# Robust Image Processing for an Omnidirectional Camera-based Smart Car Door

CHRISTIAN SCHARFENBERGER
SAMARAJIIT CHAKRABORTY
GEORG FÄRBER

Over the last one decade there has been an increasing emphasis on driver-assistance systems for the automotive domain. In this paper we report our work on designing a camera-based surveillance system embedded in a "smart" car door. Such a camera is used to monitor the ambient environment outside the car – e.g., the presence of obstacles such as approaching cars or cyclists who might collide with the car door if opened – and automatically controls the car door operations. This is an enhancement to the currently available side-view mirrors that the driver/passenger checks before opening the car door. The focus of this paper is on fast and robust image processing algorithms specifically targeting such a smart car door system. The requirement is to quickly detect traffic objects of interest from gray-scale images captured by omnidirectional cameras. While known algorithms for object extraction from the image processing literature rely on color information and are sensitive to shadows and illumination changes, our proposed algorithms are highly robust, can operate on gray-scale images (color images are not available in our setup) and output results in real-time. We present a number of experimental results based on image sequences captured from real-life traffic scenarios to demonstrate the applicability of our algorithm.

General Terms: Embedded Computing, Algorithms, Performance, Image Processing

Additional Key Words and Phrases: Robust image processing, embedded computing, omnidirectional vision, road user extraction, driver assistance systems, smart car door

## 1. INTRODUCTION

Driver assistance systems are increasingly gaining importance in high-end cars. Examples of these include Lane Departure Warning System (LDW), Adaptive Cruise Control (ACC), Forward Collision Warning (FCW) and Blind-Spot Detection (BSD). While there are many safety-oriented driver-assistance systems that function when the car is moving, a number of accidents also happen while the car is stationary and one of its doors is being opened. A standard practice is to check the side-view mirrors of the car before opening the door. However, it is still fairly common for approaching cyclists to hit suddenly-opened car doors. In other words, many passengers check if there is an obstacle next to the door, but they do not pay sufficient attention to approaching obstacles like cyclists or other cars.

In this paper we report our work on designing a smart car door that is equipped with one omnidirectional camera on each side of the car. These cameras monitor the ambient environment outside the car and warn passengers – or car door users – about obstacles like approaching cars, bicycles or pedestrians. Collision avoidance systems use this information to control, stop or lock car door operations in order to avoid potential accidents.

Fig. 1 gives a high-level overview of our smart car door system. In [Strolz et al. 2008], we presented a generic control system for intelligent, actuated car doors with arbitrary degrees of freedom.

This work was supported by the BMW-Group Germany.
Author's addresses: C. Scharfenberger, Institute for Real-Time Computer Systems, TU Munich, Germany, c.scharfenberger@tum.de; S. Chakraborty, Institute for Real-Time Computer Systems, TU Munich, Germany, samarjit@tum.de; G. Färber, Institute for Real-Time Computer Systems, TU Munich, Germany, georg.faerber@tum.de

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

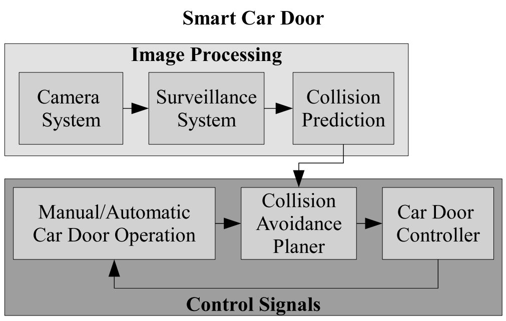
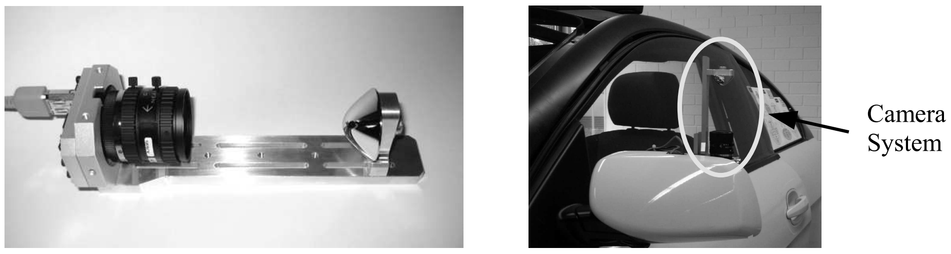


Fig. 1: The *Smart Car Door System*. Information from the camera and surveillance system are used as input to control, stop or lock car door operations in order to avoid accidents.



(a) (b)

Fig. 2: The omnidirectional camera system (a): A perspective camera focuses on a hyperboloidal mirror and takes pictures with a field-of-view of $360°$. Our camera system integrated with the side-view mirror of a car (b).

That paper focuses on the mechanical design and the control of the door. However, an important component of such a smart door is the image capturing and processing subsystem, whose output serves as an input to the control subsystem. In this paper we focus on the camera subsystem, on robust foreground detection algorithms and on algorithmic modifications for object extraction from image sequences captured by the camera. The extracted objects from images serve as an input for a *collision prediction* system that estimates the risk of collisions when opening the car door. This paper mainly focuses on robust background estimation and foreground detection, but we also briefly describe the function of the collision prediction system that generates a signal to lock or to stop door operations. The cameras in question are omnidirectional vision sensors consisting of a perspective camera focused on a cone-like hyperboloidal mirror. Fig. 2(a) illustrates such an omnidirectional camera that is integrated with the side-view mirror of a car to monitor the external environment (see Fig. 2(b)). Given the large field-of-view of the vision sensors, the camera is able to monitor the side of the car door in its entirety (see Fig. 3) and the associated image processing algorithms enable early-detection of impending obstacles.

We focus on detecting approaching obstacles like cyclists as they are often ignored by passengers compared to stationary obstacles next to the door. It is important to detect such traffic participants before any car door operation is performed, and they must be identified even if they are relatively far away from the car. However, stationary obstacles close to the door can be detected while the door is opening with a single camera using motion-stereo algorithms [Okutomi and Kanade 1993] and their applications in automotive research [Suhr et al. 2010]. Additionally, there is a small time interval between parking and door operations. This allows us to formulate certain preconditions under which our object detection and extraction algorithms may and should operate: (i) We may

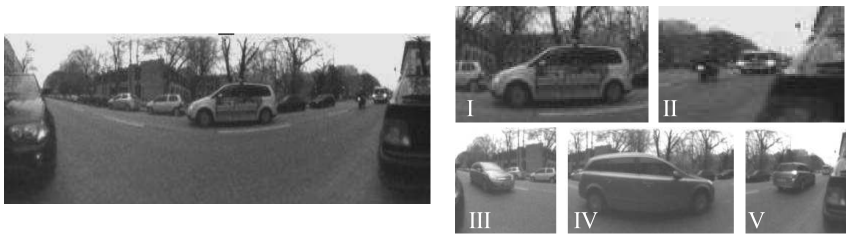


Fig. 3: Left: Panoramic image of the environment around a car door. Right: Different views and sizes of approaching and passing traffic participants due to the large field of view of the camera. The detection algorithm must be able to handle different sized objects (large (I) and very small objects (II)) and to robustly detect passing objects that differently appear in panoramic images (Different views of a passing car (front view (III), side view (IV) and rear view (V)).

| | Initia-lisation | Prior Knowl. | Static Ob-jects | Obj. Size Changes | Perspective Changes | Paral-leliza-bility | Execution Time |
|---|---|---|---|---|---|---|---|
| Template Matching | no | needed | ind | bad | bad | middle | slow |
| HMM | no | needed | ind | bad | bad | middle | slow |
| Feature Based | no | needed | ind | bad | bad | good | fast |
| Optical Flow | no | no | bad | ind | ind | good | middle |
| Background Estimation | needed | no | ind | ind | ind | good | fast |

Table I: Comparison of different object detection methods for environment surveillance and the performance of these methods for specific object properties such as static objects. Optical flow based methods perform bad when detecting static objects whereas the detection of static objects is independent (ind) for background estimation. *Background estimation* seems to yield the best results in our setting.

assume a static camera for a short time interval between after parking processes and first door operations. This interval may be used for learning the environment around the door. (ii) Objects that are further away from the camera occupy less pixels on each video frame (i.e., they occupy fewer pixels) and are hence not easy to differentiate from the background. (iii) Due to the large field-of-view of the cameras used, there is a different view of the same object (front, side and back) as it moves (see Fig. 3). (iv) Algorithms for detecting approaching traffic and obstacles must operate in real-time. Cars driving at $13.89m/s$ (about $50km/h$) move approximately $0.5m$ between two frames tracked with cameras having frame rates of 30 frames per second (fps). Hence, fast-moving cars or motorbikes have to be detected within a maximum of two or three frames.

Our studies illustrated a need of cameras with frame rates of at least 30fps to detect approaching objects and provide safe door operations. Consequently, we have to guarantee that our algorithms extract objects and predict the risk of collisions within at most 33ms ($\approx$30fps). The most time-critical part of our system is the object extraction algorithm. Therefore, an efficient realization of these algorithms – e.g., on a multi-core CPU or on an FPGA-based embedded system – is highly desirable. Later in this paper, we present a conceptual mapping of the time-critical object extraction algorithm on an FPGA. In doing so, real-time object extraction is feasible for images captured from high resolution, high frame rate cameras.

A number of techniques for object detection and extraction exist in the image processing literature, viz., Hidden Markov Models [Rabiner 1989], Template Matching [Brunelli 2009], feature-based detection methods, optical flow-based methods [Achler and Trivedi 2002],[Ghandi and Trivedi 2004] and background separating methods. Table I gives a brief overview of possible methods and compares them in terms of the required computation time, parallelizability for fast implementation on a multi core CPU or on an FPGA, and the ability to handle perspective changes. Background estimation and optical flow are not sensitive to perspective changes and object size, and they do not need prior knowledge of object properties, e.g., color, shape and geometry.

One disadvantage of optical flow is that it can not detect static or very slow moving objects. Although background estimation solves this problem, it needs a small time interval to learn the background. Fortunately, in our settings, such an interval is available, viz., the time interval between parking and door operation. Fig. 4 gives a high-level overview of our object extraction and risk prediction algorithm. As mentioned above, this paper focuses on real-time moving extraction algorithms using robust background estimation techniques, but we also briefly describe the function of the collision prediction system that generates a signal to lock or to stop door operations. In particular, we present extensions to background estimation (e.g., illumination compensation and shadow elimination for gray-scale images) that are specifically tuned to our setting of a smart car door equipped with omnidirectional cameras. The details of our algorithm for robust background estimation and foreground detection are described in what follows.

## 1.1. Related Work and Our Contributions

The problem of extracting objects from a video sequence has been widely studied in surveillance [Haritaoglu et al. 1998], traffic monitoring [Friedman and Russel 1997] and vehicle guidance. In most applications, separating the foreground from the background is the first step for object tracking. Background subtraction and foreground modeling are powerful methods whose advantages are feature-independent segmentation (e.g., textures, direction of move, speed). Some common techniques for background subtraction include Kalman filtering [Karman and Brandt 1990], kernel density estimation [Elgammal et al. 2002], hidden Markov models [Stenger et al. 2001], mixture of Gaussians [Zivkovic and Heijden 2006], and the use of color-based intensity-independent features [Ardoe and Berthilsson 2006]. Most of these algorithms represent each background pixel using a probability density function (PDF) and classify the pixels from new images as background depending on the description of the pixels by their density functions.

As an alternative, Bhaskar *et al.* [Bhaskar et al. 2007] developed a foreground detection algorithm using cluster density estimation based on a Gaussian mixture model. This algorithm is suitable for handling illumination changes as well as dynamic backgrounds. Similar work was done in [Zhong and Sclaroff 2003] using Kalman filtering to iteratively estimate the dynamic background texture and the regions of foreground objects. Kalman filtering was also used by Karman *et al.* [Karman and Brandt 1990] to model the background dynamics of each pixel by choosing two different gains, thereby allowing fast adaptation of background changes and slow adaptation of foreground pixels. Ridder *et al.* [Ridder et al. 1995] improved this approach and presented a shadow detection method assuming small differences between overshadowed and non-overshadowed background. However, strong shadows caused by direct sunlight cannot be detected.

Although many background subtraction techniques have been proposed, the majority of the algorithms address shadow detection and illumination compensation by exploiting color information (see [Elgammal et al. 2002; Zhong and Sclaroff 2003]). In scenarios where monochromatic video cameras are used – such as ours – the existing methods are no longer suitable. Our camera system consists of a monochromatic VGA camera that is designed for (cost sensitive) applications in the automotive domain. Expensive, high resolution color video cameras may be useful for research, but are impractical for real applications. For applications, where only monochromatic cameras are available, a common method to increase the robustness of image processing algorithms is by transforming intensity-based images into lighting invariant frames. This transformation, e.g., based on Census filtering [Zabih and Woodfill 1994; Dinkar and Nayar 1996] is widely used, but intensity

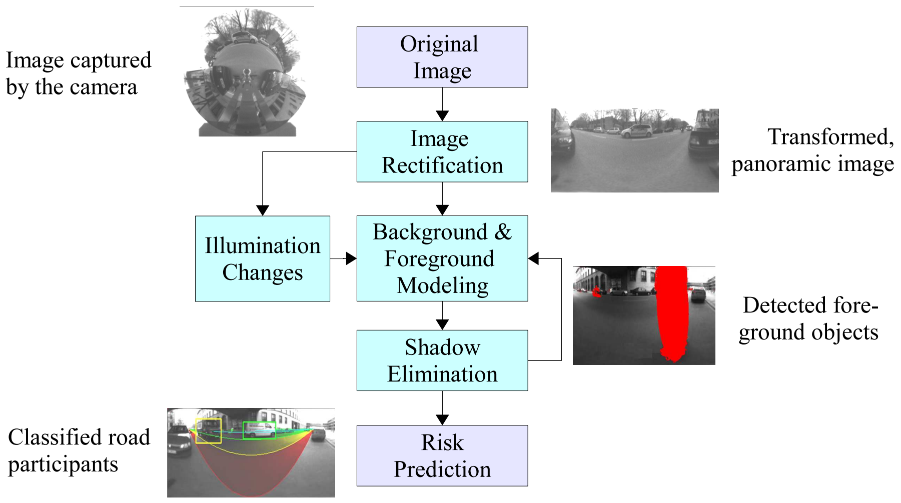


Image captured by the camera
Original Image
Image Rectification
Transformed, panoramic image
Illumination Changes
Background & Foreground Modeling
Detected fore-ground objects
Shadow Elimination
Classified road participants
Risk Prediction

Fig. 4: Block diagram for object detection using background estimation, shadow compensation and handling of illumination changes.

information of homogeneous regions are lost. In other words, it may be no longer possible to distinguish between homogeneous foreground (cars, trucks) and background regions (walls). Therefore, the use of intensity-based images is highly desirable for our application. But intensity-based monochromatic images lead to several challenges in object detection. For example, it is difficult to differentiate between small illumination changes caused by shadows or by small, valid foreground objects in gray scale images.

Another challenge is in detecting small objects in low resolution images captured by an omnidirectional vision system. Furthermore, for safe door operations, it is important to predict the risk of possible collisions in advance: This prediction is based on the object's positions in image data and on recognizing if there are objects with dangerous trajectories. Shadows could cause an inaccuracy in position determination and may lead to a wrong prediction of possible collisions. Hence, the shadow pixels both for moving objects like cars and static objects like pedestrians next to the door must be detected and suppressed. These problems, along with the accuracy of background subtraction, the handling of sudden illumination changes and the possibility of parallelizing the algorithms are the underlying motivations of our work.

Inspired by the background estimator of Ridder *et al.* [Ridder et al. 1995] and by the shadow detector proposed by Jacques *et al.* [Jacques et al. 2005], we develop robust background estimation and foreground detection algorithms for gray scale images. Ridder *et al.* proposed an extension to the Kalman-based background algorithm of Karman *et al.* [Karman and Brandt 1990] that takes weak shadows from stationary or moving objects into account. They assume that weak shadows have the same characteristics as illumination changes that may be adapted into the background. Therefore, their algorithm automatically increases the threshold for foreground detection using the variance of the estimated background values over time. The threshold is high if the variance of the estimated background values (e.g., caused by shadows) is high. However, the pixels from small foreground objects – such as motorbikes – also cause a high variance when detected as background and might be suppressed. Strong shadows cannot be identified in [Ridder et al. 1995], as they are detected as foreground. Once detected as foreground it is impossible to differentiate between shadow and foreground.

A good shadow detector for gray scale images was introduced by Jacques *et al.* [Jacques et al. 2005] using normalized cross correlation (NCC). The detector assumes shadow pixels as scaled versions (darker) of the corresponding background pixels, so that the NCC in a neighboring region is close to unity. On the other hand, this shadow detector misclassifies valid foreground pixels with small differences as shadow pixels. To overcome these limitations, we combine and modify these different methods to design a powerful background subtractor. We also extend the shadow detector with zero means cross correlation (ZNCC) in order to better distinguish between shadows and valid foreground pixels. The proposed shadow detector is suitable for detecting and eliminating all types of shadows – e.g., shadows from moving objects like cars or leaves moving in the wind and shadows from static objects. Our proposed algorithm detects illumination changes using local search windows and updates the background to compensate for slow or sudden illumination changes. Our experiments in complex outdoor and indoor environments under various lightning conditions demonstrate promising results. Additionally, we tested the background initialization algorithm in a heavy snowy scenario. We also evaluated and compared our approach with the approaches of Jacques *et al.* [Jacques et al. 2005] and Ridder *et al.* [Ridder et al. 1995]. We further evaluated our algorithms for their parallelizability potential and compared sequential and parallel implementations (on an AMD Quad-Core CPU). Our results indicate that they work in real-time on a multi-core CPU and are therefore suitable for potential implementation on an embedded platforms. In Section 5, we outline a conceptual mapping of the background estimation and foreground detection algorithm on an FPGA-based embedded system.

The rest of the paper is organized as follows. We describe the image rectification techniques in Section 2, the background initialization and the background estimator, the shadow detector, as well as our handling of illumination changes in Section 3. We briefly introduce our collision risk prediction algorithm in Section 3.7, and discuss our results obtained in Section 4. In Section 5, we present the conceptual mapping of the object detection algorithm on an FPGA. Finally, we conclude by briefly outlining some possibilities for future work.

## 2. CALIBRATION AND IMAGE RECTIFICATION

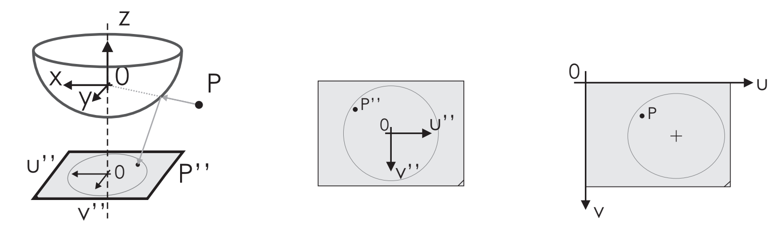
In this section we provide the technical details related to the omnidirectional camera subsystem. Original images from omnidirectional vision sensors are distorted and are not easy to handle for conventional image processing algorithms. The main problem is in extracting geometric and perspective relations like size and position of objects. To overcome this limitation, original images are transformed into panoramic (rectified) images. But in this case the camera model and the calibration parameters must be precisely known. We designed an omnidirectional vision system based on the well-known single point of view (SPOV) theorem of Baker and Nayar [Baker and Nayar 1999]. SPOV is also known as the projection center of the mirror onto which the perspective camera should focus. This is a prerequisite for geometrically correct panoramic image transformation. Our omnidirectional camera uses a mirror whose surface follows a hyperboloidal equation. Using such mirrors, the SPOV constraint is only valid for an accurate alignment of the mirror and the camera. However, this is difficult to realize and hence the camera system must be calibrated to compensate for misalignments as well as to obtain a precise relation between the 3D world point coordinates and the camera sensor coordinates.

### 2.1. Calibration

To determine the 3D positions of object points that are projected on the sensor plane, a calibration function $f(\vec{p})$ has to be found that describes a relation between a vector $\vec{p}$ to a 3D point $\mathbf{P}$ in world coordinates $x_P$, $y_P$ and $z_p$ and the camera coordinates $u_P$ and $v_P$.

$$\vec{P} = \begin{bmatrix} u_P \\ v_P \end{bmatrix} = f(\vec{p}) \text{ with } \vec{p} = \lambda \cdot \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}, \lambda > 0 \tag{1}$$

Different techniques are known for determining the function $f(\vec{p})$ [Baker and Nayar 2001; Scaramuzza et al. 2006a]. We use the calibration method developed by Scaramuzza *et al.* [Scaramuzza et al. 2006b]. All points lying on a light ray (vector $\vec{p}$) in world coordinates (see Fig. 5(a)) are mapped to the projection point $P''$ on the virtual plane $E''$ (see Fig. 5(b)). Point $P''$ on the vir-



(a) World to virtual sensor plane $E''$ (b) Point $P''$ on the virtual sensor plane (c) Real sensor plane

Fig. 5: The camera model (a) used in this paper. The world point P is mapped on a virtual sensor plane $E''$ (b) and the projection transformed to the real sensor plane (c) using affine transformations.

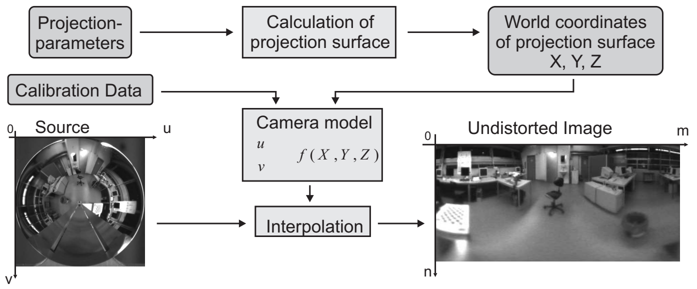


Fig. 6: Proposed image rectification process.

tual sensor plane $E''$ is then transformed to the real sensor plane using a transformation function that takes misalignments of the imaging device to the camera sensor into account (see Fig. 5(c)). Scaramuzza *et al.* propose the relation between world and virtual sensor plane as follows:

$$\vec{p} = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \lambda \cdot \begin{bmatrix} u_{P''} \\ v_{P''} \\ f(\rho) \end{bmatrix} = \begin{bmatrix} x_p \\ y_p \\ a_0 + a_2\rho^2 + \ldots + a_N\rho^N \end{bmatrix} \tag{2}$$

with $\lambda = 1$ and $\rho = \sqrt{x_{\vec{p}}^2 + y_{\vec{p}}^2}$. Furthermore, they approximate the component $z$ of $f(\vec{p})$ depending on the curvature as a polynomial function. The relation between the real sensor plane and the virtual or ideal sensor plane is given as an affine transformation (see Eq. 3).

$$\vec{P''} = \mathbf{A} \cdot \vec{P} + \vec{t} \quad \text{with} \quad \mathbf{A} = \begin{bmatrix} c & d \\ d & 1 \end{bmatrix} \quad \vec{P} = \begin{bmatrix} u_P \\ v_P \end{bmatrix} \quad \text{and} \quad \vec{t} = \begin{bmatrix} u_{center} \\ v_{center} \end{bmatrix} \tag{3}$$

The parameters $a_0 \ldots a_n$, $\mathbf{A}$ and $\vec{t}$ are the calibration parameters.

## 2.2. Image Rectification

Using the camera model and the calibration data, the projection of any 3d point onto the sensor plane can be calculated and vice versa. This allows us to determine a projection area based on individual projection parameters like width $M$, height $N$ as well as a region of interest (ROI) for image rectification as a first step. Secondly, each pixel $[m, n]^T$ of the projection area is stored in a $M \times N \times 3$ dimensional matrix $\mathbf{F}$ containing its world coordinates $\mathbf{X}(m, n)$, $\mathbf{Y}(m, n)$ and $\mathbf{Z}(m, n)$. Lastly, the corresponding pixel position of each point on the projection area is calculated and stored in a look up table (LUT) [Scharfenberger et al. 2009]. Using the information in the LUT and bicubic interpolation, every original image can be transformed into a panoramic image. Fig. 6 illustrates this flow.

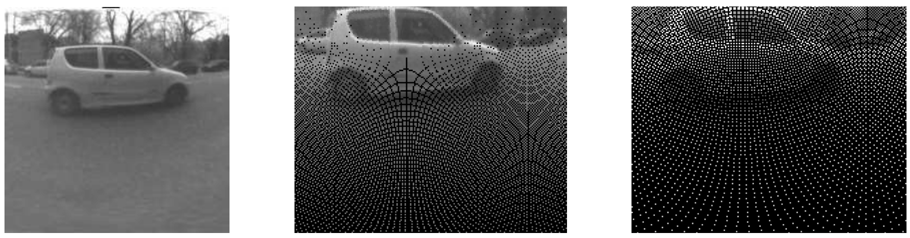


Fig. 7: Left: Rectified panoramic image. Middle: Pixel positions of rectified gray-scale images without interpolation. Right: Pixel positions of a rectified RGB-image captured by a simulated color camera (Bayer pattern). The use of color cameras with Bayer pattern reduces the resolution of rectified panoramic images.

The resolution of regions in rectified images highly depends on their image positions. Due to the physical properties of the camera there are more pixels available at the outer parts of the mirror than at the inner part for image rectification. Consequently, regions in original images close to the image center have lower resolutions than regions close to the outer bounds. This may lead to a lack of information in lower parts of rectified images as shown in figure 7(b). The resolution is further decreased for images captured by color cameras as they use Bayer pattern [Bayer 1976] to obtain color information (see Fig. 7(c)). This effect can be reduced using high-resolution (color) cameras, but on the other hand the execution time for image processing will drastically increase.

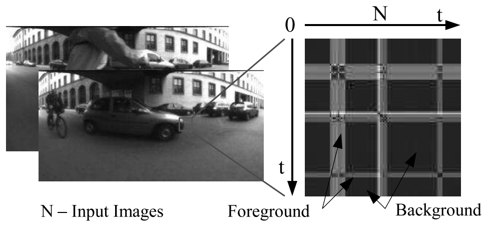
We use a VGA-resolution camera. However, the usage of imaging devices such as mirrors to enhance the horizontal field of view leads to a loss of resolution in panoramic images. This is because the mirror reflects light onto a limited space on the VGA-sensor (here $480 \times 480$ pixels) (see Fig. 6, bottom left). Removing the pixels on which the camera sees itself leads to a resolution of $480 \times 204$ pixels for panoramic images.

## 3. BACKGROUND MODEL

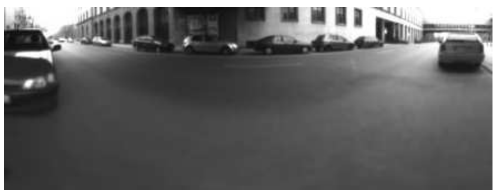
As discussed in Section 1, we used background estimation for extracting objects of interest from the captured images. Our background model is based on the approach of Karman *et al.* [Karman and Brandt 1990] and Ridder *et al.* [Ridder et al. 1995]. It is extended to provide better shadow detection and to be more robust against illumination changes for our application. In this section, we present the mathematical details of the background model along with the shadow detector and a method to account for illumination changes (see Figure 4). We describe the background initialization in Section 3.1 and the background model based on Kalman filtering proposed by Karman *et al.* [Karman and Brandt 1990] to model the dynamics of each background pixel in Section 3.2. We classify pixels as background or possible foreground pixels using thresholding. Possible foreground pixels are then classified as valid foreground or shadow pixels using the NCC and the ZNCC (see Section 3.3 and 3.4). Finally, we present a method to account for global illumination changes in Section 3.5.

### 3.1. Background Initialization

Due to typical parking situations with moving cars, bicyclists, pedestrians, etc. it is not possible to record a separate background image without any objects as required in many approaches. Every background pixel in the approach of Karman *et al.* [Karman and Brandt 1990] is initialized with a fixed value that is adapted during the training period using a large number of frames. Jacques *et al.* [Jacques et al. 2005] use median-based background initialization over a large number of frames. Median-based initialization allows to record background from busy-street scenarios, but it assumes that pixels contain background content for at least half of the initialization frames. Real life experiments demonstrated that this assumption may be violated for several traffic scenarios.



(a) Similarity matrix for pixel position $p_i$.



(b) Initialized background.

Fig. 8: This figure illustrates the Similarity Matrix for one pixel position $p_i$ (a). Low differences between the blocks for each pair of frame are shown as dark, high differences between the blocks as bright areas. b: Estimated background image as initialization for the background model.

Farin *et al.*, [Farin et al. 2003] proposed a powerful method to solve this problem. The principal idea of their algorithm is to roughly segment each pixel from input frames into foreground and background scenes. The segmentation is carried out on small blocks for each pixel position from the input frames. Background content is classified by searching for the subsets of frames that show stable content in the blocks. In other words, the content of blocks with background varies less than content of blocks with foreground. To identify the blocks containing background pixels, the similarity of block contents over a fixed trainings period is stored into a *Similarity Matrix*. This matrix contains the differences (realized with Sum of Absolute Differences, SAD) between image content at the block position for each pair of frames. Low values in the matrix relate to background regions whereas high values correspond to foreground regions due to differences between these blocks. The matrix for each position is decomposed into two parts, one that may contain background (low values) and one that may contain foreground (high values). The background image is then calculated based on pixels with background content using median algorithm [Massey and Bender 1996]. Such segmented background image represents the initial system state $\hat{I}_{x,y}(t_0)$ of the Kalman background model that is presented in Section 3.2.

Although the method proposed by [Farin et al. 2003] is very robust in background estimation, it is very time consuming due to the block difference calculation using SAD. Following the definition of SAD

$$SAD = \sum_{x=1}^{m}\sum_{y=1}^{m}|I_1(x,y) - I_2(x,y)| \tag{4}$$

the absolute, pixel wise differences of the intensities $I_1(x,y)$ and $I_2(x,y)$ for pixels $(x,y)$ in a frame 1 and 2 is computed for a block $B_i$ with size $m \times m$. These absolute differences have to be computed for every image pair in a set of $n$ frames. This leads to high computation times for background initialization depending on the number of frames $n$. Due to this, we replaced the SAD for block difference calculation by block averaging and by computing the absolute difference based on the block averages for each corresponding block in consecutive frames. In other words, we compute an average $\mu_n(p_i)$ for each block and use $\mu_n(p_i)$ for further computations. Previously computed mean-values can be used so that only one block difference computation is necessary. On the other hand, nine difference computations for blocks with size of 3x3 or more pixels would be required using SAD. In Eq. 5, we propose an efficient algorithm for block difference calculation based on block averaging.

$$\begin{aligned}
t_1 &: \mu_1(p_i) = mean(B_i(p_i)) \\
t_2 &: \mu_2(p_i) = mean(B_i(p_i))\ ,\ d_{1,2}(p_i) = |\mu_1(p_i) - \mu_2(p_i)| \\
t_n &: \mu_n(p_i) = mean(B_i(p_i))\ ,\ d_{j,n}(p_i) = |\mu_j(p_i) - \mu_n(p_i)|\ \ j \in [1, n-1]
\end{aligned} \tag{5}$$

For each pixel position $p_i$ in a frame of $n$ training frames, the (block)-average $\mu_n(p_i)$ of a block $B_i$ in the neighborhood of a pixel $p_i$ is determined as a first step. Thereafter, the absolute differences of blocks for each pair of frames are computed using the previously computed block-averages $\mu_n(p_i)$. Clearly, the block average $\mu_1(p_i)$ for each pixel in the first frame is computed at time step $t_1$. At time step $t_2$, the block averages $\mu_2(p_i)$ for all pixels in the second frame are computed and the absolute difference $d_{1,2}(p_i)$ is calculated using the previously determined block averages $\mu_1(p_i)$. The results are then stored in the matrix elements $d_{j,n}(p_i) = d_{n,j}(p_i)$ of the similarity matrix. At time step $t_3$, the block averages $\mu_3(p_i)$ are computed and the absolute differences $d_{1,3}(p_i)$ and $d_{2,3}(p_i)$ are determined. As in time step $t_2$, the previously determined block averages $\mu_1(p_i)$ and $\mu_2(p_i)$ from time steps $t_1$ and $t_2$ can be used for further processing (see Alg. 1).

However, block averaging leads to less robust initialization results (see Section 4) but it allows previously determined block average values $\mu_{(n-1)}(p_i) \cdots \mu_1(p_i)$ to be reutilized for background initialization. The result is a ten times speed up of the processing time. The computed background image serves as an initialization for the Kalman-based background estimation and foreground detection presented in the next section. The number of frames required for background initialization strongly depends on the number of foreground objects in the training sequence. Ten frames are sufficient to learn the background for empty scenarios and at least 40 frames for parking scenarios with high traffic volume.

### 3.2. Kalman Background Estimation

In this section, we describe the background model on which our approach is based. The background model has been proposed by Karman *et al.* [Karman and Brandt 1990] and was extended by Ridder *et al.* in [Ridder et al. 1995]. The intensity of a pixel at position (x,y) at time $t$ is given by $I_{x,y}(t)$. The estimated system state of the background model is denoted as $\hat{I}_{x,y}(t)$ and its derivative as $\hat{\dot{I}}_{x,y}(t)$. The estimation on the background is

$$\begin{bmatrix} \hat{I}_{x,y}(t) \\ \hat{\dot{I}}_{x,y}(t) \end{bmatrix} = \begin{bmatrix} \tilde{I}_{x,y}(t) \\ \tilde{\dot{I}}_{x,y}(t) \end{bmatrix} + K_{x,y}(t) \cdot \left( I_{x,y}(t) - H \cdot \begin{bmatrix} \tilde{I}_{x,y}(t) \\ \tilde{\dot{I}}_{x,y}(t) \end{bmatrix} \right) \tag{6}$$

Following Eq. 7, the prediction $\tilde{I}_{x,y}(t)$ of the system state $\hat{I}_{x,y}(t)$ and its derivative $\tilde{\dot{I}}_{x,y}(t)$ at time $t$ is given by:

$$\begin{bmatrix} \tilde{I}_{x,y}(t) \\ \tilde{\dot{I}}_{x,y}(t) \end{bmatrix} = S \cdot \begin{bmatrix} \hat{I}_{x,y}(t-1) \\ \hat{\dot{I}}_{x,y}(t-1) \end{bmatrix} \tag{7}$$

The system matrix $S$, the measurement matrix $H$ and the Kalman gain $K$ are:

$$S = \begin{bmatrix} 1 & s_{1,2} \\ 0 & s_{2,2} \end{bmatrix}, H = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \text{and} \quad K_{x,y}(t) = \begin{bmatrix} k1_{x,y}(t) \\ k2_{x,y}(t) \end{bmatrix} \tag{8}$$

In [Karman and Brandt 1990], $s_{1,2} = s_{2,2} = 0.7$ was used for modeling the background dynamics. Because the camera returns only the intensities $I_{x,y}(t)$, the measurement matrix $H$ is a constant. The Kalman gain was chosen depending on detected foreground or background using a pre-estimation of the next system state (see Eq. 9 and Eq. 10).

$$m_{x,y}(t) = \begin{cases} 1, & if \quad \left[d'_{x,y}(t) \geq th_{bg}\right] \vee \\ & \quad\;\; \left[(d'_{x,y}(t) < th_{bg}) \wedge \right. \\ & \quad\;\; \left. (d''_{x,y}(t) \geq th_{bg})\right] \\ 0, & if \quad \left[d'_{x,y}(t) < th_{bg}\right] \wedge \\ & \quad\;\; \left[d''_{x,y}(t) < th_{bg}\right] \end{cases} \tag{9}$$

$$\begin{aligned} d'_{x,y}(t) &= |I_{x,y}(t) - \tilde{I}_{x,y}(t)| \\ d''_{x,y}(t) &= |I_{x,y}(t) - \hat{I}'_{x,y}(t)| \\ \text{with} \quad \hat{I}'_{x,y}(t) &= \tilde{I}_{x,y}(t) + \beta \cdot \left[I_{x,y}(t) - \tilde{I}_{x,y}(t)\right] \end{aligned} \tag{10}$$

Pixels whose differences of the intensity to the system state are smaller than a fixed threshold $(d' < th_{bg})$, do not necessarily indicate background. To identify such pixels, a pre-estimation $\hat{I}'_{x,y}(t)$

**ALGORITHM 1:** Background initialization.

```
1: for all images $I_j,\ j \in \{0,\dots,n\}$ do
2:     for all pixels $p_i,\ i \in \{0,\dots,m\}$ do
3:         $\mu_{j,i}(p_i) =$ mean $(B(p_i))$                                   //compute the mean value $\mu_{j,i}(p_i)$ of each block $B(p_i)$
4:         $d_{x,j}(p_i) = |\mu_{x,i} - \mu_{j,i})|,\ x \in \{0,\dots,j-1\}$   //compute the entries of the similarity matrices $SM(p_i)$ (see Eq. 5)
5:     end for
6: end for
7: for all similarity matrices $SM(p_i),\ i \in \{0,\dots,m\}$ do
8:     $bg(p_i) =$ decomposeSM($SM(p_i)$)                                    //extract the background (see [Farin et al. 2003])
9: end for
```

of the next system state is calculated assuming that these pixels belong to background. If the pre-estimated value $d''$ is greater than $th_{bg}$, this pixel nevertheless belongs to foreground. A binary map $m_{x,y}(t)$ represents the segmentation of pixels (1 for foreground and 0 for background), and the Kalman gain $k1,2_{x,y}(t) = \alpha$ or $k1,2_{x,y}(t) = \beta$ is chosen depending on the binary map $m_{x,y}(t)$ (see Eq. 11).

$$k1,2_{x,y}(t) = \begin{cases} \alpha, & if \quad m_{x,y}(t) = 1 \\ \beta, & if \quad m_{x,y}(t) = 0 \end{cases} \tag{11}$$

### 3.3. Shadow Detection

As mentioned in Section 1, it is important to predict the risk of possible collisions in advance. This prediction is based on the object's positions in image data and on recognizing if there are objects with dangerous trajectories. Shadows cause an inaccuracy in position determination that lead to a wrong prediction of possible collisions: Therefore, shadows must be detected and suppressed. The proposed shadow detection algorithm is suitable both for detecting stationary and dynamic shadows. The characteristic of shadows from moving leaves is identical to the characteristic of shadows of moving cars and can hence be detected and suppressed. We use the normalized cross correlation (NCC, [Jacques et al. 2005]) as an initial step for shadow detection and refined it using zero means normalized cross correlation (ZNCC) to handle foreground pixels with small differences with the background.

Let $\tilde{I}_{x,y}(t)$ be the estimated background image and $I_{x,y}(t)$ an image given by the camera system. For each foreground pixel, we generate a template $T_{xy}(n,m)$ such that $T_{xy}(n,m) = I_{x+n,y+m}(t)$ for $-N \leq (n,m) < N$ where $\bar{t}$ is the mean of the template $T_{xy}(n,m)$. Furthermore, let $B_{xy}(n,m)$ be the template of the background such that $B_{xy}(n,m) = \hat{I}_{x+n,y+m}(t)$ where $\bar{b}$ is the mean of template $B_{xy}(n,m)$. The ZNCC as well as the NCC ($\bar{t} = 0$, $\bar{b} = 0$) between $T_{xy}(n,m)$ and $B_{xy}(n,m)$ at pixel $(x,y)$ can be calculated using Eq. 12:

$$ZNCC_{x,y} = \frac{EZR_{x,y}}{EZB_{x,y} \cdot EZT_{x,y}} \tag{12}$$

with

$$\begin{aligned} EZR_{x,y} &= \sum_{n=-N}^{N} \sum_{m=-N}^{N} |(B_{xy}(n,m) - \bar{b})||(T_{xy}(n,m) - \bar{t})| \\ EZB_{x,y} &= \sqrt{\sum_{n=-N}^{N} \sum_{m=-N}^{N} (B_{xy}(n,m) - \bar{b})^2} \text{ and} \\ EZT_{x,y} &= \sqrt{\sum_{n=-N}^{N} \sum_{m=-N}^{N} (T_{xy}(n,m) - \bar{t})^2} \end{aligned} \tag{13}$$

where $EZT_{x,y}$ considers the energy of the image template and $EZB_{x,y}$ considers the energy of the estimated background. A pixel may potentially be classified as shadow if its NCC in the neighborhood is close to unity and its energy $ET_{x,y}$ is lower than the energy of the background $EB_{x,y}$ (see Eq. 14).

$$NCC_{x,y} \geq th_{NCC} \text{ and } EB_{x,y} > ET_{x,y} \tag{14}$$

$EB_{x,y}$ as well as $ET_{x,y}$ can be determined by calculating $EZB_{x,y}(\bar{b}=0)$ and $EZT_{x,y}(\bar{t}=0)$.

### 3.4. Shadow Refinement

Depending on the chosen threshold $th_{NCC}$ with $(th_{NCC} < 1.0)$, many foreground pixels with small differences to background pixels may be misclassified as shadow pixels. To overcome this limitation, we refined the classification of shadow- and nonshadow-pixels using the ZNCC. The advantage of ZNCC is light invariance, so that only differences in texture cause significant changes in its value. The refinement stage verifies if there are significant changes through textures and not through illumination. Although the ZNCC is light invariant, image noise (texture changes) influences the ZNCC and causes an offset. This offset $\theta$ can be determined while learning the background model or can be considered by the threshold $th_{ZNCC}$. Similar to the NCC, a pixel is a shadow candidate if the ZNCC in the neighborhood is close to the learned initial value and the energy $ET_{x,y}$ of the template is lower than the energy of the background $EB_{x,y}$. But contrary to the NCC, $EZT_{x,y}$ and $EZB_{x,y}$ represent only the energy of the textures from the background and the template. Hence, the energy of texture from a valid foreground pixel can be lower than the energy of texture from background. This is the case for large homogenous objects like cars. A pixel must then be a shadow candidate if the energy $EZT_{x,y}$ is approximately equal to the energy $EZB_{x,y}$ (see Eq. 15).

$$\begin{aligned} |ZNCC_{x,y} - (1.0-\theta)| &\leq th_{ZNCC}\ and \\ |EZB_{x,y} - EZT_{x,y}| &\leq th_{comp}\ and \\ ET_{x,y} &< EB_{x,y} \end{aligned} \tag{15}$$

### 3.5. Active Light Adaptation

Background models based on Kalman filtering can follow slow illumination changes in the background. However, when foreground objects cover the background, illumination changes in the background cannot be detected. Furthermore, sudden illumination changes cannot be respected by the background because depending on the chosen threshold sudden illumination changes cannot be classified as valid foreground or illumination changes. So there is a need to modify the background, taking illumination changes into account. Therefore, we subdivide every new image into $m$ subimages at each position $p_{x(m)}$ and $p_{y(m)}$ fitting the whole image and calculate their mean gray values (see Eq. 16).

$$\mu(m,t) = \frac{1}{J \cdot I} \sum_{j=-J/2}^{J/2} \sum_{i=-I/2}^{I/2} I(p_{x(m)}+j, p_{y(m)}+i, t) \tag{16}$$

Here, $J$ and $I$ are the subimage sizes. The global illumination change $\Delta(t)$ can now be detected by calculating the median of all local illumination changes $\delta(m,t)$:

$$\Delta(t) = \underset{m}{median}\,\delta(m,t) \tag{17}$$

with

$$\delta(m,t) = \mu(m,t) - \mu(m,t-1) \tag{18}$$

Because small illumination changes are adapted by the background model, we decided to use simple thresholding in order to avoid modifying the background model too frequently.

$$\Delta(t) = \begin{cases} 0, & if\ \ \Delta(t) < th_{\Delta} \\ \Delta(t), & if\ \ \Delta(t) \geq th_{\Delta} \end{cases} \tag{19}$$

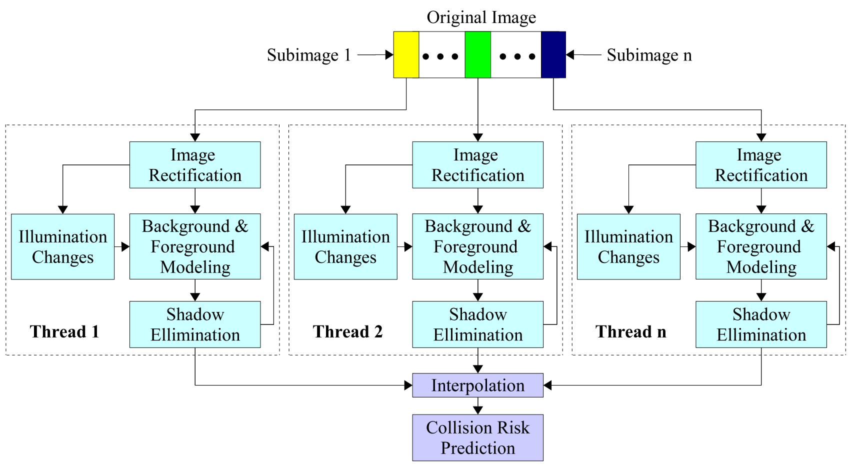


Fig. 9: Parallelization concept of the object detecting algorithm. Each input frame is subdivided into n-frames and the same number of concurrent threads is generated to extract road users. The results are then merged by the thread interpolation.

Finally, Eq. 7 for system state prediction is modified to overcome illumination changes and to update the background.

$$\begin{bmatrix} \tilde{I}_{x,y}(t) \\ \tilde{\hat{I}}_{x,y}(t) \end{bmatrix} = S \cdot \begin{bmatrix} \hat{I}_{x,y}(t-1) \\ \hat{\hat{I}}_{x,y}(t-1) \end{bmatrix} + \begin{bmatrix} \Delta(t) \\ 0 \end{bmatrix} \tag{20}$$

Using this approach, the background model can account for slow as well as for sudden illumination changes.

## 3.6. Parallelization

For all our proposed techniques, it may be seen that different pixels may be processed in parallel. In other words, image rectification, the background estimator, the shadow detector as well as the illumination compensation can all be run in parallel on a multi-core CPU. As mentioned before, our algorithms have to work in real-time and hence such parallelization is highly desirable. This is useful for speeding up the system for time critical tasks like detection of traffic participants. The original image returned by the camera subsystem is divided into $n$ subimages and the same number of threads is generated to run on a multi-core platform. After processing each image the results from all threads must be merged and interpolated (e.g., when an object being detected is split across two or more subimages) for further object detection and classification algorithms. Fig. 9 illustrates our realized parallelization technique.

## 3.7. Collision Risk Prediction

In this section, we briefly describe our algorithm that predicts the collision risk of approaching objects with the car door. The algorithm generates information that serves as an input for the door collision avoidance planner. The prediction of potential collisions is based on actual object positions in image data and on recognizing if there are objects with dangerous trajectories.

The ambiance besides the door is subdivided into three danger-zones: in a red, in a yellow and in a green danger-zone. These danger zones define the potential collision risk of objects located in one of these zones and allow risk prediction of approaching objects in advance (see Fig. 10, left). In other words, the collision risk of an approaching road user depends on its presences in one of these three danger-zones. Objects moving in the green zone will not collide while collisions are highly probable for objects located in the red zone. The width of the red zone is equal to the maximum workspace

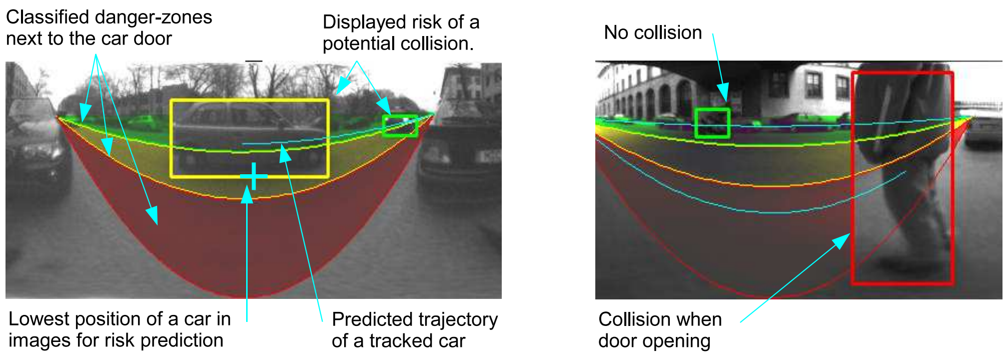

Classified danger-zones next to the car door
Displayed risk of a potential collision.
Lowest position of a car in images for risk prediction
Predicted trajectory of a tracked car
No collision
Collision when door opening

Fig. 10: Left:) Danger-zones next to the car and trajectories of objects used to predict the risk of collisions. Right:) Examples of detected objects and predicted risk of potential collisions.

of the car door. Objects within the yellow-zone do not directly cause a collision, but may move into the red-zone and need hence to be indicated as a potential risk. The size of the green zone and of the yellow zone can be specified during car manufacturing, and the corresponding areas in images can be determined by intrinsic and extrinsic camera calibration. For these reasons, the location of extracted objects – here the lowest point of an object (see Fig. 10, left) – their approximate distance from the car and their presences in one of these zones can be obtained from image data only. So, the proposed algorithm is suitable for distinguishing between objects that pass the door very closely and objects passing within a suitable large distance.

To increase the robustness of risk prediction, the algorithm also estimates the trajectory of detected objects by tracking the object's positions. A trajectory is computed using a quadratic function and the vertex of the parabola is determined. The location of the vertex in one of these zones additionally defines the collision risk for an approaching object. Trajectory estimation prevents undetected, potential collisions for objects in the green zone. Such objects may move into the red zone and may be classified as hazard-free if only their actual positions are used.

Finally, the algorithm generates three types of messages for the door collision avoidance planner (see Fig. 1): One message that indicates a collision and leads to a door lock, one that indicates a low weighted collision (for objects in the yellow zone) allowing only slow door operations and one for no collision. Fig. 10 illustrates examples of extracted objects and their predicted collisions. Since the proposed algorithm distinguishes between objects that pass the door very closely and objects that pass the door in a suitable large distance, car door users are able to open the door while parking in a busy street when there are no objects potentially colliding with the car door. However, if there are objects passing the car very closely, one may not be able to open a car door.

## 4. EXPERIMENTAL RESULTS

To verify and to evaluate our approach, we conducted experiments in complex environments containing weak and strong shadows as well as small differences between foreground and background scenes using an omnidirectional vision system (ODVS). Image rectification was used to transform the captured images into panoramic images of size $480 \times 204$ pixels. Images from the ODVS were used to test the algorithm under various conditions (dark and light regions, image noise and different resolutions due to image rectification and interpolation). We also conducted experiments to test the algorithm in terms of fast and slow illumination changes and present its performance in the presence of heavy snow. Similar to all image processing algorithms using off-the-shelf CMOS cameras, the performance drastically decreases for dark scenarios, and the algorithm does not work in absolute darkness.

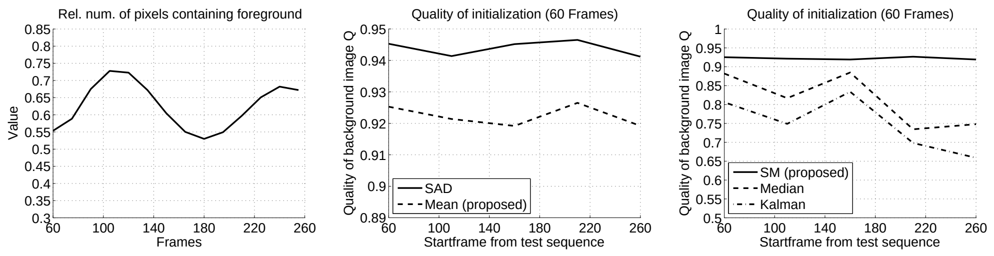


Fig. 11: Left: Number of image pixels containing foreground relative to the total number of image pixels. Middle: Quality differences using SAD and averaging. Right: Quality of the obtained background images for different numbers of training images from the initialization process started at different frames of the trainings sequence.

## 4.1. Background Initialization

In this section, we present our results on background initialization. We introduce the background quality $Q$ to compare and to evaluate different methods for background initialization (see Eq. 21). The background quality is defined as the ratio between the number of correctly extracted background pixels $N_{Background}$ and the number of ground truth pixel $N_{Groundtruth}$. A valid background pixel $pb_i$ is a pixel that have a maximum difference $d = |pb_i - pbg_i| \leq THRES \quad \forall i \in [1, n]$ to the corresponding ground truth background pixel $pbg_i$, where $THRES = 5$ for our application.

$$Q_{background} = \frac{N_{Background}}{N_{Groundtruth}} \tag{21}$$

Additionally, we generated trainings sequences from real life parking scenarios with many non stationary foreground objects to evaluate the applicability of the background initialization in busy-street scenarios. Fig. 11 (left) illustrates the percentages of foreground pixels in frames from a chosen initialization sequence. Due to the large field of view of the camera, such a high number of foreground objects in images is quite common for busy-street scenarios. The background image is estimated using a block similarity matrix (SM) (see Section 3.1), whose entries are determined by calculating the similarity between the pixels (in a block) for each pair of frames. To speed up the execution time, averaging instead of SAD for block differencing was used. However, the quality of our initialized background image is lower compared to the background image based on SAD (see Fig. 11), middle, but experiments demonstrated a fast adaptation of wrongly initialized background pixels.

Fig. 11, right, compares our method for background initialization to the median-based [Jacques et al. 2005] and to the kalman-based [Ridder et al. 1995] initialization. Our initialization process started at different frames of the trainings sequence for a different number $N$ of input images. The comparison demonstrates that at least 40 images are enough for SM-based background initialization compared to other methods where more training images are needed for better initialization results. Thereby, the challenge for the other methods is the large number of pixels containing foreground content for more than the half of input frames.

We also conducted experiments to determine the performance of background-initialization in the presence of heavy snow. Fig. 12 illustrates training images of a snowy scenario (with highlighted snowflakes) and the resulted initialized background images. Simulations demonstrated the same performance for rainy scenarios. So, the proposed background initialization is mostly robust against heavy snow and heavy rain.

## 4.2. Detection of Shadow Pixel Candidates

To detect small differences in intensity between foreground and background objects, the threshold $th_{bg}$ (see Eq. (9)) must be low. An experimentally obtained value was $th_{bg} \geq 5$ that allows the

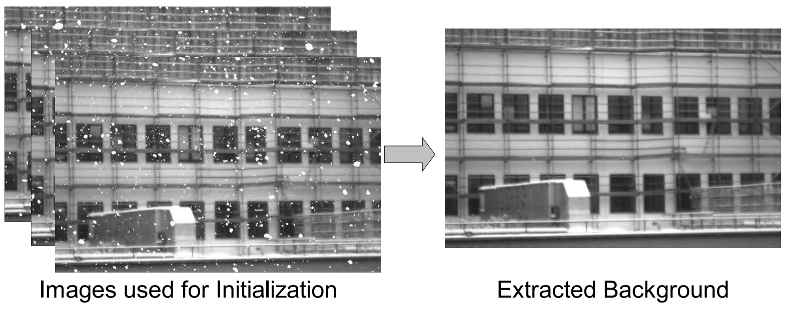


Fig. 12: Training images of a snowy scenario (highlighted snowflakes) and initialized background image.

detection of small intensity differences, but still many shadow pixels and noise are detected. Fig. 13 illustrates the result of foreground detection (BG/FG) for one pixel over time using NCC and ZNCC. NCC was useful to pre-estimate shadow pixels, but valid foreground pixels are often misclassified as shadow pixels that can be seen on the noisy characteristic for the foreground (*BG/FG NCC*). ZNCC overcomes these limitations by taking textural changes into account, so that foreground pixels are not misclassified as background. The result is a smoother characteristic of the foreground/ background characteristic (*BG/FG ZNCC*).

Fig. 14 compares our results of shadow detection to the shadow detection algorithm proposed by [Ridder et al. 1995]. Ridder *et al.* assume, that weak shadows have the same characteristic as illumination changes that may be adapted into the background. Therefore, their algorithm automatically increases the threshold for foreground detection using the variance (see Fig. 14, variance) of the estimated background values over time. The threshold is high if the variance of the estimated background values (e.g. caused by shadows) is high. However, pixels from small foreground objects – such as motorbikes – that are classified as background also cause a high variance and might be suppressed (see frames 160-175).

Strong shadows cannot be identified as they are detected as foreground. Once detected as foreground it is impossible to differentiate between shadow and foreground (see frames 115-130). Increasing the threshold up to $15$ may suppress strong shadows, but foreground objects with small differences to the background may be suppressed as well. Shadow detection based on NCC and shadow refinement based on ZNCC allows the use of a small threshold to extract foreground objects (see Fig. 14, ZNCC) and is suitable to eliminate strong shadow borders.

## 4.3. Illumination Changes and Background Adaptation

Fig. 15 illustrates our experiments with various illumination changes. The reference characteristic of the background is presented in Fig. 15(a) and various characteristics of the background disturbed by illumination changes Fig. 15(b). The background model accounts for slow illumination changes even if the background was covered for a short time interval and illumination changes were not to large (see Fig. 15(b), frames $0 - 100$). Sudden illumination changes, which are larger than $th_{bg}$, cause wrong foreground information (see frames (280 - 310) and (380 - 400)). Lastly, figure (15(c)) demonstrates that detected illumination changes can successfully be compensated if they are accounted for by the background model (see Eq. (20)).

We also conducted experiments to find the optimal number of search windows (NoW) for detecting global illumination changes. The number of search windows must be chosen so that influence of illumination changes caused by foreground objects is minimized (see Eq. (17)). We generated a test profile of illumination changes (IC) and tracked it by detecting illumination changes with different numbers of search windows. Experiments showed that at least 60 search windows were necessary to track the light profile sufficiently. The main problem of less then 60 search windows

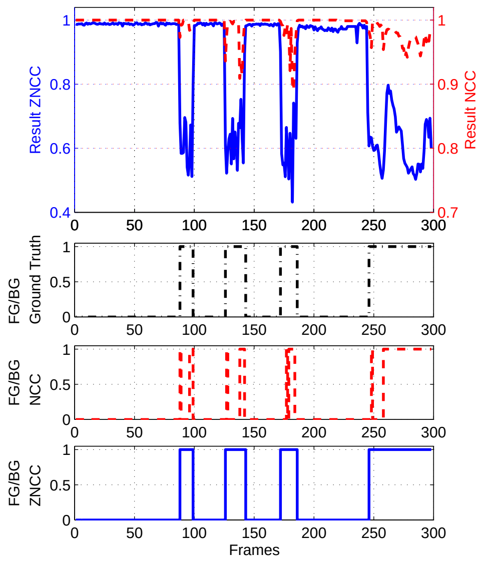


Fig. 13: Top: Characteristic of NCC and ZNCC from an image pixel. Middle and Bottom: Classification of foreground ($FG/BG = 1/0$) using NCC and refined by ZNCC. ZNCC can better distinguish valid foreground from shadow.

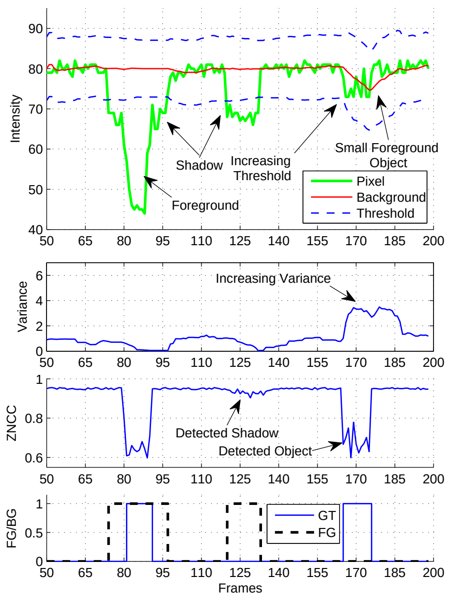
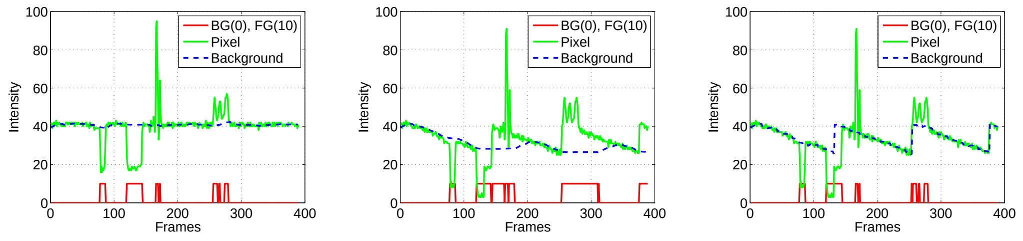


Fig. 14: Top: Characteristic of an image pixel and the variance of background pixels over time. Middle: Variance for threshold adaptation proposed by Ridder and our shadow refinement. Bottom: FG/BG classification and shadow detection using the approach of Ridder (dashed line) and our approach.



(a) No illumination changes. (b) Detection result by slow and fast illumination changes. (c) Compensation of illumination changes.

Fig. 15: (a) One pixel and the detected foreground over time (reference). (b) Misclassification of foreground pixels caused by fast and slow illuminations changes. (c) Adaptation of fast and slow illumination changes.

is the high influence of lighting changes caused by foreground objects. The influence of foreground objects is almost suppressed using $90$ NoW. Our measurements of tracking the test profile using different numbers of search windows illustrates Fig. 16. We also derived from our experiments that one search window should not be smaller than $(15 \times 15)$ pixels due to the increased influence of

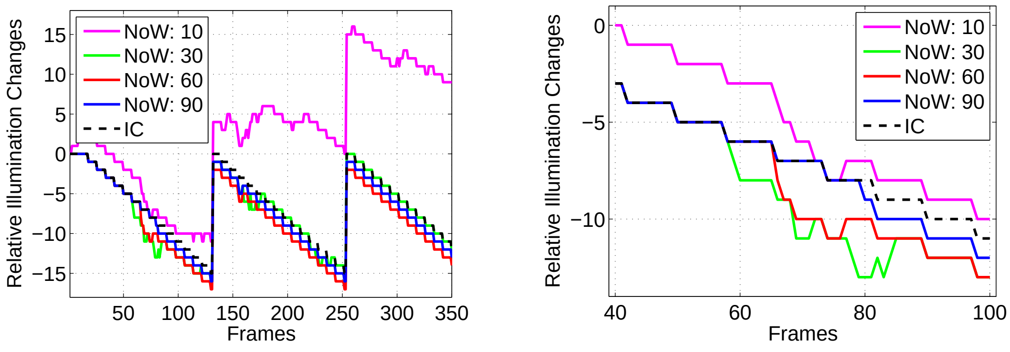


Fig. 16: Left: Illumination changes, Right: Frames 40 - 100. The more number of search windows (NoW) can be used for detecting illumination changes(IC), the better is the detection result. Good results give a NoW of 60 - 90.

image noise from smaller window size. An offset less than 2 (less than $th_\triangle$) in tracking the profile was tolerable and automatically adapted by the background model.

## 4.4. Validation of Foreground Pixels

Not all detected foreground pixels need to be valid (true positives = t.p.), i.e., there might also be false positives. For example, shadow pixels are often misclassified as valid foreground (false positives = f.p.). On the other hand, pixels having small differences to background can falsely be classified as background pixels (f.n., false negatives). Fig. 17 illustrates an example of a typical road scenario containing both true and false positives as well as false negatives. We also evaluated our algorithm in terms of false negative, true positive and false positive detection rates under various conditions like diffused light, direct sunlight and indoor conditions and compared the results obtained with a perfect detection. These results are shown in Table II, where the percentages were computed based on ≈200 test images. Shadow pixels in images with sunlit scenarios can easily be misclassified as valid foreground pixels. In general, small objects like motorbikes are also extracted in sunlit scenarios even when they are far away from the car: but only $76\%$ of their pixels are classified as valid foreground. Clearly, such regions may consist of only 20 pixels – and approximately five pixels of such objects not being classified as valid foreground resutls in a false negative rate of $25\%$. The false negative rates drastically decrease when such objects approach to the car. High false negative rates may lead to collision warnings due to bad collision prediction for objects far away from the car, but the prediction becomes more precise when such objects approach the car. Good detection rates were achieved for large foreground objects in all tested scenarios.

Fig. 18 illustrates the detection of a simulated object surrounded by snow flakes in a snowy scenario. The snowflakes are detected as small, rapidly moving objects and are removed using median filtering [Massey and Bender 1996]. Fig. 18, right, illustrates the result for objects in snowy scenarios with removed snow flakes: Large snowflakes close to the camera overlapping the boundary of objects cannot be removed using median-filtering and may lead to a wrong estimation of the object position in images. However, tracking the object over long image sequences or using cameras with high frame rates overcomes this limitation. Since snowflakes move very fast, the number of images in a huge data set containing inaccuracies in object detection caused by snowflakes or heavy rain is small.

## 4.5. Computation Time and Parallelization

We chose a complex indoor environment with three walking people, shadow effects and some illumination changes (switching light on/off) to measure the execution time of the proposed algorithm.

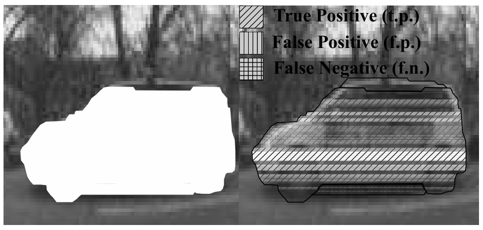


Fig. 17: Left: Detected pixels of a foreground object. Right: Examples of true positives and false negative pixels in an detected foreground object.

| Scenario | Obj. Size | t.p. | f.n. | f.p. |
|---|---|---|---|---|
| Diffuse light | small | 85% | 15% | 2% |
| | large | 95% | 5% | 3% |
| Sunlight | small | 76% | 24% | 7% |
| | large | 93% | 7% | 10% |
| Indoor cond. | small | 90% | 10% | 1% |
| | large | 97% | 3% | 4% |

Table II: Overview of our validation results: Percentages of true positive, false negative and false positive pixels in foreground regions.

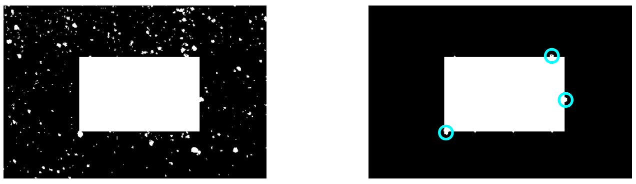


Fig. 18: Left: Simulated foreground object with extracted snowflakes. Right: Removed snowflakes using median-filtering and remaining disturbances at the detected object.

| time | Averaging | Similarity | Decomp. | Total Time |
|---|---|---|---|---|
| $t_0$ | $\approx$ 2.7 ms | - | - | 2.7 ms |
| $t_1$ | $\approx$ 2.7 ms | $\approx$ 0.2ms | - | 2.9 ms |
| $t_2$ | $\approx$ 2.7 ms | $\approx$ 0.4ms | - | 3.1 ms |
| $t_{39}$ | $\approx$ 2.7 ms | $\approx$ 8.2 ms | - | 10.9 ms |
| $t_{40}$ | $\approx$ 2.7 ms | $\approx$ 9.0 ms | - | 11.7 ms |
| $t_{41}$ | - | - | $\approx$ 412 ms | 412 ms |

Table III: This table illustrates the computation time on an AMD Phenom 9650 quad-core CPU at 2.54 GHz for background initialization.

We use about 400 test images of this data set and calculated the mean execution time as well as the standard deviation (Std. Dev.). Table III gives an overview of the execution times for our background initialization algorithm (see Section 3.1) using $N = 40$ input frames with size $480 \times 204$.
We also conduct experiments to determine the execution time for position determination, trajectory estimation and risk prediction. Therefore, we chose a scenario containing five small and large approaching objects: The execution times for all objects was $\approx 2ms$ using a single core. Table IV, left demonstrates the execution times for rectification, background modeling as well as shadow detection, illumination changes and collision detection using a single core of a 2.54 GHz AMD Phenom 9650 quad-core CPU. As discussed in Section 3.6, we parallelized our object detection algorithm using multithreading on the quad-core CPU and measured the execution times for the same test data set. The image was divided into $n$ subimages, each of that was processed by a different concurrent thread. The result of all threads is then merged using small thread called interpolation. Clearly, as the total computation time will be decreased with increasing number of threads, the time for merging and interpolation increased. Table IV, right gives an overview of the measured times. Table IV also illustrates the performance (throughput) in frames per second (fps) of the proposed algorithm. Fig. 19 illustrates the single computation time for each processing step as well as the overall compu-

| | Mean Time (1 core) | Std. Dev. | Image Size | 2 cores | 4 cores |
|---|---|---|---|---|---|
| Rectification | ≈ 3.6 ms | 0.7 ms | 640 × 480 | ≈ 1.9 ms | ≈ 1.0 ms |
| Background | ≈ 30.1 ms | 1.2 ms | 480 × 204 | ≈ 15.1 ms | ≈ 7.5 ms |
| Shadow Dect | ≈ 24.1 ms | 2.3 ms | ” | ≈ 12.2 ms | ≈ 6.3 ms |
| Ill. Comp | ≈ 10.4 ms | 1.6 ms | ” | ≈ 5.6 ms | ≈ 2.8 ms |
| Interpolation | 0.0 ms | 0.0 ms | ” | ≈ 2.0 ms | ≈ 2.2 ms |
| Collision Prediction | ≈ 2.0 ms | | ” | ≈ 2.0 ms | ≈ 2.0 ms |
| Total Time | 70.2 ms | | ” | 38.8 ms | 21.8 ms |
| Throughput | 14.25 fps | | ” | 25.77 fps | 45.87 fps |

Table IV: This table illustrates the computation time on a 2.54 GHz AMD Phenom 9650 quad-core CPU for the non parallelized algorithm (left), the parallelized algorithm (right) and the achieved throughput in frames per second [fps] using 1-4 cores.

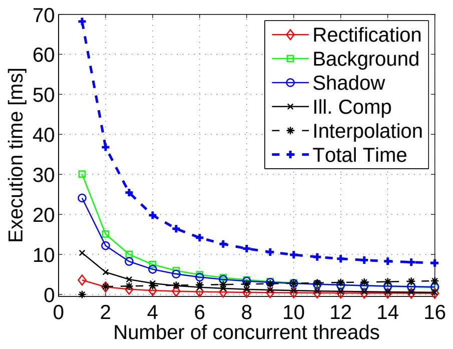


Fig. 19: Estimation of computation time using more than 4 cores (without collision prediction).

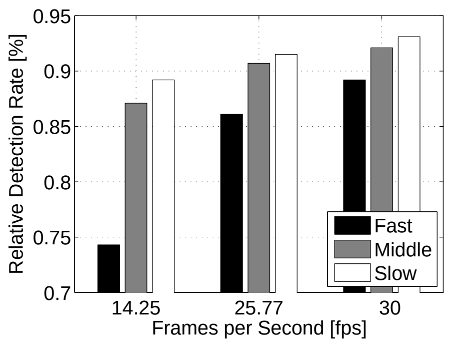
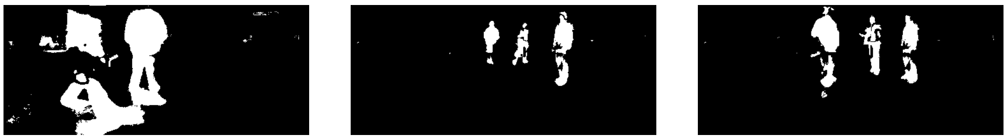


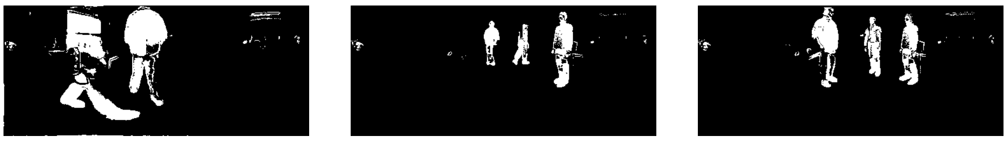
Fig. 20: Detection rate of fast, middle and slow objects depending on the frame rate.

tation time for more than 4 cores. This estimation is based on our measurements for 1, 2 and 4 cores on the quad-core CPU and is extrapolated up to 16 cores. Due to the increasing time for merging and interpolation the results of all threads there is no appreciable speed up using more than 16 cores.

In Fig. 20, we present our results on object detection and risk prediction using different sized objects and frame rates. We refer object detection and risk prediction as detection and compared the results with perfect detection. Detection rates for fast objects might not be sufficient for cameras with low frame rates due to poor risk prediction results. This can be explained due too large distances between one object in consecutive frames: Since their positions are used for trajectory estimation, the trajectories cannot sufficiently be estimated for fast moving objects, in particular when an object is not detected. Higher frame rates overcome this limitation due to smaller differences in object position for consecutive frames. On the contrary, slow objects result in good detection rates independent of the chosen frame rate. Finally, we evaluated our proposed method using various scenarios and compare our approach with other well known algorithm described in Section 1.1. Fig. 21 illustrates one of our evaluated scenarios containing up to 500 test frames. Difficulties of this scenario are a less textured environment as well as weak and strong shadows induced by different lighting sources. While the approach in [Ridder et al. 1995] modeled the background well, shadow detection failed in some cases. Similarly, while the shadow detector in [Jacques et al. 2005] performed well, the background model has some limitations. One limitation is that once the background is learned the background is not updated, which results in many noisy foreground pixels caused by illumination changes etc. The combination of both algorithms and the modification

Original image sequence.


Less noise in background modeling [Ridder et al. 1995], but detected strong shadows pixels.


There is a good shadow detection in [Jacques et al. 2005], but still image noise.

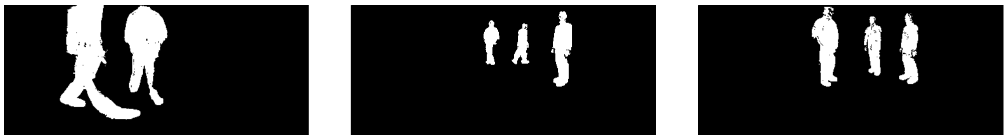

Our approach with proposed shadow and foreground detection. Noise as well as shadows can be better suppressed.

Fig. 21: Evaluation of our approach with different background models and shadow detection algorithms.

of the shadow detector as well as the light compensation led to a powerful background estimator that resulted in better foreground detection on grayscale images, when compared with state of the art techniques (see Fig. 21). Fig. 22 illustrates two parking scenarios and detected road users like bicycles, motorbikes, cars and people walking close to the car. The images in the first row illustrate detected objects in a normal street parking scenario with oncoming traffic, whereas the images in the second row illustrate road participants detected in a two-lane one-way street parking scenario. It can be shown, that fast moving, small objects like motorbikes are also detected, although they are relatively far away from the car (see Fig. 22, 2. row, middle). Risk prediction is performed with using the image positions of extracted objects in one of the danger-zones and their trajectories. The prediction result is displayed in the image.

## 5. POTENTIAL IMPLEMENTATION ON AN FPGA

Robust detection of fast moving objects like cars or cycles is a prerequisite for safe door operations. Cars driving at $13.89 m/s$ (about $50 km/h$) move approximately $0.5m$ between two frames tracked with cameras having frame rates of 30 fps and have to be detected within two or three frames. The most time-critical part of our system is the background estimation and foreground detection algorithm: Therefore, we introduced a parallelization scheme in Section 3.6 to guarantee real-time foreground detection within 33ms ($\hat{=}$ 30 fps) for panoramic images with a resolution of $480 \times 204$ pixels. For our implementation, we used a quad-core CPU and obtained a maximum throughput of $\approx$ 45 fps. However, the core-based implementation provides only a throughput of 84 fps (see Fig. 19) for images with resolutions of $480 \times 204$ pixels. Furthermore, this throughput drastically decreases when the image resolution increases. On the other hand, high-resolution cameras with

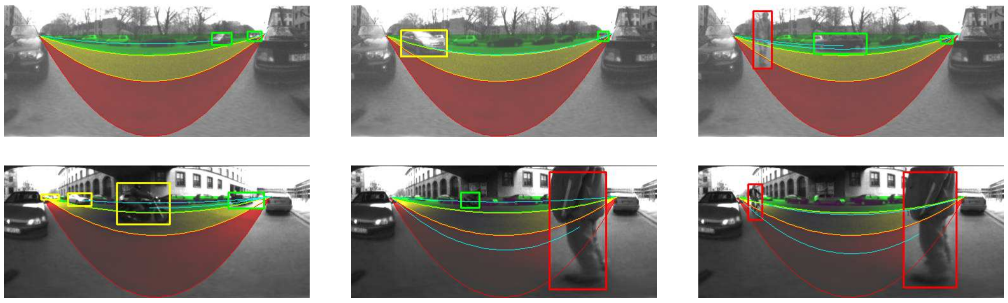


Fig. 22: Detected traffic participants and road users in a normal street (top) and in a two-lane one-way street (bottom).

frame rates of 80-120 fps would lead to a faster and more precise detection of traffic participants and are therefore highly recommended. For this reason, we introduce a potential implementation of the foreground detection algorithm on an FPGA to guarantee real-time foreground detection for high-resolution high-frame rate cameras. The parallelization scheme presented in Section 3.6 allows to port the algorithm onto small form-factor embedded systems – e.g., on low-power, cost-effective FPGA's or cameras with FPGA's integrated within the package.

In our setup, we use firewire-cameras that are suitable for stream-based FPGA implementations. However, high parallelization concepts may be preferred as image rectification, background initialization and shadow detection require information from neighboring pixels – e.g., information from pixels from the following two or three rows. But this information is normally not available for stream-based implementations. In the following sections, we propose an exemplary, highly parallel implementation of the foreground detection algorithm for FPGA-based embedded systems to guarantee real-time foreground detection for high-resolution high-frame rate cameras within at least 33ms.

### 5.1. Background Initialization

The initialization of the background as presented in Section 3.1 is the first stage for foreground detection. The background is initialized by choosing the pixels that contain background in a training sequence of $n$ frames. For each pixel position $p_i$, a similarity matrix is computed containing absolute, mean-based block differences. Background pixels are selected within a matrix decomposition stage and the initial background value is calculated using median filtering. Fig. 23 illustrates a FPGA-based hardware-architecture for background initialization for one pixel position $p_i$. Thereby, let $p_i, i \in [1, .., N \times M]$ be pixels of original images with size $N \times M$, $p_{ri}$ be pixels of rectified/transformed images and $pb_i$ be pixels used for background initialization. Original images are transformed into rectified images as a first step. For each pixel position $p_i$ in original images, a Look-Up-Table (LUT) is provided that may be stored in the external memory of the embedded system. The LUT may be obtained by camera calibration (see Section 2.1) and contains the pixel positions $p_{L1}..p_{Ln}$ in original images whose intensity values are used for bicubic interpolation within the hardware-block *Interpolation*. The resulting intensity value is stored at pixel position $p_{ri}$ in a rectified image (see block *Rectification*). The hardware module *Background Initialization* realizes the background initialization. Block-based averaging (see hardware block *Mean*) requiring intensity values from neighboring pixels $p_{r1}..p_{rm}$ may be used to build the similarity matrix on that background pixel selection is based. For each pixel position in a frame of the training sequence, the hardware block *Mean* computes the average of $m$ intensity values and stores the result $\mu_m(p_i)$ in a buffer. Based on the mean-values $\mu_m(p_i)$, the entries of the similarity matrix $d_{j,m}(p_i)$ are computed with the block *Similarity* and the results may be stored in an external memory. *Similarity* also provides a signal *control* to select the values needed for similarity matrix computation. Finally, the

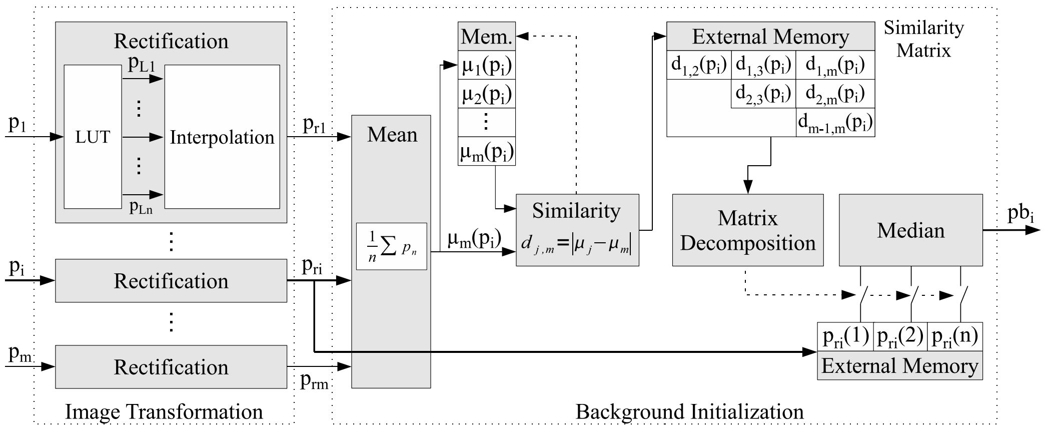


Fig. 23: Hardware architecture for background-initialization.

hardware block *Matrix Decomposition* is a realization of the decomposition algorithm proposed by Farin *et al.* [Farin et al. 2003] and selects the pixels of the training sequence that relate to background. The final intensity value of the background may be computed within the hardware module *Median* and serves as initialization for the Kalman-based background model.

## 5.2. Realization of the Object Detection Algorithm

In this section, we describe the hardware architecture for the proposed background estimation and object detection algorithm (see Section 3). Fig. 24 illustrates a potential implementation of background estimation and foreground detection for one pixel position $p_{ri}$. The architecture consists of two hardware modules, *Dynamic Background Estimation* and *Shadow Detection*. The hardware blocks *Rectification* presented in Section 5.1 may also be used to transform pixels $p_i$ from original images into pixels $p_{ri}$ that serve as input for the foreground detection module. The system state of the background module $\hat{I}_{pri}(t)$ and $\hat{\dot{I}}_{pri}(t)$ for each pixel position $p_{ri}$ may be stored in an external memory and is initialized with the background value $pb_i$.

The hardware-block *System State Prediction* considers global illumination changes $\Delta_{pri}(t)$ (see Section 3.5) and predicts the next system state to determine whether $p_{ri}$ belongs to foreground or to background (see block *FG BG*). The system state is updated using the hardware block *System State Update* by choosing different Kalman-gains $k1, 2$ depending on a detected foreground or background pixel. Block *FG BG* also generates a signal *control* that activates and deactivates the blocks *NCC/ZNCC* and *Shadow Classification* within the module *Shadow Detection*. Shadow detection is not required for background pixels, and hence the blocks *NCC/ZNCC* and *Shadow Classification* may be deactivated when a background pixel is identified. The block *NCC/ZNCC* consists of a hardware realization for NCC and ZNCC computation and calculates the energy-values $ET_{pri}$, $EB_{pri}$, $EZT_{pri}$ and $EZB_{pri}$. NCC and ZNCC computation is based on image blocks containing the pixel $p_{r1}..p_{rm}$ in an actual frames and on the corresponding image blocks in the background image (system states $\hat{I}_{pr1}..\hat{I}_{prm}$) whereas $p_{ri}$ represents the center of each image block (see Section 3.3). Block *Shadow Classification* represents a hardware-realization of the shadow classification algorithm (see Eq. 15) and distinguishes between shadow pixels or valid foreground pixels. This block also generates a control signal *class* to specify the value of the output pixel position $p_{FGi}$ that indicates the background estimation and foreground detection result – e.g., $0$ for background and shadow and $255$ for valid foreground. The result is then stored in an external memory as an image that serves as an input for further processing stages like collision risk prediction.

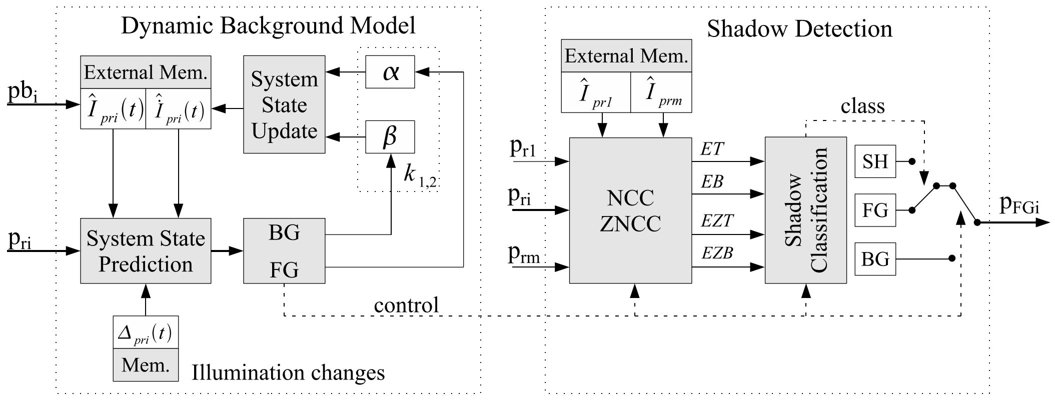


Fig. 24: Hardware architecture of the background-estimation and foreground detection algorithm.

## 6. CONCLUDING REMARKS

In this paper, we presented algorithms for robust background estimation and foreground detection in gray scale images captured by omnidirectional cameras embedded inside a smart car door. The focus lay in modifying and developing algorithms for fast object detection to quickly detect fast approaching objects for safe door operations. In this implementation, we achieved a throughput of at least 30 frames per second using a quad-core CPU. The proposed algorithm initially estimates background images even from scenarios in which no empty background is available during initialization. Foreground is segmented based on the recorded background image and based on Kalman-filtering as a next stage. Shadow pixel candidates are determined using NCC, and a refinement is carried out using ZNCC to distinguish between foreground pixels with small differences from background or shadow pixels. In order to reduce the influence of illumination changes that cause foreground detection to fail, illumination changes were detected by local search windows and compensated for by updating the background model. The proposed algorithms were successfully implemented to track objects like walking humans, motorbikes, bicycles and cars. They were parallelized to obtain attractive speedups on multi-core processors. Finally, they were evaluated against other state-of-the-art methods and showed enhanced foreground detection in our setting. We briefly described an algorithm to predict the risk of collisions and to block the car door in case of a collision. However, further studies on optimally selecting the different parameters associated with the algorithms are necessary to make them work better in a wide variety of traffic scenarios and lighting conditions such as heavy rain or snow. Another interesting direction to explore would be to implement our algorithms on small form-factor embedded platforms (e.g., ones made up of a heterogeneous collection of reconfigurable and general-purpose processors) which blend well with a standard automotive electronics setup. Therefore, we presented an implementation scheme to port the time-critical background initialization and foreground detection algorithms onto a low-power, cost-effective FPGA. Such platforms might also lead to more efficient/fast implementations and therefore better reaction times for the smart door due to a higher throughput compared to multi-core-based solutions.

## ACKNOWLEDGMENTS

The authors would like to thank Dip Goswami and the unknown reviewers for their comments and suggestions which greatly improved the paper. This work has been supported in part by BMW AG in the framework of CAR@TUM.

## REFERENCES

ACHLER, O. AND TRIVEDI, M.-M. 2002. Real-time traffic flow analysis using omnidirectional video network and flatplane transformation. In *Proceedings of the Conference on Intelligent Transportation Systems (ITS), Chicago, Illinois*.

ARDOE, H. AND BERTHILSSON, R. 2006. Adaptive background estimation using intensity independent features. In *Proceedings of the 17th British Machine Vision Conference (BMVC), Edinburgh*.

BAKER, S. AND NAYAR, S. K. 1999. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision 35,* 2, 175–196.

BAKER, S. AND NAYAR, S. K. 2001. In *Panoramic vision*. Springer-Verlag New York, Inc., Chapter Single viewpoint catadioptric cameras, 39–71.

BAYER, B. E. 1976. Color imaging array. US Patent 3971065.

BHASKAR, H., MIHAYLOVA, L., AND MASKELL, S. 2007. Background modeling using adaptive cluster density estimation for automatic human detection. *Lecture Notes in Informatics 2*, 130–134.

BRUNELLI, R. 2009. Template matching techniques in computer vision: Theory and practice. *Wiley, ISBN 978-0-470-51706-2 ([1] TM book)*.

COMMISSION, E. 2009. Europeans information society, driver-assistance systems.

DINKAR, B. AND NAYAR, S. 1996. Ordinal measures for visual correspondence. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA*.

ELGAMMAL, A., DURAISWAMI, R., HARWOOD, D., AND DAVIS, L. 2002. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE 90,* 7, 1151–1163.

FARIN, D., DE WITH, P., AND EFFELSBERG, W. 2003. Robust background estimation for complex video sequences. In *Proceedings of the Conference on Image Processing (ICIP), Barcelona, Spain*.

FRIEDMAN, N. AND RUSSEL, S. 1997. Image segmentation in video sequences: A probalistic approach. In *Proceedings of the Conference on Uncertainity in Artificial Intelligence (UAI), Providence, Rhode Island, USA*.

GHANDI, T. AND TRIVEDI, M. 2004. Motion based vehicle surround analysis using an omni-directional camera. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV), Parma, Italy*. 560 – 565.

HARITAOGLU, I., HARWOOD, D., AND DAVIS, L. 1998. W4: Who? when? where? what? a real-time system for detecting and tracking people. In *Proceedings of the Conference on Automatic Face and Gesture Recognition (FG), Nara, Japan*.

JACQUES, J., JUNG, C., AND MUSSE, S. 2005. Background subtraction and shadow detection in grayscale video sequences. In *Proceedings of the 18th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), Vale do Rio dos Sinos, Brasilia*.

KARMAN, K.-P. AND BRANDT, A. V. 1990. In *Time-varying Image Processing and Moving Object Recognition (2)*. Elsevier Publishers B.V., Chapter Moving Object Recognition Using an Adaptive Background Memory, 297–308.

MASSEY, M. AND BENDER, W. 1996. Salients stills: Process and practice. *IBM Systems Journal 35,* 3.4, 557–573.

OKUTOMI, M. AND KANADE, T. 1993. A multiple-baseline stereo. *Transactions on Pattern Analysis and Machine Intelligence 15,* 4, 353–363.

RABINER, R. L. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77,* 2, 257–286.

RIDDER, C., MUNKELT, O., AND KIRCHNER, H. 1995. Adaptive background estimation and foreground detection using kalman-filtering. In *Proceedings of the International Conference on Recent Advances in Mechatronics (ICRAM), Istanbul, Turkey*.

SCARAMUZZA, D., MARTINELLI, A., AND SIEGWART, R. 2006a. A flexible technique for accurate omnidirectional omnidirectional camera calibration and structure from motion. In *Proceedings of the IEEE Conference on Computer Vision Systems (ICVS), New York, USA*.

SCARAMUZZA, D., MARTINELLI, A., AND SIEGWART, R. 2006b. A toolbox for easily calibration omnidirectional cameras. In *Proceedings of the IEEE International Conference on Intelligent Systems (IROS), Beijing, China*.

SCHARFENBERGER, C., BOEHM, F., AND FAERBER, G. 2009. Image rectification: Evaluation of various projections for omnidirectional vision sensors using the pixel density. In *Proceedings of the Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), Lisbon, Portugal*.

STENGER, D., RAMESH, V., PARAGIOS, N., COETZEC, F., AND BUHMANN, J. 2001. Topology free hidden markov models: application to background modeling. In *Proceedings of the International Conference on Computer Vision (ICCV), Vancouver, British Columbia, Canada*.

STROLZ, M., MÜHLBAUER, Q., SCHARFENBERGER, C., FÄRBER, G., AND BUSS, M. 2008. Towards a generic control system for actuated car doors with arbitrary degrees of freedom. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Eindhoven, The Netherlands*.

SUHR, J., JUNG, H., BAE, K., AND KIM, J. 2010. Vorrichtung und Verfahren zum monokularen Motion-Stereo-basierten Detektieren von freien Parkplätzen. German Patent DE102009012435A1.

ZABIH, R. AND WOODFILL, J. 1994. Non-parametric local trans-forms for computing visual correspondence. In *Proceedings of the European Conference on Computer Vision (ECCV), Stockholm, Sweden*.

ZHONG, J. AND SCLAROFF, S. 2003. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *Proceedings of the International Conference on Computer Vision (ICCV), Nice, France*.

ZIVKOVIC, Z. AND HEIJDEN, F. 2006. Efficient adaptive density estimation per image pixel for task of background subtraction. *Pattern Recognition Letters 27*, 773–780.