiting object detections within the computed driving corridor and the total number of frames. The evaluation results are depicted in Figure 5.13.

Apparently, in terms of robustness, the classic approach outperforms the new one by lengths. Nevertheless, to put things into perspective, a few explaining words are required. Firstly, the unified scheme allows for multiple object segments along every column. As a result from partly strong disparity noise in low-textured areas (such as sky or road) it occurs that small obstacles hover above the road. Even when the evaluation is limited to Stixels with a base point beneath 2 m in height, that effect occasionally leads to implausible object registrations within the driving path. The classic approach, on the other hand, only extracts the first obstacle row that starts at the end of the freespace which does not necessarily correspond to the closest obstacle.

Secondly, the classic approach applies a few but significant thresholds and low-pass filters. This mainly relates to the question of when and how measurements are registered into the occupancy grids. For instance, disparity measurements with a height below 0.25 m are typically discarded for this step (for further details see [10]). As a consequence, noise on the ground surface has almost no effect for the reconstruction result. Now one can argue that this is a good property. However, at the same time, the reconstruction of small obstacles such as high curbs or pylons is strongly restricted.

In terms of this evaluation, another important aspect concerns the spatial and temporal characteristic of false detections. Thus, in the following, these aspects are highlighted with more detail.

At first, the number of connected phantom Stixels is analyzed. In Figure 5.14a it is shown that about 75 % of all false positives consist of five Stixels or less. Accordingly, false detections emerge as relatively small objects.

Secondly, the temporal characteristic of false positives is targeted. To proceed with this objective, the following (pessimistic) strategy is suggested: Upon a false detection,

the corresponding Stixel is projected into the next time step by using the vehicle's ego-motion information. In this process, the Stixel is assumed to represent a static obstacle. Then, if within a spatial range of 2 m around that projected Stixel another false detection is registered, those two detections are considered as temporarily related.

The corresponding result of this test is presented in Figure 5.14b. Accordingly, about 68 % of all false detections relate to singular events, and more than 95 % have a lifetime of three frames or less. Thus, it is deduced that the temporal occurrence of false positives is quite sporadic. With respect to object tracking that means that more than 95 % of falsely tracked objects are discarded within the first three frames of their lifetime.

Note that a very similar characteristic is observed when performing the same test for the occupancy grid based Stixel approach. Hence, these results are applicable for the classic computation scheme just as well.

5.3 Evaluation of the Trackers and Optical Flow Methods

Within Section 4.2 different Stixel tracking strategies have been presented. For this reason, the current section aims to evaluate their performance and the quality of the estimated motion states.

In terms of the object tracking, a core aspect is the computation of the Stixel displacement between the images of two consecutive time steps. For that task, Section 4.2 discussed multiple approaches that differ with respect to their technical prerequisites, their scope of action to combine the optical flow computation directly with the actual tracking process, and their computational effort. Thus, we will investigate how those strategies perform for different scenarios. Additionally, for the proposed patch-KLT method (see Section 4.2.5) different parameter variations are tested in order to determine a good choice for the size of the feature patches.

Hence, for the evaluation we proceed as follows. At first, the Stixel tracking is tested in stationary environments that contain no moving objects. Even though our own car is moving, the objective is to detect that the environment remains static. This strategy is straightforward and can be carried out quite easily. Secondly, robotic vehicles are used. Those are equipped with high-precision inertial measurement units that output a highly accurate motion state. Subsequently, that sensor data is used as ground truth data for the evaluation process to judge the accuracy of the extracted motion state and to test the response of the filter systems. Thirdly, a non-restricted open-road scenario is considered. Two vehicles participate in that scenario, one that drives ahead and one that follows. Using both the vehicles' inertial motion sensors and RADAR technology allows to obtain valuable reference data.

5.3.1 Static Environments

For the first test, a static environment represented by a narrow street with cars parked on both sides is chosen. That environment contains no further moving objects. Consequently, the expectation for the motion state of all tracked objects is to have no velocity. A snapshot of that sequence is depicted in Figure 5.15. The scenario was recorded several

(a) Correct estimation of the environment to consist of static objects only. Thus, all Stixels are drawn with a white coloring which denotes a velocity close to zero.

(b) Three typical sources for velocity errors during tracking within static environments are illustrated. The left figure shows a reflecting surface, the middle figure shows a jump in depth, and the third figure shows difficulties with motion estimation at large distances.

Figure 5.15: Color visualizes motion. Ideally, all static objects should have a white coloring with zero velocity. When driving through static environments, that issue is used as a quality rating for the different tracking schemes. Figure (a) shows a good example, Figure (b) shows typical sources of error.

times while driving at different speeds. That includes an ego-motion velocity of 4, 8, 14 and 20 $^{\mathrm{m}}/_{\mathrm{s}}$.

For estimating the optical flow between consecutive time steps the Census-based feature flow proposed by Stein et al. [196], the dense TV-L1 based optical flow scheme proposed by Müller et al. [161], the KLT-based feature tracker proposed by Tomasi et al. [214] and our patch-KLT method were used. For the latter, a patch size of 40×16 px (width×height) was chosen. According to our experiments, this patch size turned out to yield the best performance. Nevertheless, an additional benchmark for different patch sizes is shown later in this section.

The results for the static environment are depicted in Figure 5.16. On the left side, for rating the tracking performance of the individual tracking schemes, the mean absolute velocity of all tracked Stixels is computed. Depending on the ego-velocity of the test vehicle, each sequence contributes 300 to 1,000 frames. The evaluation is limited to a distance of 40 m to suppress unwanted influences of perspective effects.

Secondly, the figure on the right side depicts the resulting errors in estimating the object velocities depending on the depth of the Stixel. For this purpose, the patch-KLT method is used. The test is performed for all ego-velocities.

Apparently, for the considered setup, the different optical flow schemes are evenly matched, such that there is no clear winner. Depending on the driven speed it is shown,

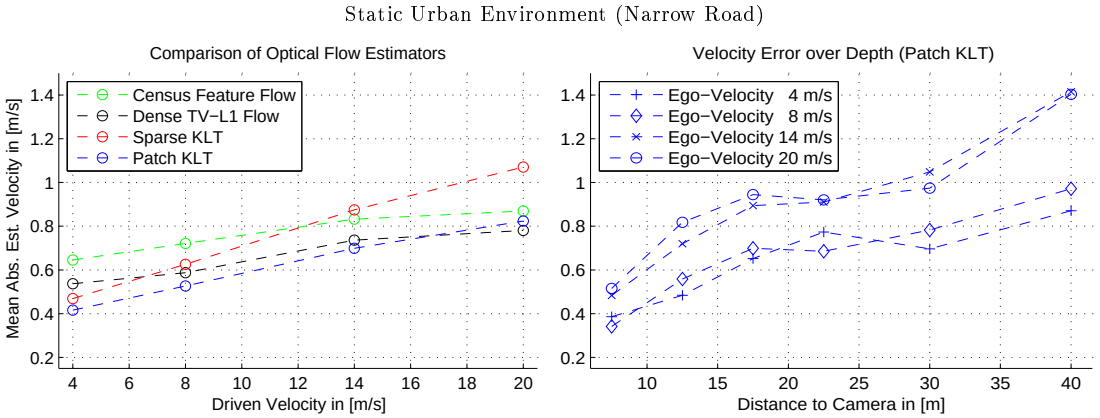


Figure 5.16: Direct comparison of the four different tracking schemes. Ideally, the mean absolute velocity should be zero. The quality of the optical flow measurement plays a significant role in that process. Thus, depending on the used optical flow scheme, that goal is reached more or less well. However, for the considered static urban environment, the differences are rather small.

that the mean velocity errors of all schemes rise with a linear characteristic. Yet, in reference to the total system complexity, that error is relatively small and lies between 6% and 8% of the driven ego-velocity. The depth dependent velocity errors also show an approximate linear increase with the distance. The obtained error curves seem plausible and match with our expectations.

Altogether, the good performance of the investigated techniques is reasoned in the fact that the considered scenario is relatively simple. Thus, by changing to a highway-like environment, a more challenging scenario is taken into account. It features neither cars nor moving objects but has guard rails on both sides of the road. Naturally, due to their repetitive patterns, guard rails are likely to cause problems for the optical flow computation when driving along in parallel at high speeds. These problems are widely referred to as the aperture problem or the blank wall problem [39, 139, 212]. A phenomenon that, in reference to object tracking, is typical for that effect is illustrated in Figure 5.17. To increase the degree of difficulty of the evaluation additionally, the ego-velocity is slightly increased. Thus, the scenario is driven at the speeds of 8, 14, 20, 28 and 36 $^{\mathrm{m}}/_{\mathrm{s}}$.

Besides that, when looking at Figure 5.17a, another important aspect becomes obvious. Problems within the optical flow estimation can lead to holes within the line of Stixels covering the guard rail, which is typically caused by missing or faulty optical flow measurements. Therefore, in order to draw a more practical conclusion, the performed tests will supplementary include the completeness measure for the guard rail. This ratio is determined using the method proposed in Section 5.2.2. The corresponding evaluation results are shown in Figure 5.18. Depending on the driven speed, each sequence contributes approximately 250 to 600 frames.

Contrary to the previous test, the highway environment reveals severe differences between the tracking techniques. Depending on the particular tracking procedure, the

(a) Unreliable optical flow estimates on guard rails lead to wrong Stixel velocity estimates. Additionally, the guard rail is not covered completely.

(b) In contrast, successful optical flow computation allows to obtain correct Stixel velocity estimates. The rails is covered much better.

Figure 5.17: A precise optical flow estimate is essential for estimating the Stixel motion state reliably. Especially for structures that suffer from aperture problems at high ego velocities, this is a very challenging task. Therefore, that matter is exemplified by means of a guard rail scenario.

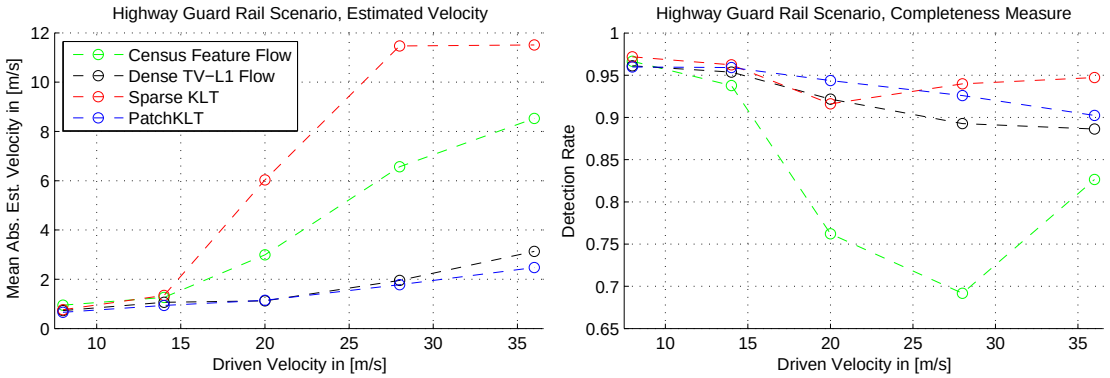


Figure 5.18: This figure shows the results of the performance evaluation for the different tracking strategies using the guard rail scenarios. The left figure denotes the remaining mean absolute velocity for the different driven vehicle speeds (8, 14, 20, 28 and 36 m/s). Correspondingly, the right side denotes the achieved completeness measure of Stixels covering the guard rail.

velocity estimates as well as the detection rates vary noticeably. The best trade-off with respect to a low velocity error and a satisfying completeness measure is achieved by using the proposed patch-KLT procedure or dense TV-L1 optical flow as proposed in [161]. Altogether, those two schemes are evenly matched. In contrast, the point feature based KLT method and the Census-based optical flow tracking scheme have serious difficulties with estimating the velocity correctly. The sparse KLT method yields a good completeness ratio, but its mean absolute estimated velocity is unacceptably high when driving faster than 14 m/s. Even though the Census-based feature flow performs slightly better, the achieved velocity estimate is still not good enough to be used in terms of our objectives. Also, that tracking scheme has severe problems regarding the detection rate. Thus, when going 14 m/s or faster, that ratio drops below 75% completeness rapidly.

The good performance of the TV-L1 based optical flow is reasoned by the fact, that for every image the assumption of the world to remain static is used as a weak but apparently effective regularization prior for the optical flow estimation. Additionally, the globally optimizing characteristic supports to find a solution that is smooth and thus supports our world model too.

With regard to the patch-KLT, things are quite similar. The used tracking scheme makes strong use of the Kalman filter prediction as a feed-forward signal. This clearly helps to resolve textural ambiguities of the tracked structures. Now, even though the sparse feature based KLT technique allows for the same procedure, things behave a little different here. For our understanding, the weak performance of the sparse KLT method (as we have used it for this test) results from not considering the change of scale for the feature patch. Thus, even though the Kalman filter system initially predicts the optical flow vector to the right position, the rigid feature patch and the ambiguity of the guard rail texture effects for the estimate optical flow vector to crawl back to the feature's position within the last frame.

Last but not least, for the static environments, the highway scenario is used to compare the performance of the different feature sizes for the patch-KLT method. For this purpose the following five configurations have been chosen: 15×9 px, 25×13 px, 40×16 px, 60×75 px and 75×30 px (width×height). At this point we remind, that using a disproportionately large feature patch does not necessarily lead to bad consequences for the tracked object (e.g. foreground fattening [184]). That aspect is accounted for by balancing the influence of each pixel using a disparity weight. Pixels that have a depth similar to the tracked object are weighted higher than those that lie in the background. For all tested variants, the remaining parameters of the algorithm are left the same. The corresponding results are depicted in Figure 5.19.

Up to an ego-velocity of 20 m/s the estimated mean absolute velocity do not differ at all. They increase equally from 0.8 m/s error at 8 m/s driven speed to 1.2 m/s error at driven 20 m/s. However, starting from there, the tracking performance of the different configurations begin to differ noticeably. Interestingly and contrary to our initial expectation is, that the obtained error does grow with the size for the feature patches. Yet again it has to be considered, that a plausible performance rating can only be made when looking at the detection rate at the same time. Here, it is exactly the other way around, such that a large feature size is helpful to obtain good detection rates.

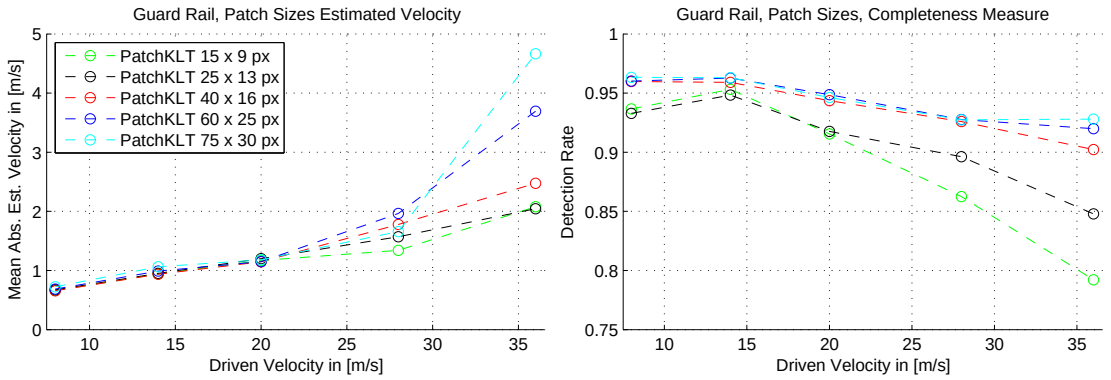


Figure 5.19: Performance comparison resulting from using different feature sizes for the proposed patch-KLT tracking method. Up to 20 m/s, the differences are insignificant. For higher ego-velocities a bigger patch size does not necessarily lead to a more precise motion estimate. However, larger patch sizes help to yield good detection rates. According to our experience and our test setup, the 40×16 patch size is the best performer.

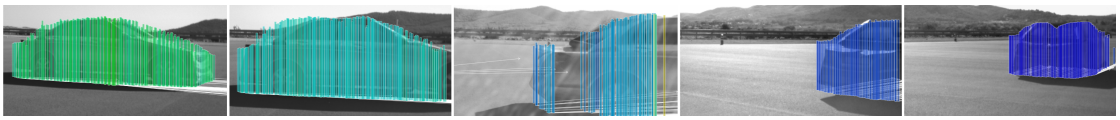
As a consequence, from our point of view, the best trade-off between small velocity errors and detection rate is achieved by using 40×16 px feature patches.

The linear increase of the motion estimate error that has been observed during all tests is mainly reasoned by weaknesses in the calibration of the sensors including the stereo camera system and the vehicles' inertial motion sensor. In addition, given the high overall system complexity makes it difficult to detect and exclude systematic programming errors.

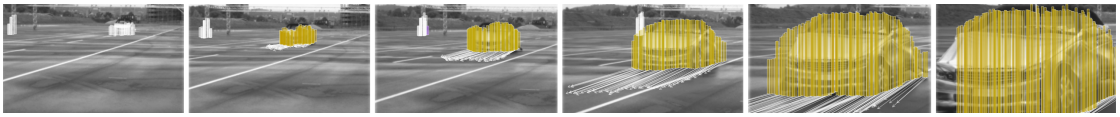
5.3.2 Robotic Scenarios

The following section aims to evaluate the quality of the estimated Stixel velocity for moving objects with available ground truth motion data. For this task, remote controlled vehicles are used that allow for driving predefined maneuvers repeatedly with high precision. To this end, they are equipped with a high-precision inertial measurement unit (IMU). In our case, two iMAR iTrace-RT-F200 [107] IMUs are used. These military-grade devices use differential GPS and fiber optic gyro technology to yield a highly accurate motion state. One IMU is attached to the robotic vehicle and the other one is attached to the testing vehicle that also carries the stereo camera equipment. The sequence recording was done on an open surface to guarantee availability of GPS satellites. The used stereo rig consists of two cameras that have a base line of 20 cm and a resolution of 1024×334 px. Note that the base line is noticeably smaller than for the LIDAR experiments.

For the experiments, two sequences were chosen. The first one shows a robotic vehicle passing our sight from left to right and the other one shows a vehicle moving straight towards our position. Snapshots of both sequences are shown in Figure 5.20.



(a) A robotic vehicle is passing from left to right. Reflections on the windshield cause bad depth and optical flow measurements. Thus, not all parts of the vehicle are tracked reliably.



(b) Sequence of an approaching robotic vehicle. That vehicle starts at a distance of approximately 60 m and closely passes our vehicle to the left.

Figure 5.20: Robotic vehicles are used to obtain precise ground truth motion data. That data is used for testing the Stixel Kalman filter systems. The corresponding quantitative test results are shown in Figure 5.21.

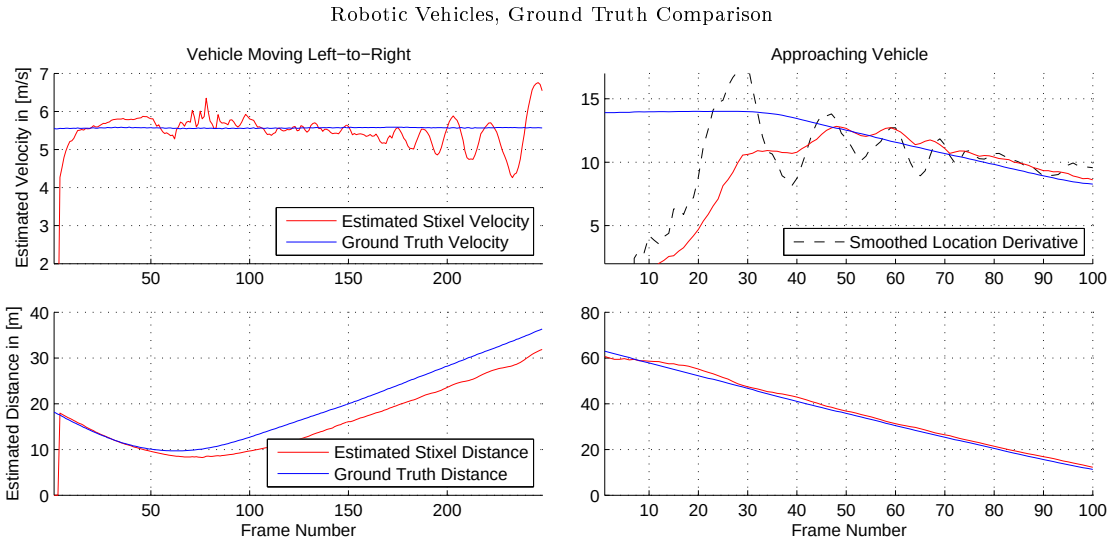


Figure 5.21: Velocity and distance estimate for the tests with robotic vehicles are shown. Using high-precision IMUs and DGPS for these platforms allows to use their location and motion for ground truth comparison. The given plots are the corresponding results to the scenarios visualized in Figure 5.20.

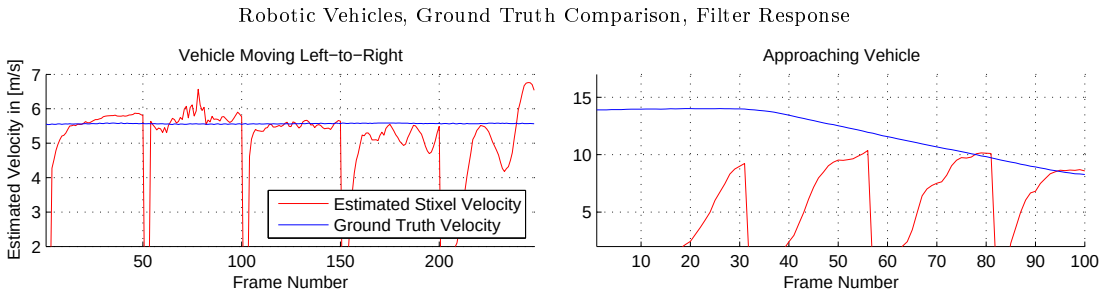


Figure 5.22: These results are obtained from resetting the tracking module every two seconds (i.e. 50 frames). The curves show the Kalman filter response. Adapting to lateral movement (left-to-right scenario) is done within a few frames. In contrast, longitudinal movement requires significantly more frames.

The test results for both scenarios are depicted in Figure 5.21. The given graphs show both the estimated velocity and the estimated distance to our test vehicle. The vehicle's motion state is computed using a robust mean velocity of all Stixels that correspond to the vehicle.

The results for the left-to-right scenario are straightforward. For this test, the robotic vehicle moves with a constant velocity of approximately 5.5 m/s. Up to the end, when the robotic vehicle increases its distance to our test platform, the velocity estimate starts to pulsate within the range of ± 1 m/s. However, that has only little effect on the estimated distance. The corresponding curve for the distance measurement is increasing smoothly.

The second scenario is challenging despite its simple arrangement. That is because the robotic vehicles approaches from a large distance at which the Kalman filters do not converge that easily. Hence, the motion trackers have to stabilize within a range that pushes the stereo-based system to the limit. Even though the vehicle is detected at 60 m distance (see the corresponding plot), the motion estimate does not build up until a distance of approximately 40 m is reached. Certainly, a larger base line of the stereo rig would help to yield a larger range.

A few extra words are required for explaining a minor, otherwise misleading effect. For both test cases, the distance measurement shows a small offset between the ground truth data and our results. However, that offset is not necessarily an inaccuracy in our vision system but is primarily reasoned by the fact that the ground truth distance to the testing vehicle is measured to its front. So, for instance, at the end of the left-to-right scenario, the car moves away from our position. As a result, the Stixels are representing its rear but not its front bumper. Hence, the deviation between ground truth and our measurements is also due to the length of the robotic vehicle.

For the last robotic vehicle test the filter response is highlighted. For that purpose, the Kalman filter system is reset every two seconds (50 frames).

The resulting velocity plots are depicted in Figure 5.22. It is shown, that the chosen Kalman filter configuration is capable to adapt to lateral movement (e.g. left-to-right scenario) much faster than to longitudinal movement (e.g. approaching vehicle). That

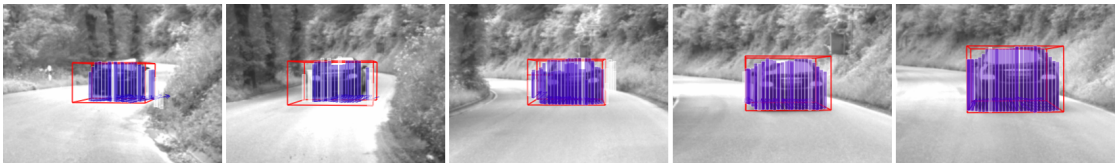


Figure 5.23: This figure shows a selection of images from a real-world sequence used for performance evaluation. The motion state of the leading vehicle is estimated using the Stixel-based tracking scheme and compared to reference data. For this purpose, the leading vehicles inertial motion sensors are used and the following vehicle is equipped with a RADAR unit.

is reasoned by the fact that the capability of the stereo camera with respect to lateral movement (angular resolution) is significantly higher than for the distance measurement. Improving the convergence speed of the filter system to longitudinal movement can be achieved in different ways. For example, the measurement variance of the filter system can be reduced. However, doing so would result in gaining much more noisy motion estimates on static objects. Alternatively, a better filter initialization can be performed, for instance by using RADAR information. Though, doing so implies that the RADAR has already detected the object properly before the stereo camera does.

5.3.3 Real-World Ground Truth

At last, with the objective to rate the precision of the Stixel motion estimate in a real-world experiment, a common non-artificial open-road environment was chosen. Two vehicles participate in that scenario, one that precedes and one that follows. Equipped with the stereo camera system in the following vehicle, the goal is to estimate the motion state of the leading vehicle using the proposed Stixel tracking scheme and to compare the extracted information to reference data. Similar to the previous test, the motion state is computed from the Stixels that correspond to the vehicle.

Since our testing vehicle is keeping pace with the leading vehicle, that scenario allows to test for high velocities and dynamic ranges. Thus, in addition, for making the evaluation as meaningful as possible, the selected road course is serpentine and the leading vehicle performs alternating acceleration and braking phases. For this test a camera pair with a resolution of 1024×440 px and a base line of 23 cm was used. To warrant reliable stereo information, the calibration was carefully counter-checked, especially for yaw and squint angle offsets.

Reference motion data for the leading vehicle is obtained by following two different strategies. At first, the inertial motion sensor (IMS) data of the leading vehicles is recorded. For this purpose, the vehicle's standard OEM sensors are used.

Secondly, the rear vehicle is equipped with a RADAR unit (Continental ARS300 long range RADAR [188]), the same one that was already used for the false positive analysis in Section 5.2.3. That RADAR unit delivers a location and a velocity estimate for all detected objects.

An illustration of that sequence is shown in Figure 5.23. The recorded sequence has a duration of approximately 140 seconds (3,500 frames). Figure 5.24a shows the direct comparison of the leading vehicle's estimated motion state to the sensor units (IMS Leading and RADAR). Further, the ego-motion sensor data of the following vehicle is shown as well (IMS Recording). Note that both vehicle motion sensors have been calibrated to avoid negative effects from ego-velocity errors.

Apparently, the three curves for the Stixel motion estimate, the RADAR information and the IMS sensors coincide very well. Therefore, for a better interpretation, the actual differences are plotted in Figure 5.24b. Additionally, Table 5.3 gives a brief summarizing overview about the most significant statistics. In total, both reference sensors validate a mean difference to the Stixel motion estimate of 0.14 m/s or better, while the corresponding standard deviation is 0.42 m/s or less. The IMS and the RADAR sensors evince a mean deviation of 0.044 m/s and a standard deviation of 0.29 m/s. Hence, the Stixel-based motion estimate performs in an error-range of the same magnitude. From this point of view, the obtained results are very encouraging.

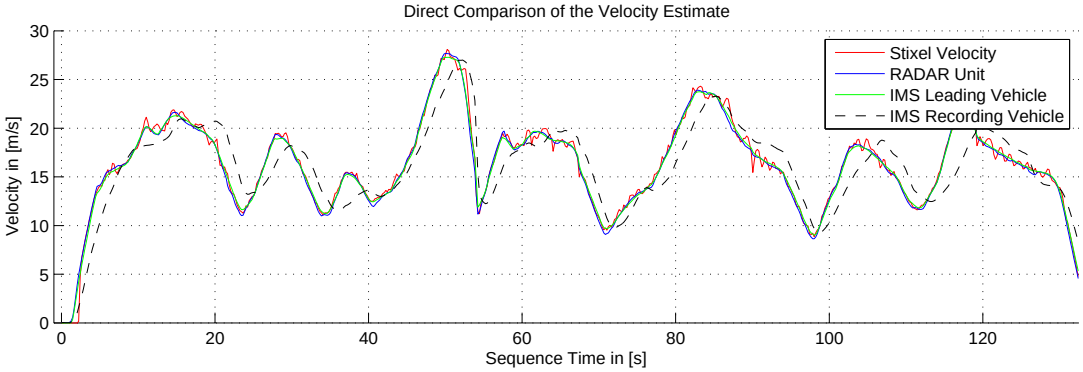
Direct Velocity Comparison

Stixel	RADAR	IMS
μ_{Stixel}	μ_{RADAR}	μ_{IMS}
16.25 m/s	16.11 m/s	16.15 m/s
σ_{Stixel}	σ_{RADAR}	σ_{IMS}
4.67 m/s	4.62 m/s	4.55 m/s

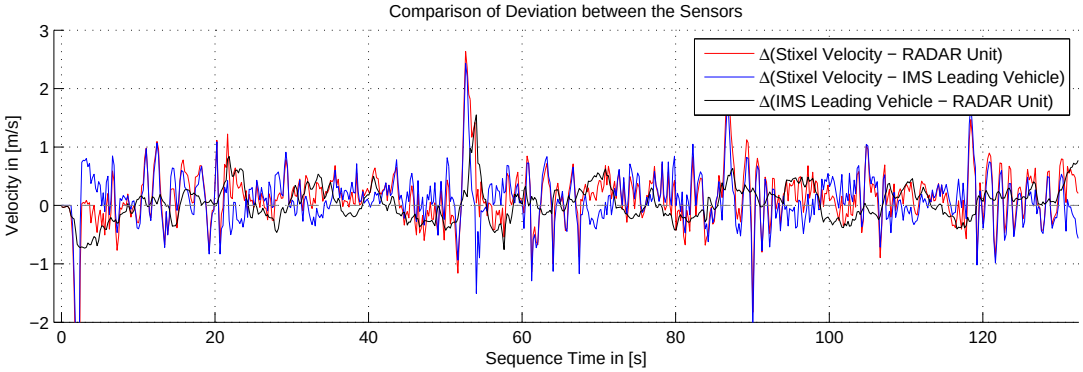
Velocity Difference Between the Sensors

Stixel − IMS	Stixel − RADAR	IMS − RADARS
$\mu_{\text{Stixel−IMS}}$	$\mu_{\text{Stixel-RADAR}}$	$\mu_{\text{IMS−RADARS}}$
0.09 m/s	0.14 m/s	0.044 m/s
$\sigma_{\text{Stixel−IMS}}$	$\sigma_{\text{Stixel-RADAR}}$	$\sigma_{\text{IMS−RADARS}}$
0.37 m/s	0.42 m/s	0.29 m/s

Table 5.3: The table gives a summarizing view on the extracted motion state of the leading vehicle using the Stixel-based tracking approach. The RADAR sensor unit as well as the inertial motion sensors of the leading vehicle are given too. The right table allows for a differential comparison of the different sensors.



(a) Direct comparison of the estimated velocity of the leading vehicle to the RADAR unit and the leading vehicles inertial motion sensors. Apparently, these three curves coincide quite well. Thus, to allow a better comparison, a differential view is given in Figure 5.24b. The dotted line denotes the velocity of the following vehicle. It is delayed to the leading vehicle a few seconds.



(b) This figure visualizes the difference between the Stixel motion estimate to the reference sensors as well as the difference of the reference sensors themselves. All curves oscillate around zero which indicates a mean-free characteristic. A summarizing view on this evaluation is given in Table 5.3.

Figure 5.24: The illustrations show a comparison of the Stixel-based vehicle motion estimate to reference sensor data. The upper figure gives the direct comparison, the lower one shows the differential sensor behavior.

6 Application of the Stixel World

In order to optimize available computational resources and to yield a structured and well-arranged architecture for the vision system, we consider the Stixel World as the ideal basic building block. Accordingly, the Stixel World found successful application within various different vision algorithms. For this reason, in the following, exemplary vision algorithms that rely on Stixels are sketched and the respective motivation and benefits for using Stixels in this process is discussed briefly.

6.1 Vehicle Tracking Using Dedicated Motion Models

In [16], Barth et al. have presented a technique for tracking vehicles in urban environments. By using a sophisticated vehicle motion model, their tracking scheme not only estimates the location and velocity, but also the acceleration, orientation as well as the yaw rate for three-dimensional, ideally box-shaped objects. The initialization trigger for their algorithm is obtained from using 6D-Vision [66, 176] point features. For updating their Kalman filter system with new measurements, the authors process sparse KLT-feature tracks and access dense stereo directly to avoid effects from filter cascades.

The initial scheme of Barth et al. exhibits two critical points. The first relates to the box initialization phase, where it is essential to have a reliable initial box alignment for the newly tracked object. The second point concerns the tracking process itself. Here it is important to continuously update the filter with reliable pose information in subsequent tracking steps for minimizing instabilities in the vehicle motion model.

Accordingly, in [18], we have proposed to rely on Stixels in order to aid the tracking process with both tasks. The illustrations shown in the following are provided by courtesy of Alexander Barth.

At first, the initialization step is discussed. Initially, Barth et al. [16] uses 6D-Vision point features to detect and isolate vehicles. The object's dimensions itself are given by fixed control parameters. However, that strategy runs the risk to yield misplaced or wrongly oriented boxes, such as the white-framed box to the right in Figure 6.1.

For this reason, we extend the object tracking initialization with an additional pose refinement step. To this end, Stixels are grouped to clusters using basic spatial criteria. Then, these clusters are probed for L-shape structures [233], where one side corresponds to either the front or the rear and the other side corresponds to one of the vehicle's long sides. The obtained surface pair is used to adjust the box to the three-dimensional shape of the vehicle. For clarity, that issue is explained visually in Figure 6.1. Results of the proposed procedure are presented in Figure 6.2. For the given misfitted example, the orange box in both Figures shows the realigned version. Obviously, it fits the observed

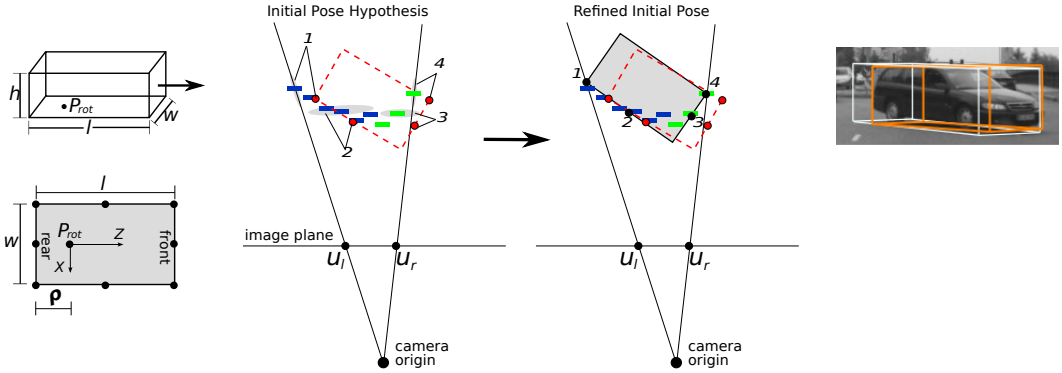
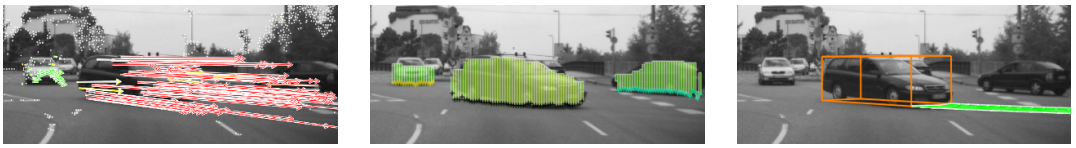


Figure 6.1: The orientation parameters of the box-shaped vehicle model are illustrated. For the tracking process to work properly it is important to have a precise initial alignment of that box to the vehicle. For this purpose, Stixels are used. They are tested for L-shapes, such that one side denotes the front/rear view of the vehicle (green), and the other side (blue) denotes the side view.



(a) 6D-Vision result. The arrows show the predicted feature position in 1 second.

(b) Clustered Stixel representation used to constrain the bounding box orientation.

(c) Resulting object box with location, orientation as well as motion and yaw rate estimate.

Figure 6.2: Different data representations of the vehicle tracking algorithm. The left image shows 6D-Vision features. They serve as initialization input for the vehicle tracking process. The middle picture shows the Stixel representation that is used for improving the pose alignment of the tracked box. The final tracking result with the predicted driving corridor is shown on the right.

vehicle much better. The Stixels span the complete object and thus reach a substantial amount. Of course, with an increasing distance, the availability of Stixels decreases. Nevertheless, this method has proven to be reliable and robust.

Secondly, to profit from the Stixel representation for the tracked object in subsequent images (e.g. when the tracked object approaches), the L-shape information is also considered during the vehicle's motion estimation process. For this purpose, it is incorporated as direct measurement for updating the estimate of the vehicle's rotation point and orientation during the tracking phase. At this point, it is not expedient to go into detail without thoroughly explaining the motion model of the vehicle and the used Kalman filter design at the same time. Hence, please refer to the corresponding publications for more detail on that part [15, 16, 18, 19].

For testing the effectiveness of our approach, we compared the proposed initialization method to two alternative strategies, namely an initialization that uses a RADAR sen-

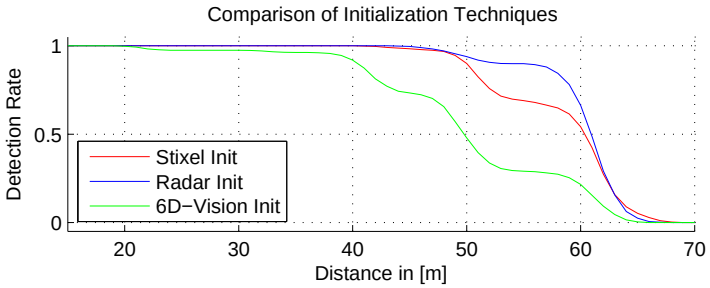


Figure 6.3: For the vehicle tracking to work reliably different methods have been tested, namely 6D-Vision, Radar, and Stixels. They are compared by means of the distance at which the tracker is successfully initialized. Thus, a higher distance value is equivalent to a better performance.

sor unit (Continental ARS300 [188]) as well as the original 6D-Vision [66, 176] based technique. To this end, robotic vehicles are used that are equipped with high-precision inertial measurement units and thus allow to obtain ground truth motion data (for details see Section 5.3.2). These vehicles are moving towards our position. Then, the distance is registered at which the stereo vision based vehicle tracker of Barth et al. is successfully initialized for the first time on these vehicles. For this evaluation a set of approximately 130 Sequences was used. The corresponding results are depicted in Figure 6.3.

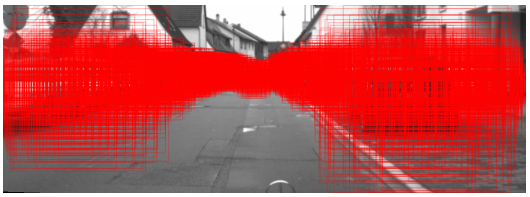
It is shown that the Stixel and the RADAR based scheme are equally matched (with slight advantages for the RADAR). According to this test, first tracker initializations are registered at a distance of about 65 m and 90 percent of all vehicles are detected at 50 m distance. Latest at about 40 m all vehicles have been successfully tracked when initializing the tracker with either Stixels or the RADAR sensor.

In this comparison, the 6D-Vision based initialization technique falls behind a little. Even though the first tracker initializations are also made at about 60 m, the 90 percent barrier is not reached until a distance of 40 m. Finally, for successfully tracking all vehicles, those even had to approach up to a distance of about 20 m.

6.2 Classification of Vehicles

In the following, a Stixel based approach for classifying front and rear sides of vehicles is presented. For this purpose, the Stixel representation is used in terms of control of interest for creating sets of two-dimensional box hypotheses. A typical selection of such views on vehicles is given in Figure 6.4. Compared to a straightforward stereo-based scheme for extracting regions of interest (ROIs), we show how using Stixels allows to noticeably improve the performance of the classifier. At the same time, the required computational effort is reduced significantly. The illustrative material used in the following is provided by courtesy of Matthias Hummel. Note that our approach is related to the work of Mitzel et al. [149] who have proposed a ROI generation scheme for a classifier with fixed time budget constraints.

Figure 6.4: A selection of rear and front views of cars. The objective of this work is to selectively create hypotheses for their efficient classification.



(a) ROIs computed using a stereo-based approach.

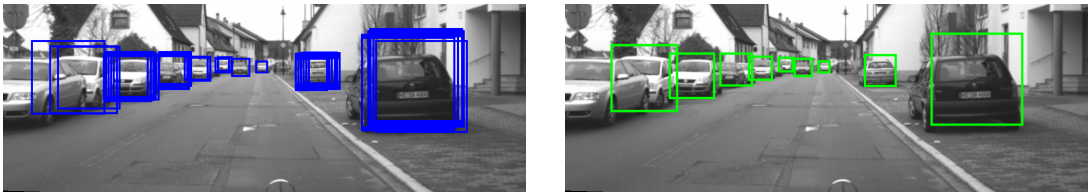
(b) Stixels are used for creating ROIs.

Figure 6.5: Two different methods for ROI generation are compared visually. The stereo-based scheme creates a significantly higher number of ROIs that are scattered throughout the scene. In contrast, the Stixel-based scheme is more selective in that manner. Even though relying on Stixels means testing significantly less hypotheses, our tests will show that using the smaller set allows to yield better classification results.

For the classification architecture we choose a combination of a neural net with local receptive fields (NN/LRF) [227]. The classification itself is taken care of by using a linear support vector machine (SVM) [47, 217]. Naturally, performing the actual classification step is computationally expensive. Therefore, it is our goal to keep the number of testing hypotheses as small as possible. Yet, of course, dismissing a lot of hypotheses in advance bears the risk to discard potentially good ones. That on the other hand can lead to an unsatisfactory detection rate.

A widespread approach for the ROI extraction task is to scatter box hypotheses across the scene while using stereo data to test for sufficient object evidence. Exemplary for this strategy, we rely on a scheme closely related to the technique proposed by Keller et al. [111]. To this end, a stereo-based weight is computed for each box in a correlation-like fashion. This weight is considered to decide whether that particular ROI is used for classification or not.

Alternatively, we suggest to follow a different strategy. The novelty in that context is that Stixels are used to decide where these hypotheses are to be spread and where not. For this purpose, both the position within the image as well as the depth of the Stixel are taken into account. Since the used classification scheme has proven to be sensitive to not having the vehicle aligned perfectly within the box (especially with respect to the lower end of the vehicle), every ROI is instantiated multiple times with a small vertical offset. A visual comparison of both techniques is shown in Figure 6.5. It clearly shows how the stereo-based ROIs are widespread while the Stixel-based ROIs are instantiated very selectively.



(a) Filtered classification result with 80% confidence. Partially, multiple ROIs match for the same object and therefore overlap.

(b) Result after applying non-maximum suppression to the raw classifier result. Mainly single boxes remain for every detected car.

Figure 6.6: The left figure shows the classification result of the ROIs. Typically, multiple boxes represent the same car. Thus, the boxes are fused using non-maximum suppression. The final classification result is shown on the right side.

Optionally for our method, instead of directly proceeding with the obtained Stixel ROIs, a more sophisticated filtering approach is chosen. Here the boxes are pre-selected by following two different basic strategies. For the first one, the real height values of all Stixels within the box are averaged. Given that the obtained value is above a certain threshold, that ROI is left out for the classification step. For the second test, the Stixels within the box are checked for left-to-right symmetry (which is related to the proceeding of [118] or [203]) and those that fail a given symmetry criterion are discarded for the classification as well.

Finally, the image along with all remaining ROIs is handed to the classifier. An exemplary classification result for the illustrated scenario is depicted in Figure 6.6a. In a post-processing step spatially overlapping boxes are fused by using non-maximum suppression. This step also allows to improve the localization precision of the box itself. The final result is shown in Figure 6.6b.

For evaluating the discussed ROI techniques, an urban sequence consisting of approximately $3,000$ frames was used. For obtaining ground truth information about the vehicles, the data set was annotated manually. The implementation of the classifier's processing pipeline is a variant of [56, 179]. Note that the required computational effort increases linearly with the size of the ROI set.

The average number of ROIs generated per frame when using the different ROI extraction techniques is compared in Table 6.1. Accordingly, for the Stixel-based ROI extraction scheme, the number of considered ROIs is up to 16 times smaller, which consequently decreases the computational effort by the same factor. Details on the underlying evaluation strategy are found in [55].

The actual classifier performance for the different configurations is illustrated in Figure 6.7 in form of its receiver operating characteristic (ROC). Apparently, all Stixel-based approaches outperform the stereo-based ROI generation method indisputably. Given a detection rate of 90%, using Stixels with heights and symmetry filtering yields a false positive rate of approximately 0.025 occurrences per frame. The stereo-based ROI variant, on the other hand, yields about 0.3 false positives per frame. Thus, the Stixel-based scheme performs about 10 times better than the stereo-based ROI approach. Alterna-

ROI Extraction Method	Average Number of ROI
plain stereo	5897
Stixel only	1084
Stixel + Height	514
Stixel + Symmetry	594
Stixel + Height + Symmetry	370

Table 6.1: The number of ROIs directly determines the computational effort of the classifier. Thus, the average number of ROIs generated per frame using the different techniques is given in a comparing overview. Using Stixels helps to reduce their total number significantly.

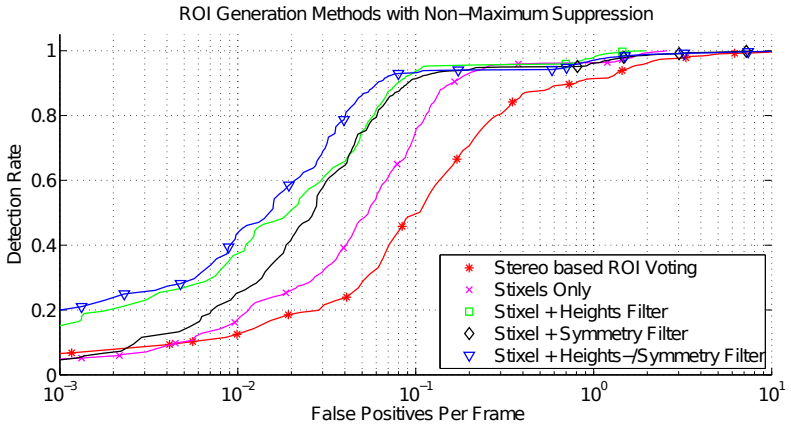


Figure 6.7: Receiver operating characteristic (ROC) for the different ROI generation techniques. Apparently, he reduction of the number of hypotheses does not lead to a performance breakdown of the classifier. Instead, using Stixels for ROI extraction allows to increase the classifier's performance up to a factor of ten. Simultaneously, the computational effort is reduced significantly (for details refer to Table 6.1).

tively, the other way around, when aiming for a false positive rate of 0.1 per frame, the Stixel-based approach yields a detection rate of 94% while the stereo-based ROI scheme only reaches 49%. Given that the set of hypotheses has been reduced significantly, this is a very promising result.

In comparison, the other methods (i.e. Stixel ROI only, Stixel ROI with height filtering as well as Stixel ROI with symmetry filtering) lie in between the other two techniques. That relates to both their classifying performance and the required computational effort. In total, the obtained results are very promising.

6.3 Segmentation and Reasoning

As discussed in the evaluation Section 5.3, a reliable estimation of both moving objects and stationary environment is a challenging task. Occasionally, Stixels representing static obstacles are estimated as moving and vice versa. These difficulties become particularly visible for regular structures, such as for those that exhibit repetitive patterns or are in parts not textured at all (e.g. reinforced concrete walls).

However, even if the motion estimation is successful, the obtained representation by default does not evince semantic relations between the individual Stixels. Thus, deciding which of those actually belong to the same moving object and which belong to stationary background requires further processing of these data.

To tackle that issue, Erbs et al. [57] have shown a probabilistic approach to detect and segment the current three-dimensional scene into arbitrary moving objects and static background. For this purpose, the authors rely on the dynamic Stixel World and thus use the Stixel motion estimate directly as measurement input. The objective of the segmentation is formulated as a maximum a posteriori estimation problem. Their approach is not limited to a certain or a priori known number of objects. To yield efficiency, dynamic programming [22] is used for the actual solving step and for extracting the most probable segmentation of the dynamic Stixels.

To cope with noise and errors in the measurement data, Erbs et al. incorporate a set of regularization priors and geometric constraints to stabilize the segmentation process and to increase the robustness of their method. These priors are either given in reference to spatial properties, the Stixel motion state or relate to appearance aspects.

For instance, similar to the vehicle tracking approach discussed in Section 6.1, in their basic shape, vehicles are assumed as box-like. Hence, the authors probe for L-shapes in the Stixel representation. Deviations from that shape are penalized, and a corresponding geometric likelihood is extracted for every tested configuration.

Further, all Stixels within a segmented group are supposed to have similar motion properties. Accordingly, that aspect is evaluated and considered an indication for the likelihood of Stixels to belong to the same group or not.

Thirdly, a meaningful measure is the so called *existence feature*. It is used to rate the likelihood of certain Stixel groups depending on the Stixel density. The more sparse a potential object cluster is covered with Stixels, the more is that particular configuration considered as unlikely and vice versa.

Exemplary results of the discussed method are depicted in Figure 6.8. These illustrations have been provided by courtesy of Friedrich Erbs. The orange boxes denote segmented groups of moving Stixels. Everything that is not framed within a box is considered as stationary background. Note how this method is able to cope with noise in the input data. For instance, look at the small wall in the left image or at the guard rail and bushes on the right side.

Additionally, within the scope of their contribution, the authors have validated their approach in comparison to a mean shift based clustering scheme [44]. For further details please refer to the corresponding publication [57].

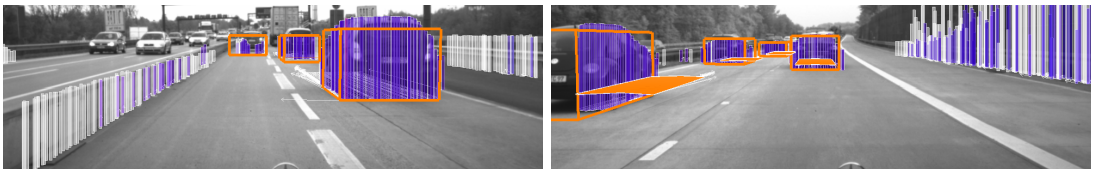


Figure 6.8: Segmentation of Stixels into moving objects (orange boxes) and stationary background according to the probabilistic method provided by Friedrich Erbs. Geometric constraints and smoothness priors allow to successfully cope with noisy and faulty data, such as shown on the crash barrier or the guard rail.

7 Conclusions and Outlook

In this doctoral thesis, a novel scheme for computing a medium-level representation called the Stixel World was presented. The proposed technique allows to model three-dimensional environments in an exceptionally compact and flexible manner. It gives direct access to essential characteristics of other objects, such as their position and motion states, and can be tagged with further useful meta information. The reliability of the extracted data was evaluated in diverse environments. For these reasons, we consider the Stixel World as ideally suited to be the basic building block for future vision systems.

Our Contribution Within the scope of this thesis we presented a novel technique for computing the Stixel World in a probabilistic and unified scheme. Large amounts of input data, such as obtained from dense stereo matching, are processed in real-time. In this process, the three-dimensional data is interpreted with regard to freespace and object information and thus is condensed within a few but highly expressive scene descriptors.

The original Stixel primitive was extended accordingly for modeling staggered objects within the same row of an image and thus gives more flexibility for describing three-dimensional environments in an extraordinary compact and precise fashion.

The proposed computation method relies on optimization and, in the presence of manifold physically motivated world model priors such as ordering and gravity constraints, allows to yield the column-wise optimal representation for the given scenario.

Beyond that, the suggested mathematical approach is free of hard-coded thresholds. Instead, adapting to new data sources and different environmental conditions is achieved by straightforwardly characterizing the particular sensor model in detail using plausible and realistic sensor parameters.

Furthermore, to evaluate temporal coherences between consecutive images novel tracking techniques were presented. Thus, in addition to spatial information, the motion state of the captured objects is extracted and prepared for further processing. Thanks to the well-engineered 6D-Vision principle, precise motion information of all represented objects is available for every Stixel independently.

In addition, we suggested different techniques for measuring the two-dimensional image displacement of Stixel features between different time steps. In this context, a new variant of the well-known KLT-tracker called patch-KLT was proposed that offers two-dimensional scale estimation with disparity-based pixel weighting.

We outlined thorough evaluation strategies for all proposed algorithms and discussed the achieved results. Among others, we also investigated the measurement accuracy and tracking precision. Reference sensors as well as straightforward comparative methods were used in both controlled environments and real-world scenarios.

Lessons Learned For the performed evaluation of the Stixel measurement accuracy we made use of high-precision LIDAR technology. Hence, it was demonstrated that using the combination of stereo vision and Stixels for the representation of three-dimensional environments allows to yield highly accurate depth information.

In terms of robustness, the newly proposed scheme is outperformed by the original occupancy grid based approach. However, the new scheme offers significantly more possibilities and, thanks to its higher sensitivity, achieves a much better visual range and coverage in perception.

The quality of the motion estimation has been evaluated in the context of different test runs that took place within manifold environments. For that task, we used GPS-based military-grade inertial measurement units, RADAR technology as well as standard OEM vehicle motion sensors. Thus, we gave an extensive overview about the performance of the whole motion tracking system. Altogether, the obtained results are very promising.

With regards to rating the measurement accuracy of the two-dimensional image displacements, the different optical flow schemes were successfully tested in real-world environments. For making the actual differences in performance more apparent, we also tested within well selected, highly demanding environmental setups. It became clear that the sketched patch-KLT method massively leverages available texture information. In addition it was found that the scheme enables diligent incorporation of further sensor information, such as depth from stereo, ego-motion information and the foreign motion properties of other tracked objects. Certainly, that scheme is costly to compute, but it helps to improve the quality of the flow estimation noticeably. Using modern graphics processors allows for an efficient, near real-time implementation.

We also sketched examples for a direct application of Stixels into other subsequent vision task. These render the true potential of the Stixel World to be highly beneficial for scene representation not only in reference to driver assistance, but also in various other domains such as robotics, geographic information systems or computer graphics.

Specifically with respect to applications with limited computational resources, we consider the Stixel World an ideal spatio-temporal scene representation to efficiently bridge the gap between single pixels and the object level. That is especially true if multiple independent processes demand to access the environmental scene data at the same time. Naturally, in advanced driver assistance, that configuration is the use case for the architecture of modern vision systems.

Outlook The proposed Stixel computation scheme is fairly young and offers a lot of room for improvement. In this context, one of the most apparent aspects concerns the requirement for horizontal smoothness of the object segmentation that was put aside for computational reasons. Comparable approaches, such as proposed by Felzenszwalb et al. [63] or Badino et al. [9, 10], have shown the usefulness when taking such priors into account. However, their objectives were in that effect simple, as they did not have to resolve any correspondence problems within the smoothing process.

Anyway, for the Stixel World as proposed in this thesis, we are convinced that a similar effect must be possible. If applicable, we believe that an iterative computation approach could lead the way for this purpose.

Secondly and not less significant, it is desirable to extend the proposed tracking scheme to support the multi-layer object representation intrinsically. In the same breath, it would be utmost beneficial to find a methodologically correct way (other than re-sampling) to maintain the regular grid structure for the Stixel representation during the tracking phase. That way, the problem of Stixel holes to gash on objects would be avoided. Also, a regular grid would considerably facilitate the work of subsequent processing steps. Accordingly, these aspects will definitely remain future work.

Furthermore, the incorporation of multiple sensor data remains an interesting and meaningful topic. Also, the extension of the proposed algorithms to large viewing angles (such as 90 degree field of view and higher) will certainly have one or the other challenging pitfall in store.

Last but not least, assigning semantic information to the Stixel World remains a promising objective. For this purpose, we presented a few examples, such as the vehicle classification or the grouping of Stixels into moving objects and stationary background. Further, we are positive that for a wide and successful application of the Stixel representation the availability of class assignments (e.g. for *vehicle*, *pedestrian*, *traffic sign* or similar) is a highly useful feature to consider and will have the potential to support subsequent vision tasks considerably.

Bibliography

[1] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *International Conference on Computer Vision (ICCV)*, pages 72–79, Kyoto, Japan, September 2009.

[2] Hadi Aliakbarpour, João Filipe Ferreira, Kamrad Khoshhal, and Jorge Dias. A novel framework for data registration and data fusion in presence of multi-modal sensors. In *Doctoral Conference on Computing, Electrical and Industrial Systems (DoCEIS)*, pages 308–315, 2010.

[3] John Aloimonos and Michael Swain. Shape from patterns: Regularization. *International Journal of Computer Vision (IJCV)*, 2(2):171–187, 1988.

[4] Franz Andert and Lukas Goormann. A fast and small 3-d obstacle model for autonomous applications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 2883–2889, 2008.

[5] David L. Andrews. *Structured light and its applications: An introduction to phase-structured beams and nanoscale optical forces.* Academic Press, 2008.

[6] Alexander Bachmann. *Dichte Objektsegmentierung in Stereobildfolgen.* KIT Scientific Publishing, 2010.

[7] Hernán Badino. A robust approach for ego-motion estimation using a mobile stereo platform. In 1^{st} *International Workshop on Complex Motion, IWCM*, Günzburg, Germany, October 2004. Springer.

[8] Hernán Badino, Uwe Franke, and Rolf Mester. Free space computation using stochastic occupancy grids and dynamic programming. In *Workshop on Dynamical Vision, ICCV*, Rio de Janeiro, Brazil, October 2007.

[9] Hernán Badino, Uwe Franke, and David Pfeiffer. The Stixel World - A compact medium level representation of the 3D-world. In *German Association for Pattern Recognition (DAGM)*, pages 51–60, Jena, Germany, September 2009.

[10] Hernán Badino, Rudolf Mester, Tobi Vaudrey, and Uwe Franke. Stereo-based free space computation in complex traffic scenarios. In *Image Analysis and Interpretation*, pages 189–192, 2008.

[11] Caroline Baillard and Andrew Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2559–2565, Ft. Collins, CO, USA, June 1999.

[12] Caroline Baillard and Andrew Zisserman. A plane-sweep strategy for the 3D reconstruction of buildings from multiple images. In *International Society for Photogrammetry and Remote Sensing (ISPRS)*, Amsterdam, 2000.

[13] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, Inc., New York, NY, USA, 2002.

[14] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transaction on Graphics*, 28(3):1–10, 2009.

[15] Alexander Barth. *Vehicle Tracking and Motion Estimation Based on Stereo Vision Sequences*. PhD thesis, Friedrich-Wilhelms-Universitaet zu Bonn, September 2010.

[16] Alexander Barth and Uwe Franke. Where will the oncoming vehicle be the next second? In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1068–1073, Eindhoven, Netherlands, April 2008.

[17] Alexander Barth and Uwe Franke. Tracking oncoming and turning vehicles at intersections. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 861–868, Madeira Island, Portugal, September 2010.

[18] Alexander Barth, David Pfeiffer, and Uwe Franke. Vehicle tracking at urban intersections using dense stereo. In 3^{rd} *Workshop on Behaviour Monitoring and Interpretation, BMI*, pages 47–58, Ghent, Belgium, November 2009.

[19] Alexander Barth, Jan Siegemund, Uwe Franke, and Wolfgang Förstner. Simultaneous estimation of pose and motion at highly dynamic turn maneuvers. In *German Association for Pattern Recognition (DAGM)*, pages 262–271, Jena, Germany, September 2009. Springer.

[20] Alexander Barth, Jan Siegemund, Annemarie Meißner, Uwe Franke, and Wolfgang Förstner. Probabilistic multi-class scene flow segmentation for traffic scenes. In *German Association for Pattern Recognition (DAGM)*, pages 503–512, Darmstadt, Germany, September 2010.

[21] Paul A. Beardsley, Philip H. S. Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In *European Conference on Computer Vision (ECCV)*, pages 683–695, Cambridge, UK, April 1996.

[22] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 6(60):503–515, 1954.

[23] Massimo Bertozzi, Luca Bombini, Alberto Broggi, Michele Buzzoni, Elena Cardarelli, Stefano Cattani, Pietro Cerri, Alessandro Coati, Stefano Debattisti, Andrea Falzoni, Rean Isabella Fedriga, Mirko Felisa, Luca Gatti, Alessandro Giacomazzo, Paolo Grisleri, Maria Chiara Laghi, Luca Mazzei, Paolo Medici, Matteo Panciroli, Pier Paolo Porta, Paolo Zani, and Pietro Versari. Viac: an out of ordinary experiment. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 175–180, Baden-Baden, Germany, June 2011.

[24] Stan Birchfield and Carlo Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *International Conference on Computer Vision (ICCV)*, pages 489–495, Kerkyra, Greece, September 1999.

[25] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[26] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[27] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatchstereo - stereo matching with slanted support windows. In *British Machine Vision Conference (BMVC)*, Dundee, Scotland, September 2011. BMVA Press.

[28] Helk A. P. Bloom and Yaakov Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33:780–783, 1988.

[29] BMW. BMW Group, München, Germany. `http://www.bmw.com/`, August 2011.

[30] Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little ben: The ben franklin racing team's entry in the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

[31] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. In *International Conference on Computer Vision (ICCV)*, pages 377–384, Kerkyra, Corfu, Greece, September 1999.

[32] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239, 2001.

[33] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. A three resolution framework for reliable road obstacle detection using stereovision. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3932–3938, Anchorage, AK, USA, May 2010.

[34] Gabriel J. Brostow and Irfan A. Essa. Image-based motion blur for stop motion animation. In *SIGGRAPH*, pages 561–566, 2001.

[35] Matthew Brown and David G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *International Conference on 3D Digital Imaging and Modeling*, pages 56–63, Ottawa, Ontario, Canada, June 2005.

[36] Myron Z. Brown, Darius Burschka, and Gregory D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):993–1008, 2003.

[37] Thomas Brox, Bodo Rosenhahn, and Joachim Weickert. Three-dimensional shape knowledge for joint image segmentation and pose estimation. In *German Association for Pattern Recognition (DAGM)*, pages 109–116, Vienna, Austria, September 2005.

[38] Thomas Brox, Mikaël Rousson, Rachid Deriche, and Joachim Weickert. Colour, texture, and motion in level set based segmentation and tracking. *Image Vision Comput.*, 28:376–390, March 2010.

[39] Thomas Brox and Joachim Weickert. Nonlinear matrix diffusion for optic flow estimation. In *German Association for Pattern Recognition (DAGM)*, pages 446–453, Zürich, Switzerland, September 2002.

[40] Ping Chen and Shunlong Luo. Direct approach to quantum extensions of fisher information. *Frontiers of Mathematics in China*, 2:359–381, 2007. 10.1007/s11464-007-0023-4.

[41] Yi-Liang Chen, Venkataraman Sundareswaran, Craig Anderson, Alberto Broggi, Paolo Grisleri, Pier Paolo Porta, Paolo Zani, and John Beck. Terramax: Team oshkosh urban robot. *Journal of Field Robotics*, 25(10):841–860, 2008.

[42] Wei-Chen Chiu, Ulf Blanke, and Mario Fritz. Improving the kinect by cross-modal stereo. In *British Machine Vision Conference (BMVC)*, Dundee, Scotland, September 2011. BMVA Press.

[43] Robert Collins. A space-sweep approach to true multi-image matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 358–363, San Francisco, CA, USA, June 1996.

[44] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(5):603–619, 2002.

[45] Richard W. Conners, Mohan M. Trivedi, and Charles A. Harlow. Segmentation of a high-resolution urban scene using texture operators. *Graphical Models /graphical Models and Image Processing /computer Vision, Graphics, and Image Processing*, 25:273–310, 1984.

[46] Nico Cornelis, Bastian Leibe, Kurt Cornelis, and Luc J. Van Gool. 3d city modeling using cognitive loops. In 3^{rd} *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2006)*, pages 9–16, Chapel Hill, NC, USA, June 2006. IEEE Computer Society.

[47] Corinna Cortes and Vladimir N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[48] Ingemar J. Cox, Sunita L. Hingorani, Satish Rao, and Bruce M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[49] Daniel Cremers, Mikaël Rousson, and Rachid Deriche. A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape. *International Journal of Computer Vision (IJCV)*, 72(2):195–215, 2007.

[50] Daimler. Daimler AG, Stuttgart, Germany. `http://www.daimler.com/`, August 2011.

[51] Radu Danescu, Florian Oniga, and Sergiu Nedevschi. Particle grid tracking system for stereovision based environment perception. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 987–992, San Diego, CA, USA, June 2010.

[52] Jennifer Dolson, Jongmin Baek, Christian Plagemann, and Sebastian Thrun. Upsampling range data in dynamic environment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, June 2010.

[53] Arnaud Doucet, Nando De Freitas, and Neil Gordon. *Sequential Monte Carlo methods in practice*. Springer, 2001.

[54] Alberto E. Elfes. Sonar-based real-world mapping and navigation. *Journal of Robotics and Automation*, 3(3):249–265, June 1987.

[55] Markus Enzweiler and Dariu Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(12):2179–2195, 2009.

[56] Markus Enzweiler and Dariu M. Gavrila. A multi-level mixture-of-experts framework for pedestrian classification. *IEEE Transactions on Image Processing*, 1(1):1–13, DOI: 10.1109/TIP.2011.2142006, 2011.

[57] Friedrich Erbs, Alexander Barth, and Uwe Franke. Moving vehicle detection by optimal segmentation of the dynamic Stixel World. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 951–956, Baden-Baden, Germany, June 2011.

[58] Ines Ernst and Heiko Hirschmüller. Mutual information based semi-global stereo matching on the gpu. In *Advances in Visual Computing, 4^{th} International Symposium, ISVC*, pages 228–239, Las Vegas, NV, USA, December 2008.

[59] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc J. Van Gool. Moving obstacle detection in highly dynamic scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 56–63, 2009.

[60] Andreas Ess, Tobias Mueller, Helmut Grabner, and Luc J. Van Gool. Segmentation-based urban traffic scene understanding. In *British Machine Vision Conference (BMVC)*, London, UK, September 2009. BMVA Press.

[61] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004.

[62] Pedro F. Felzenszwalb, Gyula Pap, Éva Tardos, and Ramin Zabih. Globally optimal pixel labeling algorithms for tree metrics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3153–3160, San Francisco, CA, USA, June 2010.

[63] Pedro F. Felzenszwalb and Olga Veksler. Tiered scene labeling with dynamic programming. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3097–3104, San Francisco, CA, USA, June 2010.

[64] David A. Forsyth, Okan Arikan, Leslie Ikemoto, James F. O'Brien, and Deva Ramanan. Computational studies of human motion: Part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2/3):79–254, 2005.

[65] Jan-Michael Frahm, Pierre Fite Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, and Svetlana Lazebnik. Building rome on a cloudless day. In *European Conference on Computer Vision (ECCV)*, pages 368–381, Heraklion, Crete, Greece, September 2010.

[66] Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan Gehrig. 6d-vision: Fusion of stereo and motion for robust environment perception. In *German Association for Pattern Recognition (DAGM)*, Vienna, Austria, September 2005.

[67] Friedrich Fraundorfer, Davide Scaramuzza, and Marc Pollefeys. A constricted bundle adjustment parameterization for relative scale estimation in visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1899–1904, Anchorage, Alaska, USA, May 2010.

[68] Chiara Fulgenzi, Anne Spalanzani, and Christian Laugier. Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1610–1616, Rome, Italy, April 2007.

[69] Michael Fussenegger, Rachid Deriche, and Axel Pinz. Multiregion level set tracking with transformation invariant shape priors. In *Asien Conferenceon Computer Vision (ACCV)*, pages I:674–683, 2006.

[70] David Gallup, Jan-Michael Frahm, Philippos Mordohai, Qingxiong Yang, and Marc Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, USA, June 2007.

[71] David Gallup, Marc Pollefeys, and Jan-Michael Frahm. 3d reconstruction using an n-layer heightmap. In *German Association for Pattern Recognition (DAGM)*, pages 1–10, Darmstadt, Germany, September 2010.

[72] Juliana Gambini, Damian Rozichner, María E. Buemi, Marta Mejail, and Julio Jacobo Berllés. Occlusion handling for object tracking using a fast level set method. In *Proceedings of the 2008 XXI Brazilian Symposium on Computer Graphics and Image Processing*, pages 61–68, Washington, DC, USA, 2008. IEEE Computer Society.

[73] Stefan Gehrig, Felix Eberli, and Thomas Meyer. A real-time low-power stereo vision engine using semi-global matching. In *International Conference on Computer Vision Systems (ICVS)*, Liège, Belgium, October 2009. Springer-Verlag.

[74] Stefan Gehrig and Clemens Rabe. Real-Time Semi-Global Matching on the CPU. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Workshops*, pages 85–92, San Francisco, CA, USA, June 2010.

[75] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, Baden-Baden, Germany, June 2011.

[76] Minglun Gong, Ruigang Yang, Liang Wang 0002, and Mingwei Gong. A performance study on different cost aggregation approaches used in real-time stereo matching. *International Journal of Computer Vision*, 75(2):283–296, 2007.

[77] Robert M. Gray. *Entropy and information theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[78] D. M. Greig, B. T. Porteous, and Allan H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society Series B*, 51:271–279, 1989.

[79] Rémi Gribonval. Should penalized least squares regression be interpreted as maximum a posteriori estimation? *IEEE Transactions on Signal Processing (TSP)*, 59(5):2405–2410, 2011.

[80] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision (ECCV)*, pages 482–496, Heraklion, Crete, Greece, September 2010.

[81] Martin Habbecke and Leif Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, USA, June 2007.

[82] Istvan Haller and Sergiu Nedevschi. Gpu optimization of the sgm stereo algorithm. In *Proceedings of the Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, IEEE International Conference on Computational Photography (ICCP), pages 197–202, Washington, DC, USA, 2010.

[83] Klaus Häming and Gabriele Peters. The structure-from-motion reconstruction pipeline - a survey with focus on short image sequences. *Kybernetika*, 46(5):926–937, 2010.

[84] Chris Harris and Mike Stephens. *A combined corner and edge detector*, volume 15, pages 147–151. Manchester, UK, 1988.

[85] Velodyne Headquarters. High Definition Lidar HDL-64E S2. `http://www.velodyne.com/lidar/`, February 2010.

[86] Velodyne Headquarters. Image Sequence Analysis Test Site (EISATS) Set 6. `http://www.mi.auckland.ac.nz/`, August 2011.

[87] Anna Hilsmann, David Schneider, and Peter Eisert. Template-free shape from texture with perspective cameras. In *British Machine Vision Conference (BMVC)*, Dundee, Scotland, September 2011. BMVA Press.

[88] Heiko Hirschmüller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 807–814, San Diego, CA, USA, June 2005.

[89] Heiko Hirschmüller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(9):1582–1599, 2009.

[90] Bernd Höfflinger. *EUREKA Verbundprojekt PROMETHEUS: Teilprojekt PRO-CHIP Custom Hardware for Intelligent Processing ; Schlußbericht für die Definitionsphase in der Zeit vom 1. Oktober 1986 bis 30. September 1987 ; Contract TV 8627 F6 ; TV 8727 F3*. 1988.

[91] Michael Hofmann and Dariu M. Gavrila. Multi-view 3d human pose estimation combining single-frame recovery, temporal integration and model adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2214–2221, Miami, Florida, USA, June 2009.

[92] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *International Conference on Computer Vision (ICCV)*, pages 654–661, 2005.

[93] Derek Hoiem, Carsten Rother, and John M. Winn. 3d layoutcrf for multi-view object class recognition and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, USA, June 2007.

[94] Florian Homm, Nico Kaempchen, Jeff Ota, and Darius Burschka. Efficient occupancy grid computation on the gpu with lidar and radar for road boundary detection. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1006–1013, San Diego, CA, USA, June 2010.

[95] Esther Horbert, Dennis Mitzel, and Bastian Leibe. Geometrically constrained level set tracking for automotive applications. In *DAGM-Symposium*, pages 472–482, Darmstadt, Germany, September 2010.

[96] Berthold K. P. Horn and Michael J. Brooks. The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208, 1986.

[97] Berthold K. P. Horn and B. G. Schunk. Determining optical flow. In *Artificial Intelligence*, volume 17, pages 185–203, 1981.

[98] Jan Horn, Alexander Bachmann, and Thao Dang. Stereo vision based ego motion estimation with sensor supported subset validation. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 741–748, 2007.

[99] Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 3946–3952, Nice, France, September 2008.

[100] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *International Conference on Computer Vision (ICCV)*, pages 1–7, Rio de Janeiro, Brasil, October 2007.

[101] Martin Humenberger, Daniel Hartermann, and Wilfried Kubinger. Evaluation of stereo matching systems for real world applications using structured light for ground truth estimation. In *IAPR Conference on Machine Vision Applications (MVA)*, pages 433–436, 2007.

[102] Karl Iagnemma, Ron Kurjanowicz, and Martin Buehler. Journal of Field Robotics - Special issue on the DARPA Grand Challenge part I. *Journal of Field Robotics*, 23(8):461–623, 2006.

[103] Karl Iagnemma, Ron Kurjanowicz, and Martin Buehler. Journal of Field Robotics - Special issue on the DARPA Grand Challenge part II. *Journal of Field Robotics*, 23(9):655–835, 2006.

[104] Emmanuel C. Ifeachor and Barrie W. Jervis. *Digital Signal Processing: A Practical Approach*. Addison-Wesley Publishing Company, Wokingham, England, 1993.

[105] Katsushi Ikeuchi. Shape from regular patterns. *Journal of Artificial Intelligence*, 22(1):49–75, 1984.

[106] Katsushi Ikeuchi and Berthold K. P. Horn. Numerical shape from shading and occluding boundaries. *Journal of Artificial Intelligence*, 17(1-3):141–184, 1981.

[107] iMAR Navigation. iTraceRT-F200. `http://www.imar-navigation.de/`, August 2011.

[108] Sören Kammel, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagzent, Joachim Schröder, Michael Thuy, Matthias Goebl, Felix von Hundelshausen, Oliver Pink, Christian Frese, and Christoph Stiller. Team annieway's autonomous system for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

[109] Nico Kämpchen, Thorsten Weiss, Michael Schaefer, and Klaus Dietmayer. Imm object tracking for high dynamic driving maneuvers. In *IEEE Intelligent Vehicles Symposium (IV)*, Parma, Italy, July 2004.

[110] Steven M. Kay. *Fundamentals of statistical signal processing: estimation theory*. Number v. 2 in Prentice Hall signal processing series. PTR Prentice-Hall, 1993.

[111] Christoph Keller, Markus Enzweiler, Marcus Rohrbach, David Fernández Llorca, Christoph Schnörr, and Dariu M. Gavrila. The benefits of dense stereo for pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–11, 2011.

[112] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 486–492, San Diego, CA, USA, June 2010.

[113] Reinhard Koch, Marc Pollefeys, and Luc J. Van Gool. Realistic surface reconstruction of 3d scenes from uncalibrated image sequences. *Journal of Visualization and Computer Animation*, 11(3):115–127, 2000.

[114] Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision (IJCV)*, 82(3):302–324, 2009.

[115] Vladimir Kolmogorov and Carsten Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(7):1274–1279, 2007.

[116] Jana Košecká and Wei Zhang. Video compass. In *European Conference on Computer Vision (ECCV)*, pages 476–490, Copenhagen, Denmark, May 2002.

[117] Kai Krajsek, Rudolf Mester, and Hanno Scharr. Statistically optimal averaging for image restoration and optical flow estimation. In *German Association for Pattern Recognition (DAGM)*, pages 466–475, Munich, Germany, June 2008.

[118] Andreas Kuehnle. Symmetry-based recognition of vehicle rears. *Pattern Recogn. Lett.*, 12:249–258, April 1991.

[119] In So Kweon and Takeo Kanade. High-resolution terrain map from multiple sensor data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14:278–292, 1992.

[120] Raphael Labayrade, Didier Aubert, and Jean-Philippe Tarel. Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation. In *IEEE Intelligent Vehicles Symposium (IV)*, 2002.

[121] Simon Lacroix, Il kyun Jung, and Anthony Mallet. Digital elevation map building from low altitude stereo imagery. In *International Symposium on Intelligent Robotic Systems*, 2001.

[122] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data.* 18. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[123] Henning Lategahn, Andreas Geiger, and Bernd Kitt. Visual slam for autonomous ground vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[124] Henning Lategahn, Thorsten Graf, Carsten Hasberg, Bernd Kitt, and Jan Effertz. Mapping in dynamic environments using stereo vision. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 150–156, Baden-Baden, Germany, June 2011.

[125] Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Mav visual slam with plane constraint. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3139–3144, Shanghai, China, May 2011.

[126] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc J. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, Minnesota, June 2007.

[127] Bastian Leibe, Konrad Schindler, Nico Cornelis, and Luc J. Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(10):1683–1698, 2008.

[128] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision (IJCV)*, 74(3):343–364, 2007.

[129] Vincent Lepetit and Pascal Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends(R) in Computer Graphics and Vision*, 1(1):1–89, 2005.

[130] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Accurate non-iterative o(n) solution to the pnp problem. In *International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, October 2007.

[131] Anat Levin and Richard Szeliski. Visual odometry and map correlation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 611–618, Washington, DC, USA, June 2004.

[132] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J. Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, Michael Sokolsky, Ganymed Stanek, David Stavens, Alex Teichman, Moritz Werling, and Sebastian Thrun. Towards fully autonomous driving: Systems and algorithms. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 945–950, Baden-Baden, Germany, June 2011.

[133] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4372–4378, Anchorage, AK, USA, May 2010.

[134] Benson Limketkai, Rahul Biswas, and Sebastian Thrun. Learning occupancy grids of non-stationary objects with mobile robots. In *Experimental Robotics VIII [ISER 2002, Sant'Angelo d'Ischia, Italy, 8-11 July 2002]*, volume 5, pages 222–231. Springer, 2002.

[135] Xiaoqing Liu, Olga Veksler, and Jagath Samarabandu. Order-preserving moves for graph-cut-based optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(7):1182–1196, 2010.

[136] Heidi Loose, Uwe Franke, and Christoph Stiller. Kalman particle filter for lane recognition on rural roads. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 60–65, Xi'an, Shaanxi, China, June 2009.

[137] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the Seventh International Joint Conference on Artifical Intelligence, IJCAI 1981*, pages 674–679, Vancouver, Canada, 1981.

[138] David J. C. Mackay. *Information theory, inference, and learning algorithms.* Cambridge University Press, Cambridge, 2003.

[139] Richard Mann and Michael S. Langer. Optical snow and the aperture problem. *International Conference on Pattern Recognition (ICPR)*, 4:40264, 2002.

[140] Michael T. Manry, Cheng Hsiung Hsieh, Michael S. Dawson, Adrian K. Fung, and Steven J. Apollo. Cramer rao maximum a-posteriori bounds on neural network training error for non-gaussian signals and parameters. *International Journal of Intelligent Control and Systems*, 1:381–391, 1996.

[141] Richard Mason, Jim Radford, Robert Walters, David Caldwell, Bill Caldwell, and Dmitriy Kogan. DARPA Urban Challenge. Technical report, The Golem Group LLC, April 2007.

[142] Norman Mattern, Robin Schubert, and Gerd Wanielik. High-accurate vehicle localization using digital maps and coherency images. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 462–469, San Diego, CA, USA, June 2010.

[143] Simon Bakerand Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, 56:221–255, February 2004.

[144] Larry Matthies and Alberto E. Elfes. Integration of sonar and stereo range data using a grid-based representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 727–733, April 1988.

[145] E. Mazor, Amir Averbuch, Yaakov Bar-Shalom, and Joshua Dayan. Interacting multiple model methods in target tracking: a survey. *IEEE Transactions on Aerospace and Electronic Systems*, 34(1):103–123, August 2002.

[146] Vicente Milanés, Enrique Onieva, Joshué Pérez, Jorge Godoy, and Jorge Villagrá. An approach to driverless vehicles in highways. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, Washington, DC, USA, October 2011.

[147] Wided Miled, Jean Christophe Pesquet, and Michael Parent. Disparity map estimation using a total variation bound. In *Canadian Conference on Computer and Robot Vision (CRV)*, pages 48–48, 2006.

[148] Isaac Miller, Mark Campbell, Dan Huttenlocher, Frank-Robert Kline, Aaron Nathan, Sergei Lupashin, Jason Catlin, Brian Schimpf, Pete Moran, Noah Zych, Ephrahim Garcia, Mike Kurdziel, and Hikaru Fujishima. Team cornell's skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527, 2008.

[149] Dennis Mitzel, Patrick Sudowe, and Bastian Leibe. Real-time multi-person tracking with time-constrained detection. In *British Machine Vision Conference (BMVC)*, Dundee, Scotland, September 2011. BMVA Press.

[150] Branislav Mičušík and Jana Košecká. Piecewise planar city 3d modeling from street view panoramic sequences. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 0:2906–2912, 2009.

[151] Branislav Mičušík and Jana Košecká. Semantic segmentation of street scenes by superpixel co-occurrence and 3d geometry. In *IEEE Workshop on Video-Oriented Object and Event Classification (VOEC)*, 2009.

[152] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Journal on Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

[153] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[154] Sandino Morales and Reinhard Klette. Ground truth evaluation of stereo algorithms for real world applications. In *IEEE Workshop on Computer Vision in Vehicle Technology:From Earth to Mars in conjunction with the European Conference on Computer Vision (ECCV)*, pages 152–162, Queenstown, New Zealand, November 2010.

[155] Sandino Morales, Tobi Vaudrey, and Reinhard Klette. Robustness evaluation of stereo algorithms on long stereo sequences. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 347 – 352, 2009.

[156] Hans P. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, Carnegie Mellon University, 1996.

[157] Hans P. Moravec and Alberto E. Elfes. High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 116 – 121, March 1985.

[158] Karsten Mühlmann, Dennis Maier, Jürgen Hesser, and Reinhard Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision (IJCV)*, 47(1-3):79–88, 2002.

[159] Thomas Müller, Clemens Rabe, and Uwe Franke. Dense6D: Position und Bewegung robust an jedem Bildpunkt. In *7. Workshop zur Fahrerassistenz*, 2011.

[160] Thomas Müller, Clemens Rabe, Jens Rannacher, Uwe Franke, and Rudolf Mester. Dense optical flow for illumination-robust motion estimation in real scenes using census signatures. In *German Association for Pattern Recognition (DAGM)*, Frankfurt, Germany, September 2011.

[161] Thomas Müller, Jens Rannacher, Clemens Rabe, and Uwe Franke. Feature and depth-supported modified total variation optical flow for 3d motion field estimation in real scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1193–1200, Colorado Springs, CO, USA, June 2011.

[162] Michael Munz, Mirko Mählisch, and Klaus Dietmayer. Generic centralized multi sensor data fusion based on probabilistic sensor and environment models for driver assistance systems. In *IEEE Intelligent Transportation Systems Magazine, Vol. 2, Issue 1*, 2010.

[163] Tobias Nothdurft, Peter Hecker, Sebastian Ohl, Falko Saust, Markus Maurer, Andreas Reschka, and Jürgen Rüdiger Böhmer. Stadtpilot: First fully autonomous test drives in urban traffic. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, Washington, DC, USA, October 2011.

[164] Yuichi Ohta and Takeo Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 7(1):139–154, March 1985.

[165] Florin Oniga and Sergiu Nedevschi. Curb detection for driving assistance systems: A cubic spline-based approach. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 945–950, Baden-Baden, Germany, June 2011.

[166] Florin Oniga, Sergiu Nedevschi, and Marc-Michael Meinecke. Temporal integration of occupancy grids detected from dense stereo using an elevation map representation. In 6^{th} *International Workshop on Intelligent Transportation (WIT)*, pages 133–138, Hamburg, Germany, March 2009.

[167] Florin Oniga, Sergiu Nedevschi, Marc-Michael Meinecke, and Thanh Binh To. Road surface and obstacle detection based on elevation maps from dense stereo. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, Seattle, WA, USA, September 2007.

[168] Mathias Perrollaz, Raphaël Labayrade, Romain Gallen, and Didier Aubert. A three resolution framework for reliable road obstacle detection using stereovision. In *IAPR Conference on Machine Vision Applications (MVA)*, pages 469–472, 2007.

[169] David Pfeiffer and Uwe Franke. Efficient representation of traffic scenes by means of dynamic Stixels. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 217–224, San Diego, CA, USA, June 2010.

[170] David Pfeiffer and Uwe Franke. Towards a global optimal multi-layer Stixel representation of dense 3D data. In *British Machine Vision Conference (BMVC)*, Dundee, Scotland, August 2011. BMVA Press.

[171] David Pfeiffer, Sandino Morales, Alexander Barth, and Uwe Franke. Ground truth evaluation of the Stixel representation using laser scanners. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, Maideira Island, Portugal, September 2010.

[172] Thomas Popham, Abhir Bhalerao, and Roland Wilson. Multi-frame scene-flow estimation using a patch model and smooth motion prior. In *Proceedings of the BMVC 2010 UK postgraduate workshop*, pages 2.1–2.11. BMVA Press, 2010.

[173] Ronald Poppe. Vision-based human motion analysis: An overview. *Journal on Computer Vision and Image Understanding*, 108(1-2):4–18, 2007.

[174] Vivek Pradeep, Gerard Medioni, and James Weiland. Piecewise planar modeling for step detection using stereo vision. In *Computer Vision Applications for the Visually Impaired (CIVAVI)*, 2008.

[175] Johann Prankl, Michael Zillich, Bastian Leibe, and Markus Vincze. Incremental model selection for detection and tracking of planar surfaces. In *British Machine Vision Conference (BMVC)*, pages 87.1–87.12, Aberystwyth, UK, August 2010. BMVA Press.

[176] Clemens Rabe, Uwe Franke, and Stefan Gehrig. Fast detection of moving objects in complex scenarios. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 398–403, Istanbul, June 2007.

[177] Clemens Rabe, Thomas Müller, Andreas Wedel, and Uwe Franke. Dense, robust, and accurate motion field estimation from stereo image sequences in real-time. In *European Conference on Computer Vision (ECCV)*, pages 582–595, Heraklion, Crete, Greece, September 2010.

[178] Fred W. Rauskolb, Kai Berger, Christian Lipski, Marcus Magnor, Karsten Cornelsen, Jan Effertz, Thomas Form, Fabian Graefe, Sebastian Ohl, Walter Schumacher, Jörn-Marten Wille, Peter Hecker, Tobias Nothdurft, Michael Doering, Kai Homeier, Johannes Morgenroth, Lars Wolf, Christian Basarke, Christian Berger, Tim Gülke, Felix Klose, and Bernhard Rumpe. Caroline: An autonomously driving vehicle for urban environments. *Journal of Field Robotics*, 25(9):674–724, 2008.

[179] Marcus Rohrbach, Markus Enzweiler, and Dariu M. Gavrila. High-level fusion of depth and intensity for pedestrian classification. In *German Association for Pattern Recognition (DAGM)*, pages 101–110, Jena, Germany, September 2009.

[180] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3D Digital Imaging and Modeling*, 2001.

[181] Falko Saust, Jörn Marten Wille, Bernd Lichte, and Markus Maurer. Autonomous vehicle guidance on braunschweig's inner ring road within the stadtpilot project. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 169–174, Baden-Baden, Germany, June 2011.

[182] Frederik Schaffalitzky and Andrew Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *European Conference on Computer Vision (ECCV)*, pages 414–431, Copenhagen, Denmark, May 2002. Springer.

[183] Daniel Scharstein and Richard Szeliski. Middlebury online stereo evaluation, 2002. http://vision.middlebury.edu/stereo.

[184] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*, 47(1-3):7–42, 2002.

[185] Matthias Schmid, Mirko Maehlisch, Jürgen Dickmann, and Hans-Joachim Wuensche. Dynamic level of detail 3d occupancy grids for automotive use. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 269–274, San Diego, CA, USA, June 2010.

[186] Sebastian Schneider, Michael Himmelsbach, Thorsten Luettel, and Hans-Joachim Wuensche. Fusing vision and lidar - synchronization, correction and occlusion reasoning. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 388–393, San Diego, CA, USA, June 2010.

[187] Gideon E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[188] Continental Automotive Industrial Sensors. ARS 300 Long Range Radar Sensor 77 GHz. `http://www.conti-online.com/generator/www/de/en/continental/industrial_sensors/themes/ars_300/ars_300_en.html`, July 2011.

[189] Yonggang Shi and William Clement Karl. Real-time tracking using level sets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 34–41, San Diego, CA, USA, June 2005.

[190] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, USA, June 2008.

[191] Jan Siegemund, Uwe Franke, and Wolfgang Förstner. A temporal filter approach for detection and reconstruction of curbs and road surfaces based on conditional random fields. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 637–642, Baden-Baden, Germany, June 2011. IEEE Computer Society.

[192] Jan Siegemund, David Pfeiffer, Uwe Franke, and Wolfgang Förstner. Curb reconstruction using conditional random fields. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 203–210, San Diego, CA, USA, June 2010.

[193] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[194] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

[195] Nicolas Soquet, Mathias Perrollaz, and Didier Aubert. Free space estimation for autonomous navigation. In *International Conference on Computer Vision Systems (ICVS)*, Bielefeld, Germany, March 2007.

[196] Fridtjof Stein. Efficient computation of optical flow using the census transform. In *German Association for Pattern Recognition (DAGM)*, pages 79–86, Tübingen, Germany, August 2004.

[197] Frank Steinbruecker, Thomas Pock, and Daniel Cremers. Large displacement optical flow computation without warping. In *International Conference on Computer Vision (ICCV)*, 2009.

[198] Pascal Steingrube, Stefan Gehrig, and Uwe Franke. Performance evaluation of stereo algorithms for automotive applications. In *International Conference on Computer Vision Systems (ICVS)*, pages 285–294, Liège, Belgium, October 2009. Springer-Verlag.

[199] Petr Stepan, Miroslav Kulich, and Libor Preucil. Robust data fusion with occupancy grid. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(1):106–115, 2005.

[200] Matthias Straka and Stefan Hauswiesnerand Matthias RÃŒtherand Horst Bischof. Skeletal graph based human pose estimation in real-time. In *British Machine Vision Conference (BMVC)*, Dundee, Scotland, September 2011. BMVA Press.

[201] Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip H. S. Torr. Combining appearance and structure from motion features for road scene understanding. In *British Machine Vision Conference (BMVC)*, London, UK, September 2009. BMVA Press.

[202] Jian Sun, Heung yeung Shum, and Nan ning Zheng. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25:787–800, 2003.

[203] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28:694–711, 2006.

[204] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall F. Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. In *European Conference on Computer Vision (ECCV)*, pages 16–29, Graz, Austria, May 2006.

[205] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall F. Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(6):1068–1080, 2008.

[206] Jean-Philippe Tarel, Sio-Song Ieng, and Pierre Charbonnier. Accurate and robust image alignment for road profil reconstruction. In *IEEE International Conference on Image Processing (ICIP)*, volume V, pages 365–368, San Antonio, Texas, USA, 2007.

[207] Olivier Teboul, Iasonas Kokkinos, Loïc Simon, Panagiotis Koutsourakis, and Nikos Paragios. Shape grammar parsing via reinforcement learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2273–2280, Colorado Springs, CO, USA, June 2011.

[208] Sebastian Thrun. Particle filters in robotics. In 17^{th} *Conference in Uncertainty in Artificial Intelligence*, pages 511–518, Edmonton, Alberta, Canada, August 2002. Morgan Kaufmann.

[209] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15:111–127, September 2003.

[210] Sebastian Thrun. *Robotic mapping: a survey*, pages 1–35. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[211] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the DARPA Grand Challenge: Research articles. *Journal of Field Robotics*, 23:661–692, September 2006.

[212] Massimo Tistarelli. Multiple constraints for optical flow. In *European Conference on Computer Vision (ECCV)*, pages 61–70, Stockholm, Sweden, May 1994.

[213] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, School of Computer Science, Carnegie Mellon University, April 1991.

[214] Carlo Tomasi and Jianbo Shi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, Los Alamitos, CA, USA, June 1994.

[215] Federico Tombari, Stefano Mattoccia, Luigi di Stefano, and Elisa Addimanda. Classification and evaluation of cost aggregation methods for stereo correspondence. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, AK, USA, 2008.

[216] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[217] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[218] Tobi Vaudrey, Hernán Badino, and Stefan Gehrig. Integrating disparity images by incorporating disparity rate. In *Robot Vision, Second International Workshop (RobVis)*, pages 29–42, Auckland, New Zealand, February 2008.

[219] Olga Veksler. Stereo correspondence by dynamic programming on a tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 384–390, Washington, DC, USA, 2005. IEEE Computer Society.

[220] Vincent Voisin, Manuel Avila, Bruno Emile, Stephane Bégot, and Jean-Christophe Bardet. Road markings detection and tracking using hough transform and kalman filter. In 7^{th} *International Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 76–83, Antwerp, Belgium, September 2005.

[221] Eric A. Wan and Rudolph van der Merwe. The unscented kalman filter for nonlinear estimation. In *The IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC*, pages 153–158, August 2002.

[222] Andreas Wedel, Hernán Badino, Clemens Rabe, Heidi Loose, Uwe Franke, and Daniel Cremers. B-spline modeling of road surfaces with an application to free-space estimation. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):572–583, 2009.

[223] Andreas Wedel, Uwe Franke, Hernán Badino, and Daniel Cremers. B-spline modeling of road surfaces for freespace estimation. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 2223–2228, Eindhoven, Netherlands, April 2008.

[224] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *European Conference on Computer Vision (ECCV)*, pages 739–751, Marseille, France, October 2008.

[225] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[226] Andrew P. Witkin. Recovering surface shape and orientation from texture. *Journal of Artificial Intelligence*, 17(1-3):17–45, 1981.

[227] Christian Wöhler and Joachim K. Anlauf. A time delay neural network algorithm for estimating image-pattern shape and motion. *Image and Vision Computing*, 17(3-4):281–294, 1999.

[228] Christian Wojek and Bernt Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *European Conference on Computer Vision (ECCV)*, pages 733–747, Marseille, France, October 2008.

[229] Oliver J. Woodford, Philip H. S. Torr, Ian D. Reid, and Andrew W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, USA, June 2008.

[230] Kai M. Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010.

[231] Kunio Yamada, Kenji Mochizuki, Kiyoharu Aizawa, and Takahiro Saito. Motion segmentation with census transform. In *Advances in Multimedia Information Processing, Second IEEE Pacific Rim Conference on Multimedia*, volume 2195, pages 903–908, Bejing, China, October 2001. Springer.

[232] Qingxiong Yang, Liang Wang 0002, and Narendra Ahuja. A constant-space belief propagation algorithm for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1458–1465, San Francisco, CA, USA, June 2010.

[233] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, December 2006.

[234] Alper Yilmaz, Xin Li, and Mubarak Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(11):1531–1536, 2004.

[235] Ke-Hai Yuan and Peter Bentler. Robust mean and covariance structure analysis through iteratively reweighted least squares. In *Psychometrika*, volume 65, pages 43–58, March 2000.

[236] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l1 optical flow. In *German Association for Pattern Recognition (DAGM)*, pages 214–223, Heidelberg, Germany, September 2007.

[237] Ke Zhang, Jiangbo Lu, Gauthier Lafruit, Rudy Lauwereins, and Luc J. Van Gool. Accurate and efficient stereo matching with robust piecewise voting. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 93–96, New York City, NY, June 2009.

[238] Ke Zhang, Jiangbo Lu, Gauthier Lafruit, Rudy Lauwereins, and Luc J. Van Gool. Robust stereo matching with fast normalized cross-correlation over shape-adaptive regions. In *IEEE International Conference on Image Processing (ICIP)*, pages 2357–2360, Cairo, Egypt, November 2009.

[239] Yilei Zhang, Minglun Gong, and Yee-Hong Yang. Local stereo matching with 3d adaptive cost aggregation for slanted surface modeling and sub-pixel accuracy. In *International Conference on Pattern Recognition (ICPR)*, pages 1–4, Tampa, Florida, USA, December 2008.

[240] Xue Zhou, Weiming Hu, Ying Chen, and Wei Hu. Markov random field modeled level sets method for object tracking with moving cameras. In *Asien Conferenceon Computer Vision (ACCV)*, pages 832–842, Tokyo, Japan, 2007. Springer-Verlag.

[241] Christian Zinner, Martin Humenberger, Kristian Ambrosch, and Wilfried Kubinger. An optimized software-based implementation of a census-based stereo matching algorithm. In *International Symposium on Visual Computing (ISVC)*, pages 216–227, Las Vegas, Nevada, September 2008.