Machine Vision and Applications (2014) 25:561–581
DOI 10.1007/s00138-011-0385-1

SPECIAL ISSUE PAPER

# Parking assistance using dense motion-stereo

## Real-time parking slot detection, collision warning and augmented parking

**Christian Unger · Eric Wahl · Slobodan Ilic**

Received: 31 August 2010 / Revised: 29 July 2011 / Accepted: 15 November 2011 / Published online: 15 December 2011

**Abstract** The ability of generating and interpreting a three-dimensional representation of the environment in real-time is one of the key technologies for autonomous vehicles. While active sensors like ultrasounds have been commercially used, their cost and precision is not favorable. On the other hand, integrating passive sensors, like video cameras, in modern vehicles is quite appealing especially because of their low cost. However, image processing requires reliable real-time algorithms to retrieve depth from visual information. In addition, the limited processing power in automobiles and other mobile platforms makes this problem even more challenging. In this paper we introduce a parking assistance system which relies on dense motion-stereo to compute depth maps of the observed environment in real-time. The flexibility and robustness of our method is showcased with different applications: automatic parking slot detection, a collision warning for the pivoting ranges of the doors and an image-based rendering technique to visualize the environment around the host vehicle. We evaluate the accuracy and reliability of our system and provide quantitative and qualitative results. A comparison to ultrasound and feature-based motion-stereo solutions shows that our approach is more reliable.

C. Unger (✉) · E. Wahl
BMW Group, Max-Diamand-Str. 13,
80788 Munich, Germany
e-mail: Christian.Unger@bmw.de

E. Wahl
e-mail: Eric.Wahl@bmw.de

S. Ilic
Technische Universität München, Boltzmannstr. 3,
85748 Garching, Munich, Germany
e-mail: Slobodan.Ilic@in.tum.de

**Keywords** Dense motion-stereo · Parking space detection · Advanced driver assistance · Collision detection · Augmented parking · Image-based rendering

## 1 Introduction

In recent years camera-based advanced driver assistance systems (ADAS) have been established in the automotive industry. Currently, popular applications like a back-up display with reverse guide lines, lane departure warning (LDW), traffic sign recognition (TSR), pedestrian detection (PD), or high beam automation (HBA) target the space in front or behind a car. These comfort functions were designed to ease the driving task, but still require the driver to be part of the control loop. Stepping from an assisting vehicle to a fully autonomous one, depth information about the car's lateral space is also required.

However, applications interpreting the lateral space of a car are still rare and mostly limited to ultrasonic, radar or LIDAR sensors. Due to physical limitations, these sensors strongly reduce details acquired from the world to a small set of depth measurements. Thus, every depth model derived from these sensors is very coarse and therefore strongly limits the number of possible applications. On the other hand their cost is still higher than the cost of simple passive cameras. Currently, only lateral ultrasonic sensors have been commercially used for the parking assistance systems. However, due to their small resolution, they have only been used for parking slot detection.

Even though cheap, lateral cameras were, so far, used for a "visual enhancement" for the driver. They enable the driver to observe areas around a car that are occluded due to geometric restrictions as shown in Fig. 2. Until now, the complexity of image processing algorithms and limited hardware resources (i.e. only mobile CPUs in our case) have hid the potential of lateral cameras from their application to driver assistance systems, since real-time performance is required for most of these applications. In this paper we investigate the usage of these lateral cameras for multiple applications such as parking slot detection, collision warning for the door opening and a virtual bird's eye view for augmented parking. We show that these passive sensors can be turned in very powerful dense depth measuring devices. This is possible thanks to the efficient real-time motion-stereo algorithm we developed, which allows obtaining dense depth maps of the observed lateral space of the vehicle on mobile CPUs. Due to the motion of the vehicle pairs of images acquired at consecutive time instances (i.e. from different viewpoints) are used to compute disparity maps based on a real-time stereo method we developed [30]. To eliminate outliers and to improve accuracy of the depth information we fuse the temporal history of disparity maps. From these measurements we extract the ground plane and obstacles and update a map of the environment accordingly. Finally, this map is used for the detection and measurement of parallel and cross parking slots. Furthermore, we also use the obtained spatial information to detect possible collisions with objects in the pivoting areas of the doors and to compute a virtual bird's eye view for augmented parking. The host vehicle, detected parking slots and surrounding obstacles are displayed over an image of the ground plane as shown in Fig. 9.

In the following, we review related work and describe the processing steps of our environment perception in more details. Since we concentrate on the applications, a brief description of the motion-stereo method will be given (i.e. the stereo matching and temporal fusion) and detailed discussion about the customer functions, which rely on the recovered depth maps, will be provided. In the end we present a quantitative evaluation of the described functionalities and also compare them to feature-based motion-stereo [35,36] and a solution based on an ultrasonic sensor [20]. A number of qualitative results and illustrations are also provided.

## 2 Related work

There are several different ways to perceive spatial information about the lateral space of the vehicle's environment. In the following we review some works in the context of parking assistance. Most popular are current commercial solutions based on ultrasonic sensors [26], but these are only passive systems with the purpose to inform the driver about distances to nearby objects. Recent products additionally utilize a laterally mounted ultrasonic sensor to detect parallel parking slots into which the vehicle may be navigated semi-automatically [18–20]. However, depending on the required measurement range, these systems may not be able to detect cross parking slots and complex geometry makes correct interpretation of sensor signals difficult. Similar is a system developed by Schanz [22]: it uses a laterally mounted laser-scanner and, due to the relatively good measurement accuracy, the system is able to detect both parallel and cross parking slots. However, in practice laser-scanners are currently too expensive for mass production. Compared to these types of sensors, another benefit of our camera-based approach is that very rich depth information is acquired at low costs.

Systems that use cameras have also been investigated. Kämpchen et al. [11] detect parking lots using a forward looking stereo vision system: a point cloud generated from sparse stereo correspondences is analyzed to detect vehicles. Generic vehicle models are used to estimate their poses and parking slots are detected by analyzing the free space between two vehicles. However, due to the orientation and the limited field of view (FOV) of the stereo system, the detection of parallel and especially cross parking slots may be difficult.

The use of PMD cameras was evaluated by Scheunert et al. [24]: from the 3D data they build a local 2D grid where every cell is in one of four modes (unknown, ground, obstacle low and obstacle high), depending on the height of a point. From this grid, the curb is determined and a distance profile is computed. This information is then used to detect free spaces. However, they do not determine the envelope of the ground plane dynamically and since they assume the presence of the curb, they did not demonstrate the detection of cross parking slots. Furthermore, the PMD technology is still very expensive and thus not suitable for serial production.

It is also possible to detect parking slots using a camera and a projection of structured light [10]. However, legal restrictions in many countries render a worldwide commercialization of such solutions impossible. Other systems detect parking slots by extracting and interpreting ground markings [9,39]. But the applicability and thus customer value is very limited, because the markings have to fulfill specific requirements on color, visibility and geometric properties.

There is also a wide range of recent methods that use the principle of motion-stereo [6,25,27,28,33,35,36]. However, these works address only a feature-based strategy and no one utilized dense disparity maps. The basic idea is to calculate characteristic features in subsequent images. Over time, this relatively small number of points is tracked and then a 3D reconstruction is analyzed to find parking slots. These approaches perform well in friendly conditions, i.e. as long as enough strong and distinctive features can be derived from the images. However, challenging are both lowly textured objects, which lead to very sparse point clouds, or also complex textures like foliage, where high ambiguity during feature matching introduces wrong distance measurements. Moreover, features are not necessarily located at the boundaries of objects. Thus the size of objects and free space might be wrongly calculated. In these situations, the accuracy and reliability of the determination of free parking areas varied in an unacceptable way.

In this work, we present a powerful approach that is based on *dense* motion-stereo, where at every frame a dense disparity map is computed. This results in important advantages, namely a very high detection rate of obstacles, a high measurement accuracy, a nearly drift free environment model and the ability to display a multitude of different customer functions.

## 3 Environment modeling

Our goal is to support the driver and eventually other occupants of the car at parking related tasks. In the first place, this includes assistance for finding a parking slot (i.e. automatic detection and measurement of free space). This also includes an adequate interface to the driver, which provides a visualization of the found parking place. In practice, we generate a bird's eye view of the ground plane with the host vehicle, parking space and obstacles overlaid. Finally, we want to inform the driver and occupants if obstacles are located in the pivoting ranges of doors in order to prevent minor damage. Another important requirement is that all calculations can be performed in real-time on standard CPUs (without any GPU) at 30 frames/s. In this paper, we demonstrate that all these goals can be achieved using one generic processing pipeline.

In our work we focus on acquiring 3D information using monocular camera systems attached to the side of the vehicle as shown in Figs. 2 and 3. The basic principle of every camera is the projection of a three-dimensional world to a flat two-dimensional image plane. The estimation of depth information is either based on complex and often wrong assumptions about the world, or by accounting on the movement during the data acquisition process. Since it is more generic, we chose to follow the second principle, also known as *motion-stereo*: depth information is calculated using corresponding image points from consecutive images acquired at different known positions [14].

### 3.1 Method overview

Following Fig. 1, our approach is composed of several processing steps. Based on the principle of dense motion-stereo, we determine a depth for every pixel in every camera image using classical stereo methods [8,14,23,30] or optical flow [1]. In our case, a rectified pair of camera images is used for the stereo matching (for example, the last two images acquired from the camera). Since these calculations are relatively expensive in terms of processing power, only efficient methods can be applied [8,16,30]. After that, the history of disparity maps is fused probabilistically in order to obtain for every camera image the most probable disparity map that exposes a minimum amount of outliers. In every fused disparity map we detect the ground plane, obstacles and from that a silhouette which limits the free space. Then, we combine all these partial silhouettes so that over time a global model of the environment is created incrementally. Within this model, we detect parallel and cross parking slots. If a free space region provides enough space for the vehicle and is bounded by obstacles, then it is a candidate for a parking slot and the exact metric size is computed.

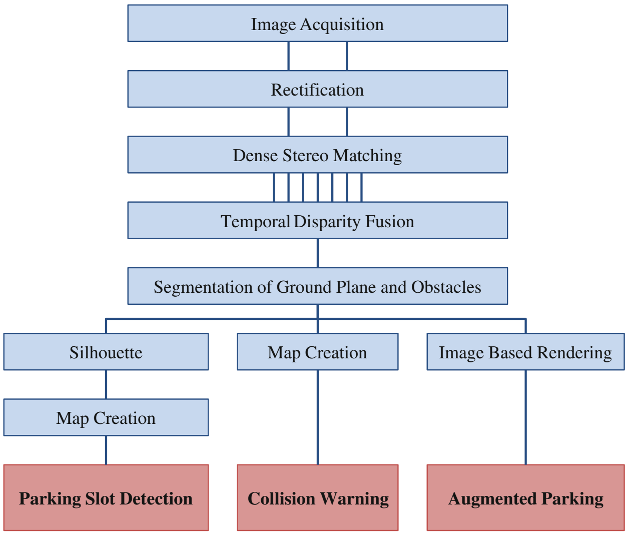
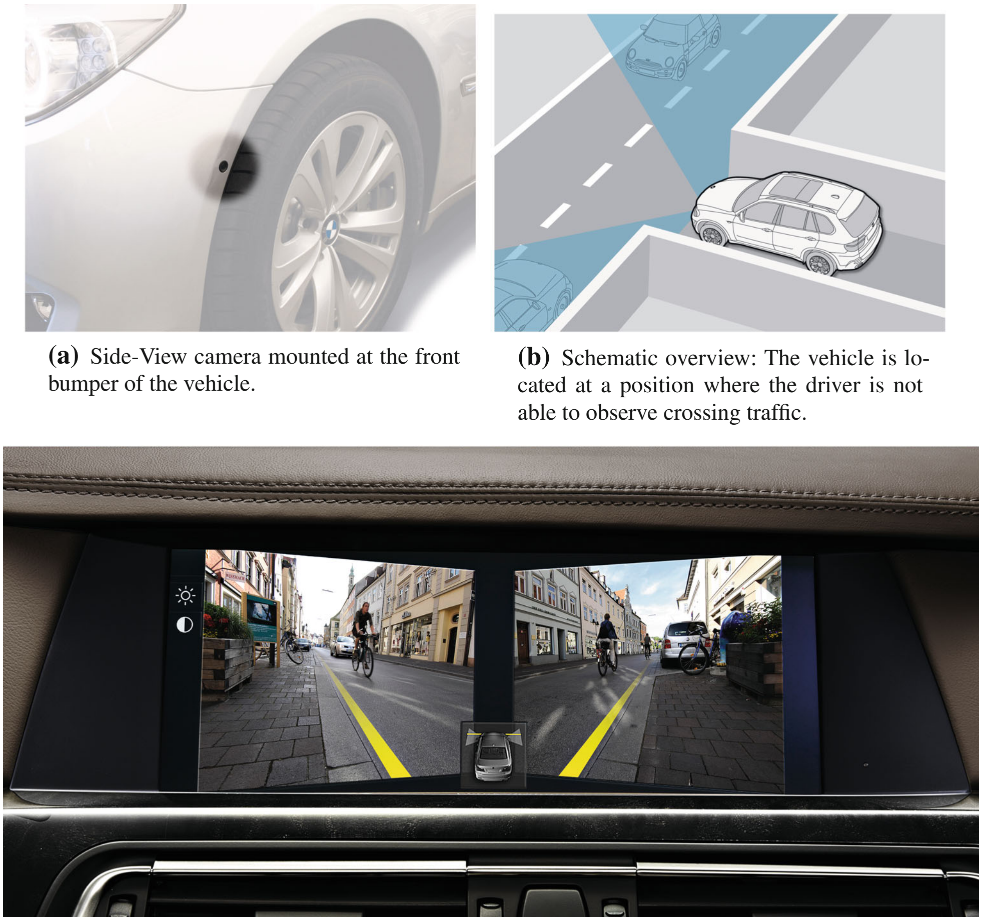


**Fig. 1** The processing pipeline of our approach: we rectify images acquired at different positions from a monocular camera and use them to compute a dense disparity map using stereo [30]. To eliminate outliers and to improve accuracy of the depth information we fuse the temporal history of disparity maps. From these measurements we segment the ground plane and obstacles. This information is the foundation for the different applications: the *parking slot detection* computes silhouettes of the observed area from the segmented disparity maps, creates a map out of them and detects and measures free spaces. The *collision warning* uses obstacle points to examine the pivoting ranges of the doors, and *augmented parking* takes advantage of segmented disparity data to render an image of the ground plane with the host vehicle, parking space and obstacles overlaid

Further, we use the disparity maps to obtain a local 3D reconstruction of specific regions of interest (for example, the pivoting range of a door). Using such a local 3D reconstruction, we perform a collision analysis and, if necessary, issue a warning to occupants to prevent minor damages. Another application is *Augmented Parking* and uses image-based rendering to compute a virtual bird's eye view to visualize the positions of the host vehicle, obstacles and the parking slot to the driver. In the following we will discuss all necessary elements of the system.

### 3.2 Camera sensors

The position and orientation of a camera with respect to the vehicle are important parameters. Two categories of cameras



**(a)** Side-View camera mounted at the front bumper of the vehicle.

**(b)** Schematic overview: The vehicle is located at a position where the driver is not able to observe crossing traffic.

**(c)** User interface: the left- and right-looking camera images are displayed so that the driver is able to observe traffic.

**Fig. 2** Side-view cameras: **a** the mounting position, **b** schematic overview of the use case and **c** the user interface

are relevant for dense motion-stereo with respect to their orientation and applied functions.

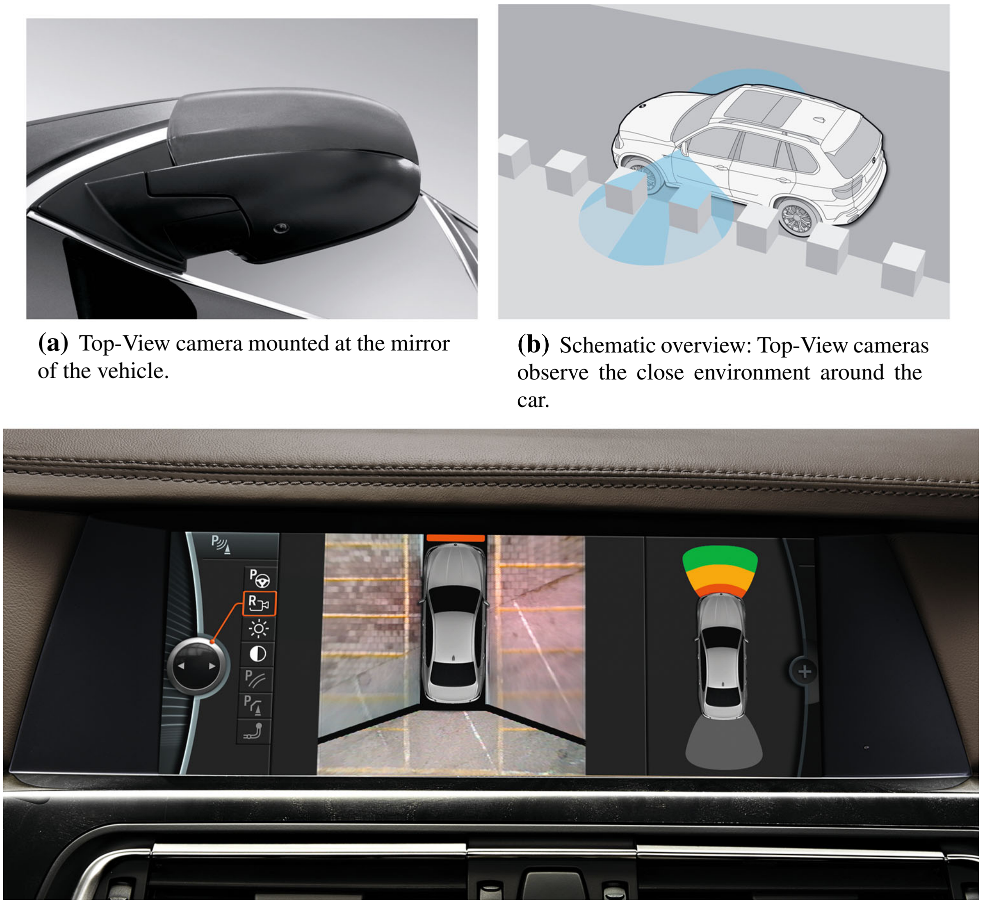
The first class is the family of side-view cameras (see Fig. 2), which is located in the front part of a vehicle. The optical axis of these cameras is parallel to the ground and orthogonal to the orientation of the vehicle so that they are well suited for "first views" in situations when the driver has an obstructed line of sight such as at the exit of car parks. Accordingly, side-view cameras are mostly equipped with standard lenses.

Another class of lateral cameras is the family of top-view cameras (see Fig. 3). Here the goal is to provide the driver with a virtual *surround view* containing the close environment around his car to give visual support during low speed or parking maneuvers. Respectively, these cameras are positioned in central parts of the body shell, where a wide angle lens allows displaying the right and left areas.

In our experiments we used both types of cameras: they operate at VGA-resolution (640 × 480 pixels) at 30 frames/s. The diagonal FOV of the side-view and top-view camera is 68° and 170°, respectively. For stereo matching, we downsample the images to a resolution of 213 × 160.

### 3.3 Calibration and rectification

For stereo, a correct camera calibration and rectification is of eminent importance. In particular, a correction of the radial lens distortion is indispensable [3,42]. Further, the poses of a camera at two different points in time has to be determined, in order to rectify pairs of images. In our case, odometry information was sufficient for that. However, if no odometry is available or if it is too inaccurate, then this rectification may be estimated from image correspondences [29]. The yaw-angle and movement in $x$- and $y$-direction can be determined relatively well from odometry information, if the position of the camera relative to the vehicle-origin is known. In practice, the recovery of pitch and roll angles as well as the movement in $z$-direction is relatively imprecise with current vehicle sensors. Due to this reason, we ignore these values in the first place. We rather use a simplified, approxi-



**(a)** Top-View camera mounted at the mirror of the vehicle.

**(b)** Schematic overview: Top-View cameras observe the close environment around the car.

**(c)** User interface: the images of the top-view and backward-looking cameras are undistorted and displayed together.

**Fig. 3** Top-view cameras: **a** the mounting position, **b** schematic overview of the use case and **c** the user interface

mated rectification (using only the yaw-angle and movement in $x$- and $y$-direction) and propose an extension [31] to our real-time stereo method [30] which makes it robust against an inaccurate estimation of the epipolar geometry. Another benefit of the simplified rectification is that the warping of images can be implemented in an optimized way. To summarize, we only approximate the rectification of images acquired at different points in time using the odometry information by resorting to a simplified model, and more complex scenarios (e.g. uneven ground) are handled by an improved stereo matching.

### 3.4 Stereo matching

For the real-time computation of disparity maps, only highly efficient methods with very low computational requirements can be used. By using the enormous processing power of today's graphics cards, many complex stereo-methods could be implemented near real-time or even real-time [2,4,21,38]. However, such hardware platforms are, especially against the background of dissipation of energy, not available in vehicles and it is absolutely necessary that all calculations can be performed on a standard mobile CPU at a frame rate of 30 Hz.

For the stereo-processing we use our local method [30] that runs significantly faster than traditional real-time implementations [8,16]. In particular, [30] is suited very well for motion-stereo setups, as it does not require a-priori knowledge about the maximum disparity, which depends on the motion model, the camera intrinsic parameters and on the depths of the observed scene. In our implementation, the use of SIMD-instructions allows us to compute a disparity-map for a $320 \times 240$ image in less than 30 ms.

*Method summary* In [30] the disparity map is computed iteratively, by performing two operations at every pixel: a minimization followed by a propagation-step. The minimization follows a line search strategy and usually finds the "next" (depending on the iteration of the disparity map) local minimum of the matching cost function. Since matching costs have many local minima a propagation-step is introduced in order to find further, better minima by investigating disparity values of adjacent pixels. Moreover, these steps are embedded into a hierarchical setup. Compared to other traditional methods or dynamic programming, works very well, especially on sequences from the vehicle. Further, [30] converges very quickly (after 2–3 iterations) even at large baselines.

*Frame decimation* Since the vehicle moves with different velocities, the baseline of adjacent frames is not constant. Especially for low velocities, the baseline becomes too small for accurate depth computations. In practice, we use a simple frame decimation [17] technique to improve stereo matching: for every reference frame, we select the matching frame in a way such that the baseline is always greater than 10 cm. We obtain the baseline from odometry information.

*Histogram equalization* To some extent, the high efficiency results from using the sum of absolute differences (SAD) as a matching cost. However, when the camera moves through its environment, lighting conditions will change constantly and sometimes very abruptly. Because of the characteristics of the camera, we decided not to work with a constant exposure time, so to always allow optimal exposure. But this results in stereo pairs with different exposures, which has adverse effects on the matching using SAD. To reduce these problems we chose to use histogram equalization.

### 3.5 Improving matching for motion-stereo

In real-world situations, the ground is not perfectly flat and will cause the vehicle to pitch and roll (e.g. due to road holes). In such scenarios the rectification would be more complex than the simplified one described in Sect. 3.3. However, in practice the simplified rectification is sufficient, if the search area of stereo matching is slightly increased, so that a set of neighboring scanlines is taken into account too. Our proposed extension is a simplified method of the generic approach that we presented in [31]. It integrates very smoothly into [30] and leads to a method that lies somewhere between optical flow and stereo. We do not only determine the horizontal displacement (disparity) but also a vertical displacement for every pixel. In practice, we assume that the vertical displacement is small, i.e. a few pixels at most.

In the following, we give a short overview to an efficient implementation, and we refer to a much more powerful method and elaborate description to [31]. We use a similar notation as in [30] and replace the map of scalar disparities $\mathscr{D}(\mathbf{p})$ by a map of two-dimensional flow-vectors $\mathscr{F}(\mathbf{p})$. [30] is an iterative approach, where the disparity of every pixel is computed using a steepest descent minimization. The minimization is alternated with a propagation step which propagates disparities to neighboring pixels. In the default formulation of the minimization, the dissimilarity of the incremented disparity $E_{\text{test}} = E(\mathbf{p}, d+1)$ is compared against the dissimilarity of the current disparity $E_{\text{cur}} = E(\mathbf{p}, d)$. If $E_{\text{test}} < E_{\text{cur}}$, then the disparity of $\mathbf{p}$ is updated to $d + 1$. At this step we introduce the vertical displacement. For $E_{\text{test}}$ we use the minimum of these three values:

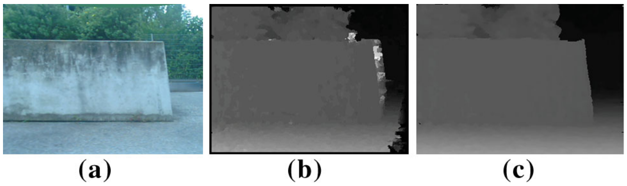


**Fig. 4** Example for the temporal fusion: **a** the camera frame, **b** a single baseline disparity map computed using [30] and **c** a fused disparity map using [32]

$$E(\mathbf{p}, (d+1, f'_y)) \quad \text{with } f'_y \in \{f_y - 1, f_y, f_y + 1\} \tag{1}$$

with $(d, f_y) = \mathscr{F}(\mathbf{p})$ being the current disparity and vertical displacement. If $E_{\text{test}} < E_{\text{cur}}$, we update the disparity of $\mathbf{p}$ to $d + 1$ and set the vertical displacement to $f'_y$. Accordingly, the vertical displacement must be updated in the propagation step together with the disparity (so, we propagate both the horizontal and vertical displacement at the same time):

$$\mathscr{F}(\mathbf{p}) \mapsto \arg\min_{\mathbf{f} \in N(\mathbf{p})} E(\mathbf{p}, \mathbf{f}) \tag{2}$$

with the neighboring displacement vectors $N(\mathbf{p})$.

In practice, this extension degrades the performance by only roughly 50% (which is a factor of 2), so that real-time processing is still possible at a resolution of 213 × 160. In contrast, if the search range is increased in traditional real-time stereo methods like [8], the execution time is multiplied by the number of scanlines that have to be taken into account.

Please note that the vertical displacements may be used to update the rectification. In practice however, we discard the vertical displacements.

### 3.6 Temporal fusion of disparity maps

In practice, real-time stereo matching is error prone and is known to have weaknesses in regions near discontinuities [8,23]. Therefore, we use the redundancy in the temporal history of disparity maps in order to obtain a fused disparity map that exposes a minimum amount of outliers. We use the method proposed in [32] since it allows real-time operation on standard CPUs and provides a very good accuracy, especially in occluded parts and in regions near discontinuities (see Fig. 4 for an example). Popular alternatives are [15,40,41], but [15,40] require a GPU for real-time operation and [41] is far from being real-time.

*Method summary* Reference [32] fuses available disparity maps probabilistically in order to produce an accurate disparity map for the current camera frame: previously computed single-baseline disparity maps are reprojected to the current reference motion-stereo pair (for example, the last two frames). Provided that all camera centers are aligned on a straight line, the reprojection can be implemented very efficiently. In practice, we dynamically determine the frames for which this requirement is approximately fulfilled and fuse at most 16 disparity maps. After reprojection, visibility constraints are maintained and a probability density function over all valid disparities in the reference view is computed using uncertainties of these reprojections and their photo-consistencies. Finally, the most probable disparity map is selected from this distribution.

### 3.7 Ground plane segmentation

One goal of our system is to retrieve information about areas that are *free* for parking and parts of the environment that are *occupied* by obstacles. To accomplish this, our strategy is to analyze individual disparity maps to identify points that belong to the ground plane and points that belong to the obstacles. We perform this classification solely in disparity maps and therefore, we look at the plane plus parallax homography induced by the ground plane:

$$\mathbf{H}_\mathrm{G} = \mathbf{I} + s\,\mathbf{e}_1\mathbf{v}^\mathrm{T} \tag{3}$$

where $s$ is proportional to the traveled distance, $\mathbf{e}_1 = (1, 0, 0)^\mathrm{T}$ is the baseline and $\mathbf{v} = (v_1, v_2, v_3)^\mathrm{T}$ is the normal vector of the ground plane. In our case, the camera is mounted on a ground vehicle, so the baseline is parallel to the ground plane and therefore we can assume that $v_1 = 0$. The other two unknown components $v_2$ and $v_3$ depend on the slope of the ground. These we want to recover from the disparity map (we could also use odometry information for that, but our experiments showed that it is less reliable). With $\mathbf{p} = (x, y, 1)^\mathrm{T}$, we obtain the following linear relationship for the disparities of points belonging to the ground plane:

$$\begin{aligned}\mathscr{D}(\mathbf{p}) &= \mathbf{e}_1^\mathrm{T}(\mathbf{Hp} - \mathbf{p}) = \mathbf{e}_1^\mathrm{T}(s\,\mathbf{e}_1\mathbf{v}^\mathrm{T})\mathbf{p}\\ &= s(\mathbf{e}_1^\mathrm{T}\mathbf{e}_1)(\mathbf{v}^\mathrm{T}\mathbf{p}) = \underbrace{(s\,v_2)}_{=:A}\,y + \underbrace{(s\,v_3)}_{=:B}\end{aligned} \tag{4}$$

This implies that the disparity of ground plane pixels is constant within a scanline and motivates us to determine the parameters $A$ and $B$ by analyzing the histograms of disparities created for each scanline.

The rough idea is to obtain initial guesses of $A$ and $B$ and then use a greedy algorithm to refine them by looking at more and more scanlines. We start this estimation at the bottom of the image and successively use scanlines above. We do it this way, because in practice in most cases the ground plane is visible in the bottom part of the image. To reject scanlines containing a high amount of outliers, we use the confidence measure which is provided by the temporal fusion for every disparity. By assuming that disparities with a small confidence are wrong, we estimate the number of outliers for every scanline. Then, during the whole estimation-process of $A$ and $B$, we use only those *reliable* scanlines whose outlier-count is below a threshold (in practice, a scanline should not contain more than 50% outliers).

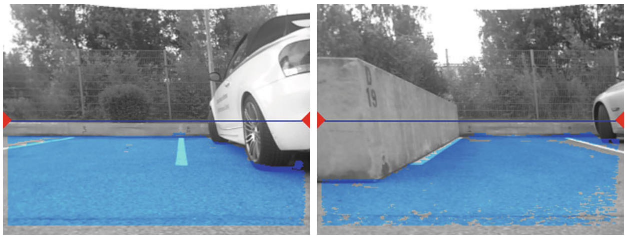


**Fig. 5** Superimposed ground plane segmentation: ground plane pixels appear tinted, the estimated horizon line is indicated using a line and the two ticks show the theoretical position of the horizon line. Please note that the segmentation also finds points under the car

*Initial estimation* We start with an initial estimation of $A$ and $B$, and iteratively refine these values. For the initial estimation, we use the bottommost four reliable scanlines and compute the parameters by $L_2$-regression: for each scanline, we compute a separate histogram of disparities and take the predominant value. Then these four values are used for the regression using (4). The initial estimates are then filtered using a Kalman-filter, together with values of the ground plane estimation of the previous camera frame.

*Iterative refinement* Once we obtained a first guess of the ground plane model, we add more scanlines to refine the model: we visit each scanline from bottom to top and look at the peaks of the histogram of each scanline. If the predominant value of the histogram fits to the ground plane model we use the value to refine the parameters $A$ and $B$ by adding the value to the regression: we do this by comparing the predominant value to the value predicted by the current ground plane model, and if the absolute difference is below a certain threshold, we update the ground plane model. During this process only those scanlines have to be considered for which the predicted disparity is positive, i.e. for $y > \frac{-B}{A}$ ($\mathscr{D}(\mathbf{p}) = 0$ is the horizon line). Finally, we are able to segment the ground plane by checking the disparity of every individual pixel against the ground plane model—independent from whether their scanline was used for the estimation of $A$ and $B$. Figure 5 is an example for such a segmentation: ground plane pixels appear tinted and the estimated horizon line is visualized using a line. The ticks on the left and right sides of the image indicate the theoretical position of the horizon line (computed using the camera extrinsic parameters and a canonical, perfectly flat ground).

*Maximum likelihood estimation* After the iterative refinement we identified a set of disparities that fulfill a simple linear model and the corresponding matrix $\mathbf{H}_\mathrm{G}$ has 2 degrees of freedom. To further improve accuracy, we randomly select 1,000 points of the ground plane segmentation and compute the parameters of $\mathbf{H}_\mathrm{G}$ using the QR-decomposition.

*Challenges* In situations, when the road is not flat (i.e. when it is curved or bumpy) then the algorithm, in some sense, approximates the true surface by a flat plane that fits best to the closest parts of the ground. Small bumps in the ground are not critical from a practical point of view because in most parking spaces there are no bumps within parking slots. Further, small holes can be filled by performing a closing operation on the ground plane segmentations in regions where no obstacle was detected.

More challenging are roads that exhibit a high curvature or rough terrain. In difficult cases, during the iterative estimation process the parameters $A$ and $B$ will "diverge" slowly to some values that fit best to the visited scanlines. After that, the final segmentation (after checking every disparity value) will often exhibit a lot of holes and the set of scanlines spanned by the segmentation is different to the set of scanlines used for estimation. To this end, the detection of these challenging cases is quite difficult and still not very reliable. One possible way is by allowing more parameters for the ground plane, but in our experiments we observed less robustness in simple situations. A temporal filtering of the parameters helps to some extent, but depends on a good initialization. However, in many parking situations the ground is relatively flat and our algorithm performs quite well on average.

### 3.8 Computation of silhouettes

Once we obtained information about obstacles and the ground plane, our goal is to compute a *silhouette* that limits the free space. The free space is bounded by obstacles and by the region borders of the ground plane. For example, in the majority of cases, the curb is not detected as an obstacle, but the ground plane segmentation stops there. We define that the silhouette is represented in image coordinates.

*Obstacle silhouette* In practice the obstacles in our scenes can mainly be approximated by large fronto-parallel planar patches (at least for side-view cameras—with top-view cameras such a requirement can be enforced by rotating the camera virtually). Our practical tests confirmed that this assumption still holds for plants like bushes, motorcycles and curved parts of other vehicles. To some extent, this can be explained by the quantization of disparity values.

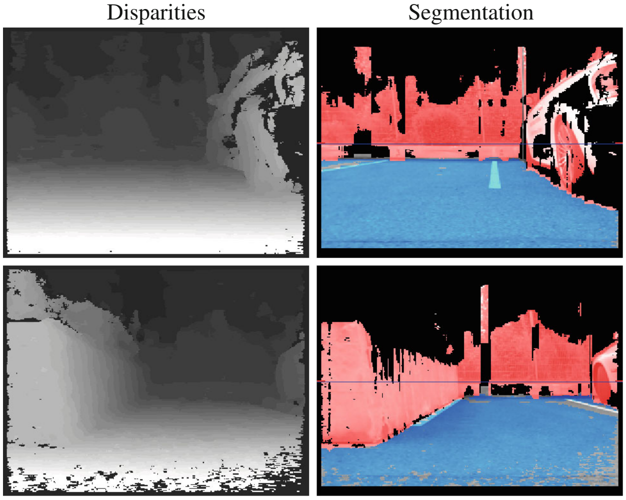


**Fig. 6** Superimposed ground plane and obstacle segmentation. *Left column*: fused disparity maps. *Right column*: corresponding rectified camera frame superimposed with the segmentation result. *Black pixels* are not considered relevant by the obstacle segmentation. All other pixels belong are either to the ground plane (*blue pixels* in color print) or to the closest estimated obstacle (*red pixels* in color print). The line in the middle indicates the estimated horizon line

Due to these properties, we first compute the histograms of disparities within single image columns, but only from those disparities which are not part of the ground plane. Building such histograms increases robustness against outliers (in difficult scenes, more stability can be attained by computing the histograms over multiple columns). From these histograms we collect the first $N_\mathrm{OBS}$ predominant entries, but only if their count is greater than a specific threshold $\theta_\mathrm{OBS}$ (to exclude disparities caused by noise).[1] From these disparities we take the largest one (minimal obstacle distance) and project the value onto the ground plane by solving (4) for $y$. This results in a single silhouette-point for an image column. By applying these steps to every image column we obtain a silhouette of the obstacles (Fig. 6).

*Ground plane silhouette* Often, the curb is not detected as an obstacle but the ground plane segmentation stops at the curb. Due to this reason, we also compute a silhouette of the ground plane by analyzing image columns of the ground plane segmentation. For a specific image column $x$, the location of the silhouette point is defined as the topmost pixel that belongs to the ground plane. If the ground plane is not visible in column $x$ we invalidate this silhouette point, because then there is no information available about the ground plane.

### 3.9 Cumulative map creation

We will use this map later for the parking slot detection and will propose slight modifications for other customer

[1] In our experiments we set $N_\mathrm{OBS} = 3$ and $\theta_\mathrm{OBS} = 10$.

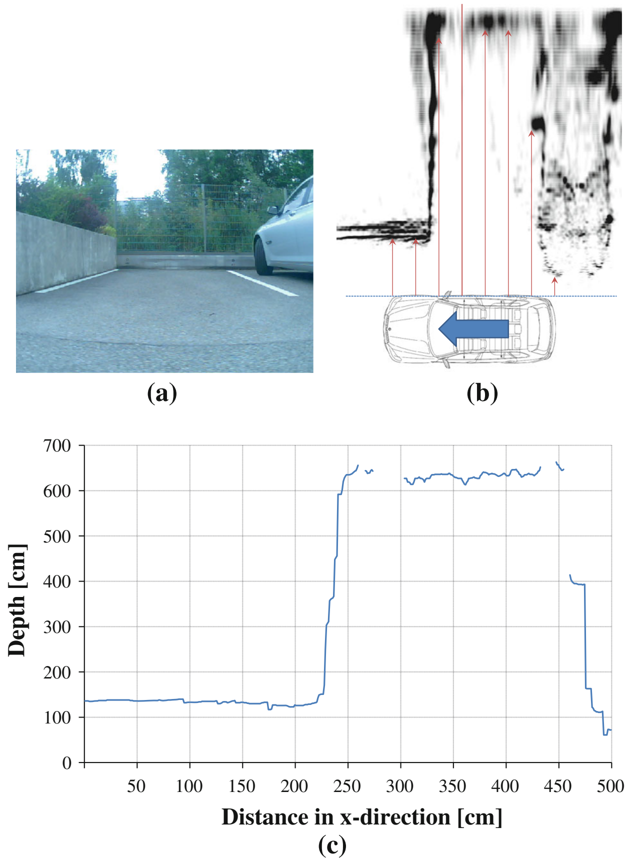


**Fig. 7** Parking slot detection: **a** the car passes by a parking slot and builds the cumulative map (**b**). **c** From this map, a distance profile is derived and the parking slot detection is performed

functions. We define our map to represent a specific region of the ground plane around the host vehicle from a bird's eye view as shown in Fig. 7b. We build this map of the environment incrementally and we divide it into small equi-sized cells where every cell stores the likelihood that the cell is occupied. We continuously update this map by "adding" the silhouettes. Both the obstacle and ground plane silhouettes are transformed from image coordinates into the bird's eye view using a transformation $\mathbf{H}_{\mathrm{BEV}}$, which warps the camera image onto the ground plane. This is equivalent to mapping points of the horizon line to infinity. This transformation may be computed using the horizon line computed using the ground plane segmentation. Every cell of the map has associated a likelihood which is increased every time a silhouette projects there.

Hence, over time more and more silhouettes are added. It must be noted that the ground plane segmentations may have different locations of the horizon line. Sudden large changes of the horizon line should be avoided, or otherwise the distance measures in the map become inconsistent. Therefore we assume that the variation of the horizon line is small, in case a parking slot is in the FOV. While an imprecision within a specific threshold is normal, large variations of the ground plane indicate either large changes in the slope of the ground, or inaccuracies in the ground plane segmentation. Both situations are easily detected using the variance of the horizon line and then the parking slot detection should disregard the corresponding region.

*Position estimation* Our goal is to use the plane plus parallax homography $\mathbf{H}_{\mathrm{G}}$ obtained from the ground plane segmentation for the position estimation. The map represents a defined portion around the host vehicle (the host vehicle has a constant position and orientation). Since the vehicle moves over time, the map must be updated by a translation and rotation. These updates have to happen continuously at every camera frame, because of the movement of the vehicle. With the transformations $\mathbf{H}_{\mathrm{BEV}}$ and $\mathbf{H}_{\mathrm{BEV}} \cdot \mathbf{H}_{G}^{-1}$, the movement of the vehicle can be taken into account. However, at this point we also have to take care of the rectification of the stereo pair. Since the transformation $\mathbf{H}_{\mathrm{G}}$ maps points of the current rectified camera frame to the previous rectified camera frame via the ground plane, we have to include the pair of rectification matrices $\mathbf{H}_{\mathrm{RC}}$ and $\mathbf{H}_{\mathrm{RP}}$ of the current and previous camera frames. Now, we obtain the plane plus parallax homography in image coordinates which warps the previous camera frame to the current one taking into account rectification:

$$\mathbf{H}_{\mathrm{RC}}^{-1}\,\mathbf{H}_{G}^{-1}\,\mathbf{H}_{\mathrm{RP}} \tag{5}$$

Finally, the transformation with which the map has to be updated may be written as:

$$\mathbf{H}_{\mathrm{BEV}} \cdot (\mathbf{H}_{\mathrm{RC}}^{-1}\,\mathbf{H}_{G}^{-1}\,\mathbf{H}_{\mathrm{RP}}) \cdot \mathbf{H}_{\mathrm{BEV}}^{-1} \tag{6}$$

Intuitively, we first warp the map to the previous image using $\mathbf{H}_{\mathrm{BEV}}^{-1}$ and then transform the map using (5) according to the camera motion. Finally, we warp the transformed map from image coordinates back to the bird's eye view using $\mathbf{H}_{\mathrm{BEV}}$.

*Cumulative map update* We need to update the likelihoods of the occupancies of the map, according to the current obstacle and ground plane silhouettes. For that, we transform the silhouette-points, given in image coordinates of the current rectified frame, into the bird's eye view using

$$\mathbf{H}_{\mathrm{BEV}} \cdot \mathbf{H}_{\mathrm{RC}}^{-1} \tag{7}$$

In general, the warped silhouette points in the bird's eye view have a specific covariance, which depends on the accuracy of disparity estimation and the uncertainty of camera locations. We take this uncertainty into account and for every silhouette point we update the occupancy of the surrounding cells according to that uncertainty. The uncertainty is spread using a Gaussian kernel around each warped silhouette point. The variance of the Gaussian in $x$-direction is constant, while the variance in $y$-direction depends on the depth uncertainty (this is, because we defined that the orientation of the vehicle always points into the negative $x$-direction in the bird's eye view). Taking the uncertainty into account will help in difficult situations (for example, when using top-view cameras) and will make the model independent of quantization.

In practice, we implemented these map updates very efficiently: these assumptions lead to a rectangular region of the bird's eye view, in which the map must be updated. We implemented these operations very efficiently by using saturated additions and by scaling precompiled Gaussian kernels.

## 4 Applications

In the following, we demonstrate three different customer-oriented functions. These applications utilize the proposed processing pipeline but use disparity information in slightly different ways.

### 4.1 Automatic parking slot detection

For the detection of parking slots we use the cumulative map of the environment and compute a global silhouette there. Since the orientation of the vehicle is defined to be aligned with the $x$-axis of the bird's eye view, we compute one silhouette point for every column of the map as shown in Fig. 7. For that, we first determine a line which runs through the position of the camera in the bird's eye view and is aligned with the orientation of the vehicle (we assume that parking slots are either parallel or orthogonal to the orientation of the vehicle). As indicated in Fig. 7, in every image column $x$ of the cumulative map, we find the closest occupied cell to that line (a cell is occupied, if its likelihood is greater than a specific threshold) and store the distance in a global distance profile $S(x)$.

Based on this global distance profile $S$, we define that a parking slot is an interval $[x_1, x_2]$ which fulfils

$$S_- < S(x) < S_+ \quad \forall x \in [x_1, x_2] \tag{8}$$

We use the parameters $S_-$ and $S_+$ to constrain the size of the parking slot (as described below). In practice, we use a small seed interval in which we check (8) and then we grow the interval to the left and right.

*Orientation* To detect cross and parallel parking slots, we perform the check using (8) with different parameters:

- To detect parallel slots we set $S_-$ and $S_+$ to 2 and 4 m.
- For cross parking slots we use 4 m and $\infty$ for $S_-$ and $S_+$.

*Depth* To compute the depth, we pick the camera frame from the history when the camera was at position $x_1$ (i.e. the nearest neighbor). Then we select two subsets of the segmented obstacle disparities:

1. The set of close disparities: all disparities of the obstacle segmentation whose silhouette position $x$ fulfils $x < x_1 - 25\,\text{cm}$.
2. The set of far disparities: all disparities of the obstacle segmentation whose silhouette position $x$ fulfils $x > x_1 + 25\,\text{cm}$.

We compute the mean values of these disparity values and compute the depth of the parking slot as the difference of these two depth values. We perform the same computations for the position $x_2$ and use the maximal value as the final depth for the parking slot.

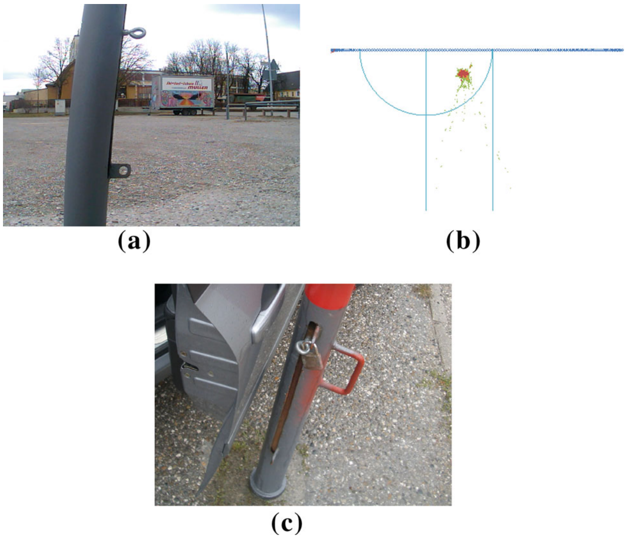
*Validation* We use several rejection cues for the validation of a free space. Every free space has associated a specific interval of camera frames in which the free space is (partially) visible. We reject a free space as a parking slot, if

- the *steering angle* exceeds defined thresholds, or
- the *velocity* is greater than a defined speed, or
- the variance of the location of the *horizon line* (of the ground plane segmentation) exceeds defined thresholds.

### 4.2 Collision warning

The sportive exterior design of cabriolets and coupés makes the pivoting ranges of doors difficult to observe, because often the door-edge is located behind the driver's head. Small objects or unthoughtful opening of a door may lead to expensive minor damage. The goal of this application is to prevent such damage by checking for possible collisions with static objects in the pivoting ranges of the doors. If such a collision is detected, occupants may be warned visually, acoustically or even haptically. In practice, we keep the history of disparity maps and segmentations and perform these checks in the moment when the vehicle stops (Fig. 8).

We realized the collision detection in a slightly different way, however with the same algorithmic components. Instead of using the silhouettes generated from the obstacle segmentation, we directly use all disparities of the obstacle segmentation. For every disparity, we compute the position on the ground plane (by solving [4] for $y$] and transfer that point to the bird's eye view using (7). Then, we update the corresponding cell of a cumulative map at this position. This time however, we do not use a Gaussian kernel, so we increment only one single cell per obstacle disparity. The collision detection is then performed by analyzing defined regions of the map (for example, the regions corresponding to the pivoting ranges of the doors). If a region contains a cell whose counter is greater than a specific value, a warning is issued.



(a) (b)

(c)

**Fig. 8** Collision detection within the pivoting area of the right front door: **a** the car passes by a pole and **b** in the moment when the vehicle stops, a local reconstruction created from the history of disparity maps is analyzed for possible collisions (**c**)

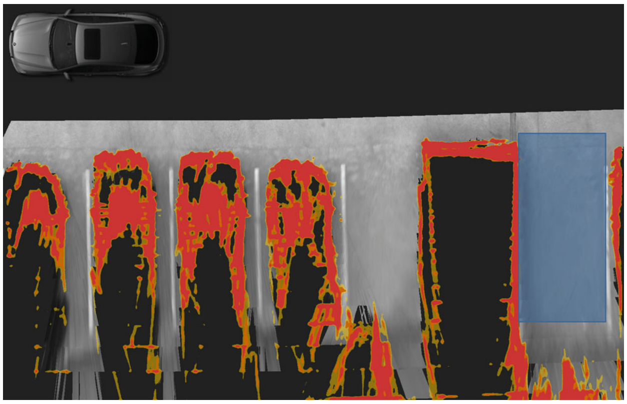


**Fig. 9** Augmented parking: the host vehicle, the detected parking slot and surrounding obstacles are displayed over the image of the ground plane

### 4.3 Augmented parking

Once a parking slot has been detected, the question is how to visualize the actual location to the driver. One idea is to generate a bird's eye view displaying the image of the ground plane with the position of the host vehicle and obstacles overlaid as shown in Fig. 9.

Also this application can be implemented with the described components. We use a slightly modified map: at this time, it represents a specific region of the ground plane, which we want to visualize. Every cell of the map holds a pair of values $(c, n)$, with $c$ being an intensity value and $n$ being a counter. In the beginning, all cells of the map are initialized to $(0, 0)$. To obtain the optimal quality of this image-based rendering, we use backward-warping: we iterate over all cells of the map and compute the location in every relevant camera frame using the inverse transformation of (7) and by chaining the history of plane plus parallax homographies (6). In practice, we keep a history of camera frames and transformations in memory. For such an image pixel $\mathbf{p}$, we check whether it is part of the ground plane segmentation and obtain the intensity value $c'$ from the camera frame using bilinear interpolation. Let the current cell of the map be $(c, n)$, then we update it according to

$$(c,\ n) \mapsto \left(\frac{c\,n + c'}{n+1}, n+1\right). \tag{9}$$

Once all cells have been visited, we render the host vehicle and visualize obstacles using the silhouettes as described above (see Sect. 3.9).

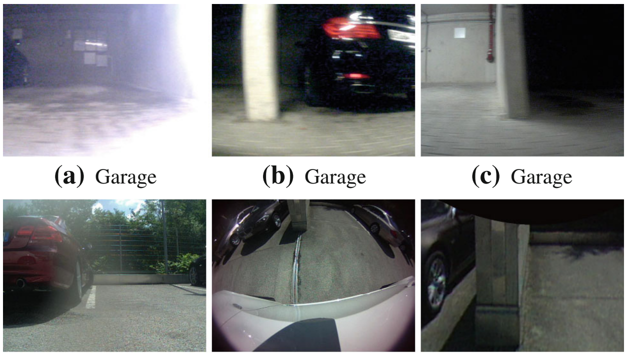
## 5 Results

In the following section we present practical results of our system, measured on the application level. We concentrated mainly on the performance at daylight conditions using the side-view camera, but we also performed tests with the top-view camera and at different environmental conditions.

For the parking slot detection, the accuracy and the false detection rates (false positives and missed slots) are most important. For assessing the collision warning, we measured only the detection rates. And finally, for the augmented navigation we present pictures of the image based rendering.

### 5.1 Methodology

Our goal was to test our method extensively on a large set of relevant scenarios. In our case, this included quantifying the performance in terms of measurement accuracy and detection rates of different algorithms and parameterizations. Especially for false detection rates, the number of test cases should be quite large. Thus, practical experimentation is usually very time-consuming and environmental influences make results of different test sessions hard or even impossible to compare. Further, at every test case, ground truth data must be acquired which requires time consuming labeling. Due to these reasons, we decided to resort to software-in-the-loop techniques [34,37]: we recorded videos synchronized with data from the vehicle (e.g. odometry information). This allowed us to execute different configurations of our method on exactly the same set of scenarios. After recording the sequences we associated ground truth measurements to them. In every video we mark all frames where a parking slot is, at least, half visible. To each interval where a parking slot is visible we associate its ground truth size measured using the laser distancemeter. We compare the results our and those of other methods against ground truth data and identify false detections.



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

**(d)** Side-view image with a dirty lens **(e)** Top-view image **(f)** Rectified top-view image

**Fig. 10** Examples of camera images used: **a**–**c** sequences recorded in the garage have very difficult lighting conditions due to inhomogeneous illumination (sun and neon light), **d** an image of the side-view camera with a dirty lens and **e** a top-view image before and **f** after undistortion

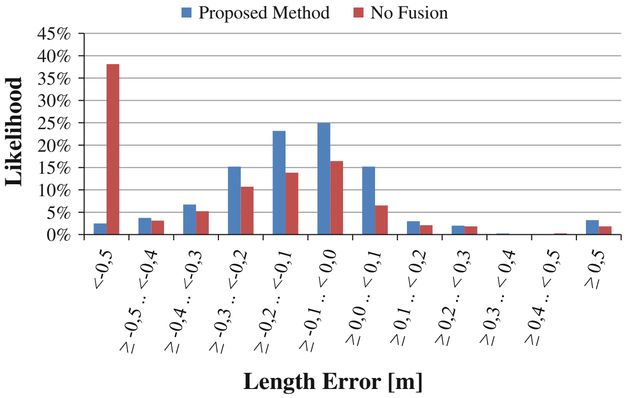


**Fig. 11** The distribution of measurement errors of our method with and without temporal fusion [32] at daylight conditions using the side-view camera. The fusion removes many outliers and therefore, less false matches are accumulated in the map. Hence, the measurements are more accurate

The whole database contains over 120 GB of uncompressed video data (approximately 2 h) of the side-view and top-view cameras and contains sequences of 718 parking slots. When recording the sequences we varied different parameters:

- Velocity: from idle speed up to 35 km/h, either with constant or varying speeds (by braking and accelerating).
- Yaw-rate: in most cases we drove on a straight line, but in some sequences we modified the steering angle (e.g. driving on a sinuous line).
- Slot length and depth (parallel and cross parking slots): lengths varied between 1.8 and 13.2 m; depths between 2 and 10 m. We configured the system in a way such that a parking space should provide at least an area of 5.5 × 2.5 m.
- Illumination: daylight (sunny, cloudy or rainy) and a subterranean garage (inhomogeneously illuminated by sun and neon light; see Fig. 10 for example images).
- Dirtiness of the lens: the database also contains a few recordings where the lens was dirty (see Fig. 10 for example images).

We define that the *length* of a slot is the dimension being roughly parallel to the direction of the host vehicle and that the *depth* is orthogonal to the length. We measured every parking slot manually using a *Leica DISTO classic* laser distancemeter, which is very accurate in practice. When measuring these dimensions by hand, we could only achieve an accuracy of ±5 cm. The length is the minimal longitudinal size between two obstacles, but the depth is more difficult to define: in general, the depth is the distance between a close and a far boundary, both delimiting the parking space into which the vehicle must fit. The far boundary is usually given by some structural installations (the curb, fences, walls or other vehicles) but it may even not be present (then it may be given by ground markings or implicitly by a change in the asphalt). The close boundary is even more complicated. In some cases it is given by ground markings or a change in the asphalt, but in many other situations such indications are missing and often it is defined by a "thought line" which is tangential to other parking vehicles or may even depend on complex scene understanding. Therefore, we followed a two-fold strategy for the evaluation:

- The accuracy of length-measurements is evaluated using the whole database and the associated ground truth measurements.
- The accuracy of depth-measurements is evaluated on a smaller set of parking slots, where the far boundary was given by a wall or the curb. For the close boundary of a parking slot we used the maximum depth of the bounding obstacles to the left and right.

### 5.2 Analysis of the stereo pipeline

The most important part of our method is the disparity computation stage which includes the single baseline stereo method and the temporal disparity fusion step. Figure 11 shows that the temporal fusion is very important for accurate measurements and that the amount of outliers is very critical for the overall performance (see also Fig. 12). If no temporal fusion is performed, the disparity maps contain significantly more outliers and imprecise object boundaries. These characteristics lead to errors in the cumulative map and result in increased measurement errors. This becomes also obvious in Fig. 12 by an increase in the false detection rates. The temporal fusion makes the whole system much more reliable.

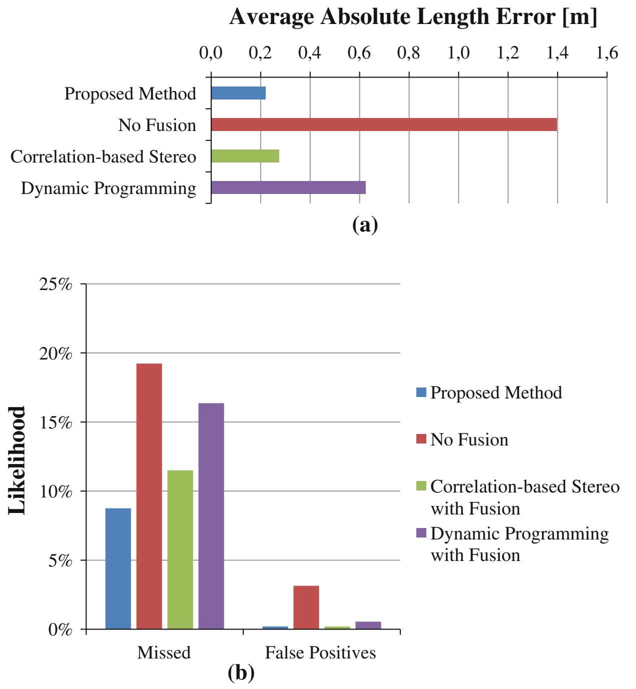


**Fig. 12** The average measurement error (**a**) and false detection rates (**b**) of our method with different variations of the stereo pipeline at daylight conditions using the side-view camera: dense stereo [30] with temporal fusion [30,32] without fusion, traditional real-time stereo [8,16] with fusion [32] and dynamic programming [23] with fusion [32]

In Fig. 12, our real-time stereo method [30] with temporal fusion [32] (proposed solution) is slightly better than traditional real-time stereo [8,16] with fusion [32] and is at the same time twice as fast. We also included dynamic programming [23] (with temporal fusion enabled) in our evaluation, because it also belongs to the class of efficient methods. However, the well known problem of streaking effects [23] limits the practical use. The different configurations of the stereo pipeline effect also the detection rates (see Fig. 12). The amount of outliers is a direct indicator for the robustness and thus a measure for the customer value.

### 5.3 Performance when using top-view cameras

Our test vehicle was equipped with side- and top-view cameras. When we recorded the sequences, we recorded the video streams of both cameras simultaneously and thus, we were able to determine the performances in exactly the same conditions. Figure 13 shows the performance when using side- and top-view cameras: both systems offer the same potential of accuracy (in 40% of all measurements the error was between ±10 cm, for both systems) and it is more likely that the estimated size of a parking slot is too small, which is a desirable property. However, it is more robust to use side-view cameras: the disparity maps computed from rectified images of top-view cameras contain much more errors than the ones from side-view cameras. This is due to the wide angle lenses used (less FOV and lower resolution in the region of interest) and a worse signal-to-noise ratio (SNR).

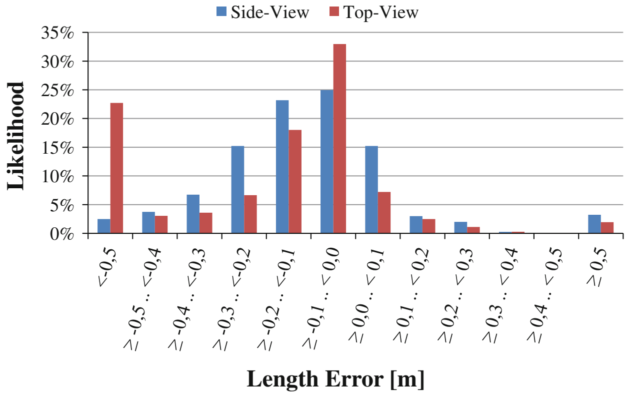


**Fig. 13** The performance of our proposal with side- and top-view cameras at daylight conditions. The *bars* show the distribution of measurement errors

### 5.4 Analysis of environmental influences

We also recorded scenes with different environmental influences:

- *Daylight* sequences recorded at daytime (1 h after dawn and 1 h before dusk) with sunny, cloudy or rainy conditions.
- *Dirty lens* we also recorded videos where the lens had (natural) dirt on it (composed of the remains of a dead insect; see Fig. 10 for example images).
- *Garage* sequences recorded in a garage with artificial lighting. In these scenes, lighting conditions were very difficult (see Fig. 10 for example images). To some extent, this is due to the fact that sunlight is sometimes visible and the exposure control of the camera adapts permanently and switches between day- and night-mode.

Figure 14 shows that the performance of the system is different for these use cases. Large mismeasurements are more likely in difficult situations but one important property always remains: it is very unlikely that the parking slot is measured too large. In only 5% of all cases, the length of the parking slot is over-estimated by more than 20 cm. This implies that if the system has found a free space, there is a very high reliability that the vehicle actually fits into it.

### 5.5 Comparison to other methods

We compare our parking slot detection application to a feature-based method [35,36] and a solution based on an ultrasonic sensor [20]. For the camera-based approaches, we used

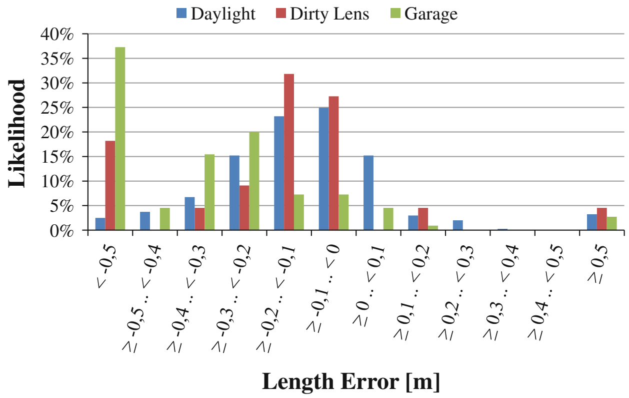


**Fig. 14** The distribution of measurement errors with different environmental influences: scenes recorded at daylight (sunny, cloudy or rainy), with a dirty lens and in a garage with very difficult lighting conditions

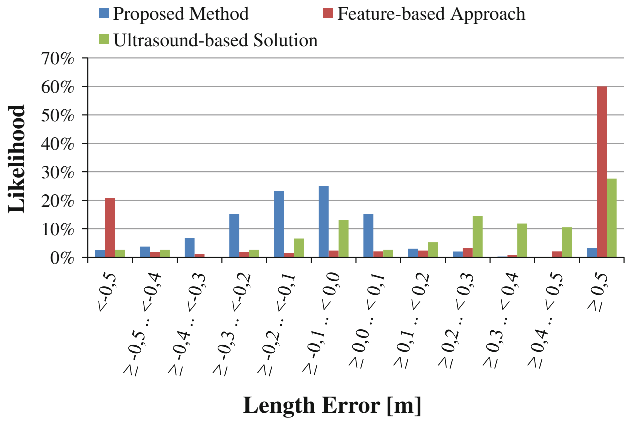


**Fig. 15** The distribution of measurement errors of different methods at daylight conditions: we compare our method based on dense motion-stereo to a feature-based method [35,36] and a solution based on an ultrasonic sensor [20]

daylight sequences from the side-view camera. The feature-based method failed completely on the top-view videos. The measurement errors of the feature-based approach are much larger than the errors of our dense method. Only a few measurements have an absolute error less than 0.5 m (see Fig. 15). Figure 16 shows the same error distributions with coarser intervals and shows that our approach achieves a very high accuracy. The reason for the large errors of [35,36] lies in the fact that the feature extractor often fails to detect features on object boundaries. In many cases, features are mainly detected at rims and license plates and this easily introduces an error of roughly 2 m per parking spot. In other cases, features are completely missing at textureless objects like walls and this leads to much larger errors. Further, sometimes features are matched incorrectly (for example, at repetitive structures) and this usually results in measurements that are too small.

The approach based on the ultrasonic sensor [20] was only evaluated on parallel parking slots (in total, 114 test cases), because due to a limited depth range, it does not detect cross parking slots. This method had mainly problems when the obstacles had a complex 3D structure. For example, bushes are not detected reliably and curved object boundaries led to large errors. Also trailer hitches and small objects seem to negatively impact the measurement accuracy. There were a few false positives (only 4 cases), but probably due to measurement errors, the number of false negatives (*misses*) was with 28% quite high (in 32 cases). However, the overall impression is that it is a very reliable system, whose performance is invariant to illumination.

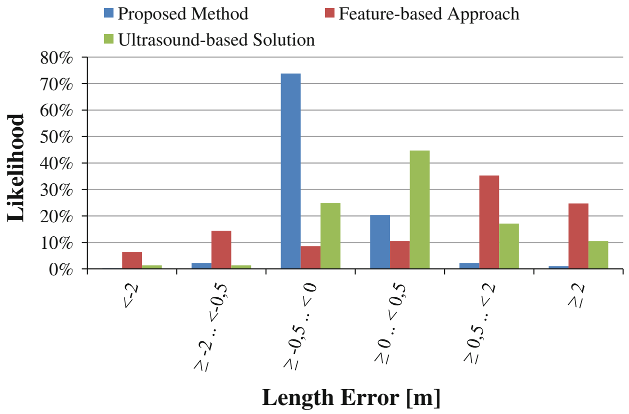


**Fig. 16** The distribution of measurement errors of different methods at daylight conditions: we compare our method based on dense motion-stereo to a feature-based method [35,36] and a solution based on an ultrasonic sensor [20]

## 5.6 Accuracy of depth measurements

The accuracy of depth measurements is difficult to compare. The feature-based system of [35,36] does not directly determine the depth of parking slots, so we cannot compare to them. The ultrasound-based solution [20] has a limited depth range (in practice, the maximum depth is between 3 and 4 m) and so we evaluated it only on parallel parking slots.

*Theoretical discussion* In our approach, the depth $Z$ is computed from a disparity $d$. The relative depth-error is therefore given by

$$Z_{\text{err}} = \frac{d_{\text{err}} \cdot Z}{f \cdot B + d_{\text{err}} \cdot Z} \tag{10}$$

with the focal length $f$, the baseline $B$ and the error of disparity estimation $d_{\text{err}}$. In practice, the quantity $f\,B$ is much larger than $d_{\text{err}} Z$, and therefore the error $Z_{\text{err}}$ is approximately a linear function of $Z$. This means that for our system, range measurements using large values for $Z$ are most challenging. Thus, to evaluate our proposal, we used large values for $Z$ (i.e. only cross parking slots), in order to obtain an upper bound for the error.

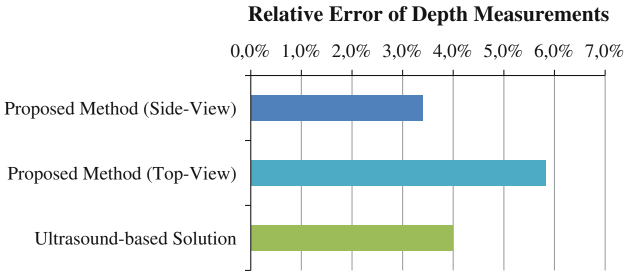


**Fig. 17** The relative error of depth measurements of different approaches: for the ultrasound-based approach, the depth of the parking spots was on average 2 m and for our method we used only cross parking slots with depths between 5 and 6 m (the distance between the camera and the far boundary was between 6 and 7 m)

Figure 17 shows the results of the depth accuracy: for the ultrasound-based approach, the depth of the parking spots was on average 2 m and for our method we used only cross parking slots with depths between 5 and 6 m (the distance between the camera and the far boundary was between 6 and 7 m). Notably, if we assume that $d_{\text{err}} = 0.125$ then the theoretical error for a depth measurement at 6.5 m is at 3.5% (for our side-view camera). The measurements of [20] were relatively accurate in practice. For the motion-stereo approach, the biggest challenge is given by repetitive structures: in one case the absolute error was 48 cm (10%) which was due to mismatches at a fence (repetitive structure). Also the characteristics of the camera play an important role: the worse SNR of the top-view camera nearly doubles the error.

### 5.7 Performance of the collision warning

We also evaluated the collision warning application. We tested it on 123 obstacles (see Fig. 18 for examples) and checked whether the warning was correct or not: if there is an object in the pivoting range of a door, a warning should be issued – if it is save to open all doors, then the warning should be suppressed. Based on these tests we determined the detection rates (see Fig. 19). In 100 cases the decision for the warning was correct (i.e. issuing the warning or not; see *Correct Detection*). In 8 cases the location of the obstacle was estimated too inaccurate (*Obstacle Missed*) and in 3 cases,

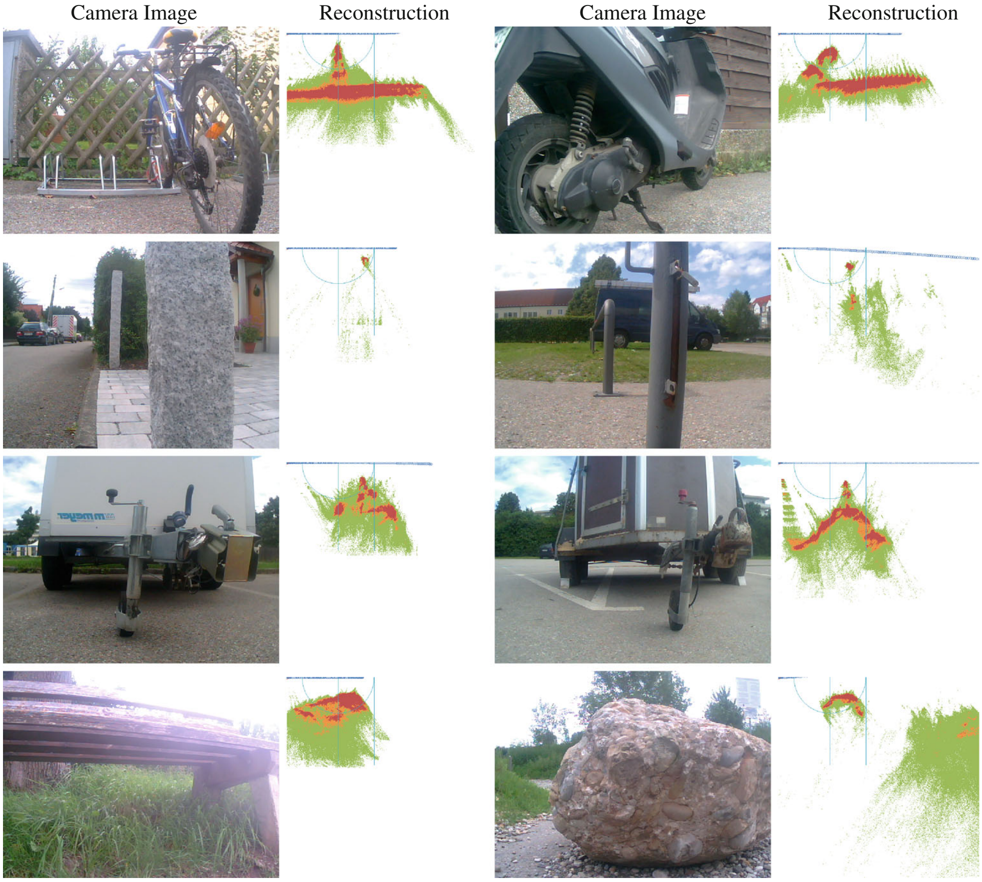


**Fig. 18** Examples for the Collision Warning: we present one camera image and the reconstruction obtained from segmented disparity maps. The objects (linewise from *left* to *right* and from *top* to *bottom*): a bicycle (no warning), a motorcycle (no warning), a pole made of stone, a pole made of steel, trailer 1, trailer 2, a bank and a stone

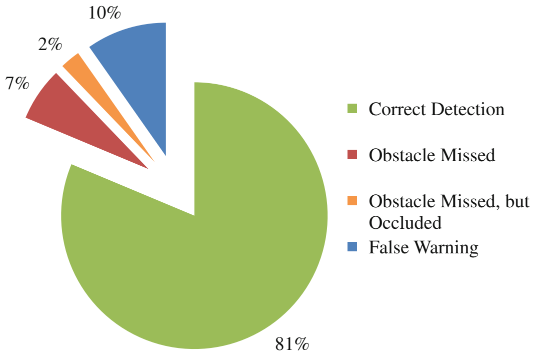


**Fig. 19** The performance of the collision warning at daylight conditions. We evaluated whether the system warned correctly against the presence of obstacles in the pivoting ranges of doors. In most cases, the decision upon the warning was correct (*Correct Detection*). In other cases, the position of the obstacle was estimated falsely outside or inside of pivoting ranges (*Obstacle Missed* or *False Warning*), or the collision-relevant part of the object was outside the FOV of the camera (*Obstacle Missed, but Occluded*)

the critical part of the obstacle was out of the FOV (*Obstacle Missed, but Occluded*; e.g. an attachable trailer and in one case the door would have hit a pole at a very high position). A *False Warning* happened in 12 cases: there was always an obstacle present, but the system falsely estimated that a collision is possible.

The collision detection works well for most objects, even if they are lowly textured, mainly because of our robust depth computation. The temporal fusion allows us to retain disparities with low confidence in binocular stereo matching because the fusion will remove inconsistent matches. In general it is of course important that the obstacles are within the FOV of the camera to be detected. For very thin objects (for example, fences made of wire mesh) it happens that our fast stereo processing is not able to recover them, because the projected size in the image is smaller than the size of the window used for matching. This is a well known problem of correlation-based stereo matching and might by improved by using pixelwise stereo matching [5,7,13]. However, it is unlikely that such thin objects result in serious damage to the car.

### 5.8 Qualitative impressions of augmented parking

We present generated bird's eye views from different sequences in Figs. 20 and 21. Challenging were situations in which the ground was not completely flat: in the bottom example of Fig. 20 small distortions are visible in the rendered image. Also reflections lead to artifacts which may be irritating in the first place. Lastly, since we did not introduce photometric registration, coloring is often inconsistent within a rendering.

### 5.9 Execution times

The execution times of the different processing steps can be found in Table 1. Dense stereo matching and the temporal fusion of disparity maps consume most time. We partitioned the whole system into several threads:

1. Acquisition-thread: acquires video frames from the camera and performs the undistortion.
2. Stereo-thread: runs the dense stereo matching algorithm.
3. Fusion-thread: performs the temporal fusion of disparity maps.
4. Interpretation-thread: executes the segmentations and the applications.
5. Visualization-thread: cares about the user-interface.

The collision warning and the augmented parking are not real-time, which is tolerable: the collision detection is only run in the moment when the vehicle stops. The latency of the augmented parking can be easily reduced to a minimum, if processing is started as soon as a parking slot is found. Further, at the augmented parking most time is spent for perspective warping and might be accelerated with dedicated hardware. Usually, we use development settings for the interpretation thread, which includes many debug visualizations. In this case, the interpretation may consume up to 20 ms, but by turning off all unnecessary outputs it can be tweaked to 3 ms (including the segmentation). Moreover, we assigned the highest priority to the fusion- and acquisition-threads, the stereo- and interpretation-threads ran with lower priority and the visualization-thread received the lowest priority.

The whole system is implemented in C++ (using Microsoft Visual C++ 9.0) and runs on Microsoft Windows. We achieved the best performance on a quad-core CPU with 2.53 GHz (Intel Core2 Extreme Q9300) and also on a dual-core CPU with 2.66 GHz (Intel Core2 Duo E8200). However, to allow real-time operation on the dual-core, we had to disable sub-pixel interpolations in the stereo and fusion algorithms. Further, time-critical parts are implemented using SIMD[2] instructions. We also performed tests with another mobile dual-core processor (in particular, an Intel T7600 with 2.33 GHz), but this CPU was not sufficient for real-time processing.

## 6 Discussion

Some of the challenges one has to face are shearing effects when using rolling shutter cameras, smearing with global shutter, and misalignments whenever interlaced images are involved. Moreover, the current cameras suffer from weak

[2] Single Instruction, Multiple Data: in particular, the SSE2 instruction set.

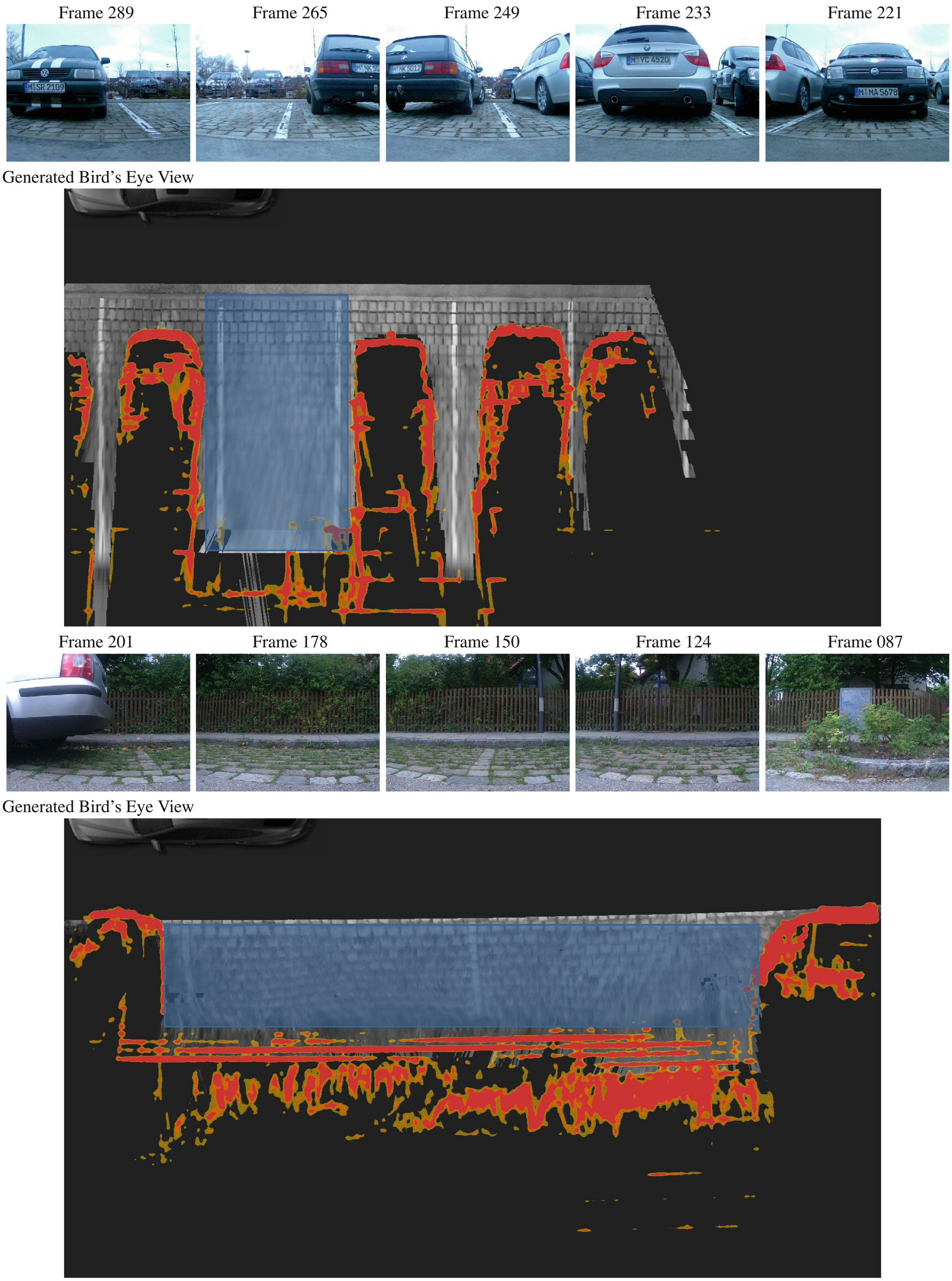


**Fig. 20** Examples for the bird's eye views on different sequences: we show selected camera frames and the generated bird's eye view. In the *bottom* example, the ground plane was not flat and caused the vehicle to pitch and movements in $z$-direction. This leads to distortions in the rendering

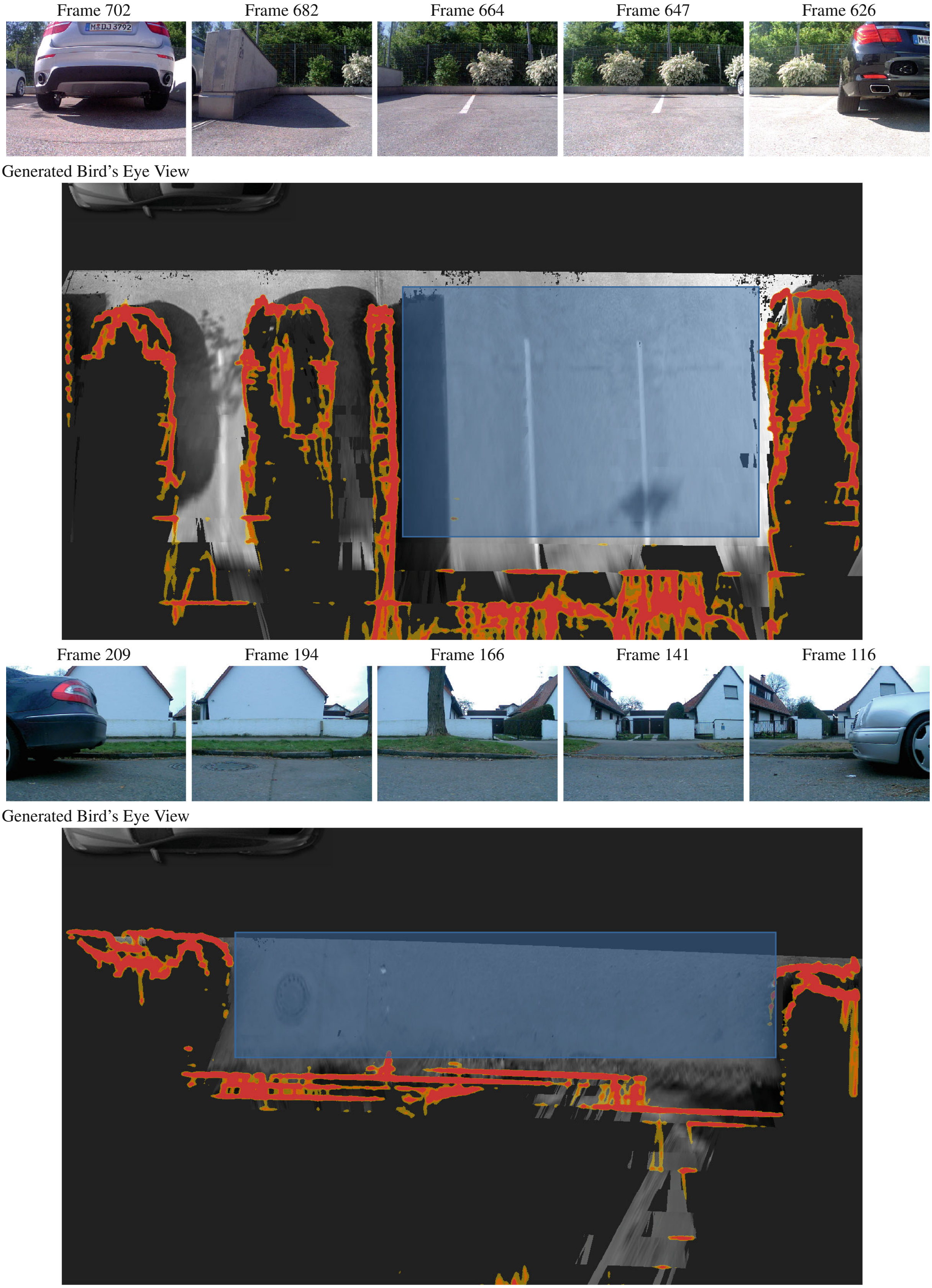


**Fig. 21** Examples for the bird's eye views on different sequences: we show selected camera frames and the generated bird's eye view. The *top row* is a good example for difficult lighting conditions

**Table 1** Execution times of the different processing steps on different processors: on the E8200 sub-pixel interpolation and many debug visualizations were disabled

| Step | Time: Q9300<br>2.53 GHz, 4 cores<br>(ms) | Time: E8200<br>2.66 GHz, 2 cores<br>(ms) |
|---|---|---|
| Undistortion | 2 | 2 |
| Rectification | 0.5 | 0.5 |
| Stereo matching | 15 | 12 |
| Temporal fusion | 31 | 29 |
| Segmentation | 2 | 2 |
| Parking slot detection | 4–11 | 1–9 |
| Collision warning | 120 | 80 |
| Augmented parking | ≈10,000 | ≈10,000 |

The collision warning and augmented parking are only run if required (i.e. event-triggered)

sensitivity in low light conditions. If an application is expected to work at night, some kind of active illumination would be required. This would involve additional costs, installation space, and often leads to legal conflicts in some countries. However, since parking maneuvers are performed with relatively low speeds and having upcoming high dynamic range imagers in mind, weaknesses of current technology are to some extent tolerable. Moreover, the temporal fusion of disparity maps turned out to be highly effective against these issues.

Furthermore, it must be noted that there are mathematical limitations for monocular systems in non-rigid scenes: in certain cases, if the motion-vectors of the host vehicle and an obstacle are collinear then the motion of the obstacle is hard or impossible to recover without additional knowledge or interpretation of data. In the worst case, this leads to wrong distances. However, our focus lies on comfort functions: the detection of parking slots and avoidance of minor damage is not safety critical. Therefore, we assume that the scene is static and there are also techniques available to detect moving obstacles by introducing other constraints [12].

Objects that move parallel to the vehicle and in the same direction happen to be less critical, because their estimated distance is actually larger than the true distance. To some extent our temporal fusion helps with moving objects, because parts that move inconsistently over time (for example, the feet of pedestrians) are getting replaced by the static background. Most critical are objects that move parallel to the vehicle and in opposite direction, because their estimated distance is closer than in reality. However, in the worst case, this results in a missing detection of a free space or a false warning.

Compared to the feature-based approach [35,36], our approach based on dense motion-stereo has important advantages:

- Redundancy of measurements due to overlap of images: redundancy can be systematically utilized to detect wrong measurements and to improve accuracy.
- Higher detection rate of obstacles: while feature-based approaches are usually specialized for a specific class of features (e.g. corners and edges), problems arise if such features are absent (e.g. regions with low texture). In our experiments the dense approach turned out to be much more flexible and detects almost all obstacles.
- Higher measurement accuracy of the measured dimensions of parking slots: high accuracy requires a precise detection of object boundaries. Feature-based approaches may miss detecting features which lie exactly on object boundaries. Such behavior introduces large measurement errors.

## 7 Conclusion

This paper presents a generic method for environment modeling based on dense motion-stereo and demonstrates its flexibility using different applications for parking assistance. Our processing pipeline exploits the principle of *dense* motion-stereo, where for every pixel in every camera image we determine a depth using real-time stereo. Since the rectification cannot be computed accurately, we introduce an extension to our real-time stereo method [30], in order to make matching robust against distortions of the epipolar geometry. After stereo matching, the history of disparity maps is fused probabilistically to obtain for every camera image the most probable disparity map that exposes a minimum amount of outliers. In every fused disparity map we detect the ground plane, obstacles and from that a silhouette which limits the free space. Then, we combine all these partial silhouettes so that over time a global model of the environment is created incrementally. Within this model, we perform an *Automatic Parking Slot Detection*.

Further, we use the disparity maps to obtain a local 3D reconstruction of specific regions of interest (for example, the pivoting ranges of doors). Using such a local 3D reconstruction, we perform a collision analysis and, if necessary, issue a *Collision Warning* to occupants to prevent minor damages. Another application is *Augmented Parking* and uses image-based rendering to compute a virtual bird's eye view to visualize the positions of the host vehicle, obstacles and the parking slot to the driver.

The accuracy and reliability of our approach is demonstrated via exhaustive experimentation and comparison to solutions based on an ultrasonic sensor and feature-based matching. The results show clearly that our proposal achieves high reliability, measures accurately and is very flexible.

**Acknowledgments** This work is supported by the BMW Group.

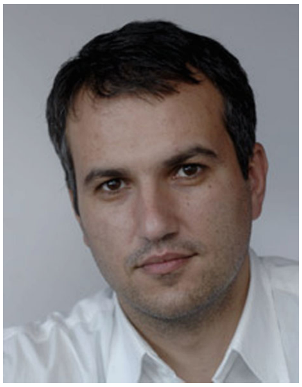
## References

1. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. Int. J. Comput. Vis. **12**(1), 43–77 (1994)
2. Brunton, A., Shu, C., Roth, G.: Belief propagation on the gpu for stereo vision. In: Canadian Conference on Computer and Robot Vision, pp. 76–81 (2006)
3. Devernay, F., Faugeras, O.D.: Straight lines have to be straight. Mach. Vis. Appl. **13**(1), 14–24 (2001)
4. Faugeras, O., Hotz, B., Mathieu, H., Viéville, T., Zhang, Z., Fua, P., Théron, E., Moll, L., Berry, G., Vuillemin, J., Bertin, P., Proy, C.: Real time correlation-based stereo: algorithm, implementations and applications. Tech. Rep. RR-2013, INRIA (1993)
5. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. In: CVPR (2004)
6. Fintzel, K., Bendahan, R., Bougnoux, S.: 3d parking assistant system. In: Proceedings of IEEE Intelligent Vehicles Symposium, pp. 881–886 (2004)
7. Hirschmüller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. In: CVPR, pp. 807–814 (2005)
8. Hirschmüller, H., Innocent, P.R., Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. Int. J. Comput. Vis. **47**(1–3), 229–246 (2002)
9. Jung, H.G., Kim, D.S., Yoon, P.J.: Parking slot markings recognition for automatic parking assist system. In: Proceedings of IEEE Intelligent Vehicles Symposium, pp. 106–113 (2006)
10. Jung, H.G., Kim, D.S., Yoon, P.J., Kim, J.: Light stripe projection based parking space detection for intelligent parking assist system. In: Proceedings of IEEE Intelligent Vehicle Symposium (2007)
11. Kämpchen, N., Franke, U., Ott, R.: Stereo vision based pose estimation of parking lots using 3d vehicle models. In: Proceedings of IEEE Intelligent Vehicle Symposium (2002)
12. Klappstein, J., Stein, F., Franke, U.: Monocular motion detection using spatial constraints in a unified manner. In: Proceedings of IEEE Intelligent Vehicle Symposium, pp. 261–267 (2006)
13. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions using graph cuts. In: ICCV, pp. 508–515 (2001)
14. Lu, Y., Zhang, J.Z., Wu, Q.M.J., Li, Z.N.: A survey of motion-parallax-based 3-d reconstruction algorithms. IEEE Trans. Syst. Man Cybern. C Appl. Rev. **34**(4), 532–548 (2004)
15. Merrell, P., Akbarzadeh, A., Wang, L., Frahm, J.M., Yang, R., Nistér, D.: Real-time visibility-based fusion of depth maps. In: ICCV, pp. 1–8 (2007)
16. Mühlmann, K., Maier, D., Hesser, J., Männer, R.: Calculating dense disparity maps from color stereo images, an efficient implementation. Int. J. Comput. Vis. **47**(1–3), 79–88 (2002)
17. Nistér, D.: Frame decimation for structure and motion. In: 3D Structure from Images-SMILE 2000, LNCS, pp. 17–34. Springer, Berlin (2001)
18. Park, W.J., Kim, B.S., Seo, D.E., Kim, D.S., Lee, K.H.: Parking space detection using ultrasonic sensor in parking assistance system. In: Proceedings of IEEE Intelligent Vehicle Symposium, pp. 1039–1044 (2008)
19. Pohl, J., Sethsson, M., Degerman, P., Larsson, J.: A semi-automated parallel parking system for passenger cars. Proc. Inst. Mech. Eng. Part D J. Automob. Eng. **220**, 53–65 (2006)
20. Pruckner, A., Gensler, F., Meitinger, K.H., Gräf, H., Spannheimer, H., Gresser, K.: Der parkassistent—ein weiteres innovatives fahrerassistenzsystem zum thema connecteddrive aus der bmw-fahrzeugforschung. In: Braunschweiger Symposium (2003)
21. Rosenberg, I.D., Davidson, P.L., Muller, C.M.R., Han, J.Y.: Real-time stereo vision using semi-global matching on programmable graphics hardware. In: SIGGRAPH 2006 Sketches (2006)
22. Schanz, A.: Fahrerassistenz zum automatischen Parken. No. 607 in 12. VDI Verlag (2005)
23. Scharstein, D., Szeliski, R., Zabih, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**, 7–42 (2002)
24. Scheunert, U., Fardi, B., Mattern, N., Wanielik, G., Keppeler, N.: Free space determination for parking slots using a 3d pmd sensor. In: Proceedings of IEEE Intelligent Vehicle Symposium, pp. 154–159 (2007)
25. Song, K.T., Chen, H.Y.: Lateral driving assistance using optical flow and scene analysis. In: Proceedings of IEEE Intelligent Vehicle Symposium, pp. 624–629 (2007)
26. Stahl, W., Hoetzel, J.: Parktronic-system (pts), aktueller stand und ausblick. Tech. Rep. 1287, VDI-Berichte (1996)
27. Suhr, J.K., Bae, K., Kim, J., Jung, H.G.: Free parking space detection using optical flow-based euclidean 3d reconstruction. In: MVA, pp. 563–566 (2007)
28. Suhr, J.K., Jung, H.G., Bae, K., Kim, J.: Automatic free parking space detection by using motion stereo-based 3d reconstruction. Mach. Vis. Appl. **21**(2), 163–176 (2010)
29. Torr, P.H.S., Murray, D.W.: The development and comparison of robust methods for estimating the fundamental matrix. Int. J. Comput. Vis. **24**, 271–300 (1997)
30. Unger, C., Benhimane, S., Wahl, E., Navab, N.: Efficient disparity computation without maximum disparity for real-time stereo vision. In: BMVC (2009)
31. Unger, C., Wahl, E., Ilic, S.: Efficient stereo matching for moving cameras and decalibrated rigs. Intell. Veh. 417–422 (2011)
32. Unger, C., Wahl, E., Sturm, P., Ilic, S.: Probabilistic disparity fusion for real-time motion-stereo. Tech. rep., Technische Universität München (2010). http://campar.in.tum.de/Main/ChristianUnger
33. Vestri, C., Bougnoux, S., Bendahan, R., Fintzel, K., Wybo, S., Abad, F., Kakinami, T.: Evaluation of a vision-based parking assistance system. In: Proceedings of IEEE Intelligent Vehicle Symposium, pp. 131–135 (2005)
34. Wahl, E., Oszwald, F., Ruß, A., Zeller, A., Rossberg, D.: Evaluation of automotive vision systems: Innovations in the development of video-based adas. In: FISITA World Automotive Congress (2008)
35. Wahl, E., Strobel, T., Ruß, A., Rossberg, D., Therburg, R.D.: Realisierung eines parkassistenten basierend auf motion-stereo. In: 16. Aachener Kolloquium (2007)
36. Wahl, E., Therburg, R.D.: Developing a motion-stereo parking assistant at bmw. MATLAB Digest (2008)
37. Wahl, E., Zeitler, W.: Video-based driver assistance systems put to test: Comparison—evaluation—series production. In: 13th International Conference: Electronic Systems for Vehicles (2007)
38. Wang, L., Liao, M., Gong, M., Yang, R., Nister, D.: High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In: Proc. Int. Symp. 3D Data Proc., Vis., and Transm. (3DPVT), pp. 798–805 (2006)
39. Xu, J., Chen, G., Xie, M.: Vision-guided automatic parking for smart car. In: Proceedings of IEEE Intelligent Vehicles Symposium, pp. 725–730 (2000)
40. Zach, C.: Fast and high quality fusion of depth maps. In: 3DPVT (2008)
41. Zhang, G., Jia, J., Wong, T.T., Bao, H.: Recovering consistent video depth maps via bundle optimization. In: CVPR, pp. 1–8 (2008)
42. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**, 1330–1334 (2000)

## Author Biographies

**Christian Unger** received his Dipl. Inf. (FH) degree in computer science from the Munich University of Applied Sciences in 2005 and the M.Sc. degree in computer science from the Technische Universität München, Germany in 2007. He is currently working on his PhD degree in computer science at the BMW Group, Munich, Germany and is responsible for stereo-based detection algorithms. His research interests include stereo, multi-view reconstruction, image-based rendering and advanced driver assistance.

**Eric Wahl** is developing camera-based advanced driver assistance systems at BMW Group, Germany. Formerly, he was research scientist at the Institute of Robotics and Mechatronics of the German Aerospace Center, where he studied object classification based on two- and three-dimensional data, pursuing which he received his PhD degree from the TU Munich in 2006. In his current function he focuses on efficient methods enabling environment interpretation for automated driving.



**Slobodan Ilic** is senior research scientist working at TU Munich, Germany. Since February 2009 he is leading the Computer Vision Group of the CAMP Laboratory at TUM. From June 2006 he was a senior researcher at Deutsche Telekom Laboratories in Berlin. Before that he was a postdoctoral fellow for one year at Computer Vision Laboratory, EPFL, Switzerland, from where he received his PhD in 2005. His research interests include deformable surface modeling and tracking, 3D reconstruction, realtime object detection and tracking, object detection and classification in 3D data. Slobodan Ilic is an Area Chair for ICCV 2011 and serves as a regular program committee member for all major computer vision conferences, such as CVPR, ICCV and ECCV as well as journals, such as TPAMI and IJCV. Besides active academic involvement, Slobodan has strong relations with industry and supervises a number of PhD students supported by industry, including BMW Group, MVTec, METAIO and Deutsche Telekom.