Computer Vision in Vehicles
- A Brief Introduction -

Reinhard Klette

The University of Auckland

CONTENTS

Monday 3rd November, 2014, 22:26

Computer Vision in Vehicles

This report is a brief introduction to academic aspects of computer vision in vehicles. It discusses lower- or higher-order tasks and single- or multi-module solutions. An Appendix at the end of the report briefly summarises notation and definitions used.

1.1 Introduction

Computer vision designs solutions for understanding the real world by using cameras. Computer vision operates today in *vehicles* including cars, trucks, airplanes, UAVs such as multi-copters (see Fig. 1.1 for a quadcopter), satellites, or even autonomous driving rovers on the Moon or Mars. The *ego-vehicle* is that vehicle where the system operates in; *ego-motion* describes the ego-vehicle's motion in the real world.

Computer vision solutions are today in use in manned vehicles for improved safety or comfort, in autonomous vehicles (e.g. robots) for supporting motion or action control, but also for misusing unmanned aerial vehicles (UAVs) for killing people remotely; a technology which also has good potentials for helping to save lives, to create 3-dimensional (3D) models of the environment, and so forth.

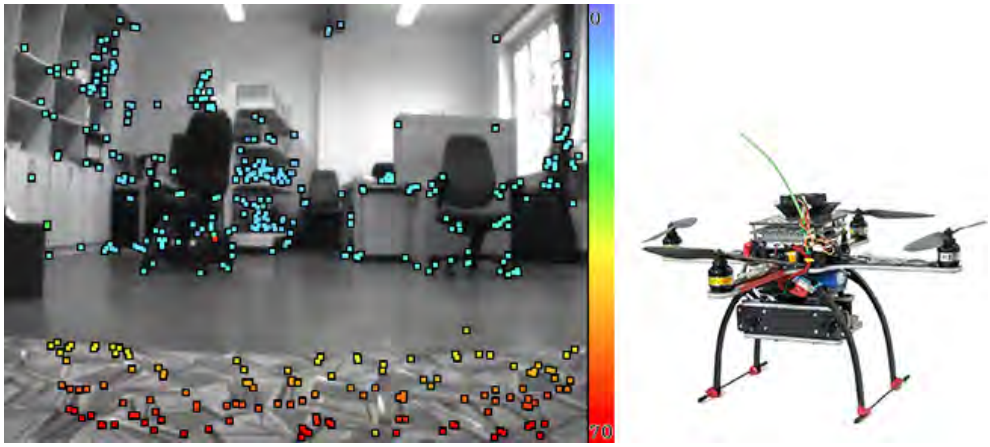

Figure 1.1 *Left*: Corners detected from a flying quadcopter using a modified FAST (see the Appendix for the definition of corners as an example for image features, and FAST as an example for a feature detector). *Right*: Quadcopter. Courtesy of Konstantin Schauwecker

Traffic safety is a dominant application area for computer vision in vehicles. Currently, about 1.24 million people die annually worldwide due to traffic accidents (WHO 2013), this is that on average about 2.4 people die *every minute* in traffic accidents. How does this compare to the numbers Western politicians are using for obtaining support for their "war on terrorism"? Computer vision can play a major role in solving the true real-world problems; see Fig. 1.2. Traffic-accident fatalities can be reduced by controlling traffic flow (e.g. by triggering automated warning signals at pedestrian crossings or intersections with bicycle lanes) using stationary cameras, or by having cameras installed in vehicles (e.g. for detecting save distances and adjusting speed accordingly, or by detecting obstacles and constraining trajectories).

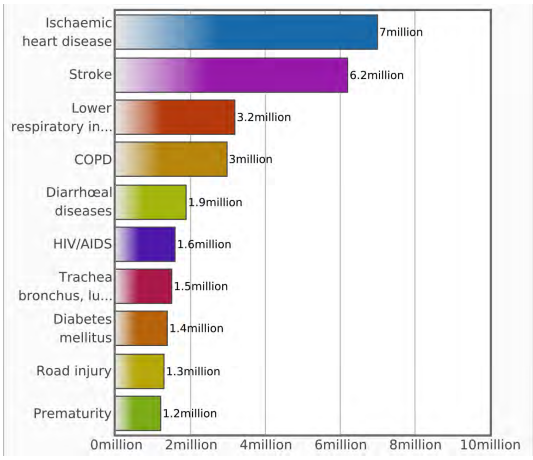

Figure 1.2 The ten leading causes of death in the world. Chart provided online by the World Health Organisation (WHO). Road injury ranked number 9 in 2011.

The Mars rovers "Curiosity" and "Opportunity" operate based on computer vision; "Opportunity" has already for 10 years. The visual system of human beings provides a proof of existence that vision alone can deliver nearly all of the information required for steering a vehicle. Computer vision aims at creating comparable automated solutions for vehicles, enabling them to navigate safely in the real world. Additionally, computer vision can also work constantly "at the same level of attention", applying the same rules or programs; a human is not able to do so due to becoming tired or distracted. A human applies accumulated knowledge and experience (e.g. supporting intuition), and it is a challenging task to embed a computer vision solution into a system able to have, for example, intuition.

As a brief summary, computer vision is an important component of intelligent systems for vehicle control (e.g. in modern cars, or in robots), and it offers many more opportunities for future developments in a vehicle context.

1.2 Tasks

Tasks in computer vision can be classified as being of *lower order* for solving a dedicated problem close to the signal level of input data (e.g. stereo matching or optical flow

calculation, face detection, traffic-sign detection, or lane-border detection). Tasks can also be classified to be of *higher order*, aiming at solving a more advanced problem in an image-understanding pipeline (e.g. free-space analysis, ego-motion analysis, tracking of other vehicles, or wrong-lane detection).

1.2.1 Lower and Higher Order versus Single and Multiple Modules

Solutions for lower-order tasks typically aim at creating one self-contained *module* for potential integration into a complex system. Solutions for higher-order tasks are typically structured into multiple modules which interact in a complex system.

Figure 1.3 Two screenshots for real-view navigation. Courtesy of the authors of (Choi et al. 2010)

Single- and Multi-Module Solutions for Visual Lane Analysis For example, (Shin *et al.* 2014) review visual lane analysis for driver-assistance systems or autonomous driving. In this context the authors discuss single- and multi-module solutions; as examples for multi-module solutions the authors list "the combination of visual lane analysis with driver monitoring..., with ego-motion analysis..., with location analysis..., with vehicle detection..., or with navigation...". They illustrate the latter example by an application shown in Figure 1.3: lane detection and road sign reading, the analysis of GPS data and electronic maps (*e-maps*), and 2-dimensional (2D) visualization are combined into a real-view navigation system (Choi et al. 2010).

A General View on Multi-module Solutions Designing a multi-module solution for a lower- or higher-level task does not need to be more difficult than designing a single-module solution. In fact, some single-module solutions (e.g. for motion analysis) are very advanced. But multiple modules require control (e.g. when many designers separately insert processors for controlling various operations in a vehicle, no control engineer should have been surprised if the vehicle becomes unstable). Designing a multi-module solution requires

1. that solutions are available and known for relevant single-module solutions,
2. tools for evaluating those solutions in dependency of a given situation (or *scenario*) for being able to select (or adapt) single-module solutions,
3. conceptual thinking for designing and controling an appropriate multi-module system, and,

4. for system optimization, more extensive testing of such a system on various scenarios than for a single-module solution (due to the increase in combinatorial complexity of multi-module interactions).

The report provides fundamentals for single- and multi-module solutions, basically by discussing examples.

Accuracy and Robustness Solutions can be characterized as being *accurate* or *robust*. *Accuracy* means correctness for a given scenario. *Robustness* means approximate correctness for a set of scenarios which also include cases of challenging scenarios. In both cases, it would be appropriate to specify the defining scenarios accurately, e.g. by using video descriptors (Briassouli and Kompatsiaris 2011) or data measures (Suaste et al. 2013). Ideally, robustness should address *any* possible scenario in the real world for a given task.

1.2.2 Examples of Tasks

This section briefly outlines some of the tasks which need to be solved by computer vision in vehicles; they do not require *per se* either a single- or a multi-module solution. (A more detailed view on a few selected tasks, together with some comments on solutions, follows in the next section).

Motion Analysis A sequence of video frames $I(.,.,t)$, all defined on the same carrier Ω, is recorded with a time difference δt between two subsequent frames; frame t is recorded at time $t \cdot \delta t$ counted from the start of the recording.

The projection of a surface point into pixel $p_t = (x_t, y_t)$ in frame t, and into pixel $p_{t+1} = (x_{t+1}, y_{t+1})$ in frame $t-1$, defines a pair of *corresponding pixels* represented by a *motion vector* $[x_{t+1} - x_t, y_{t+1} - y_t]^\top$ from p_t to p_{t+1} in Ω.

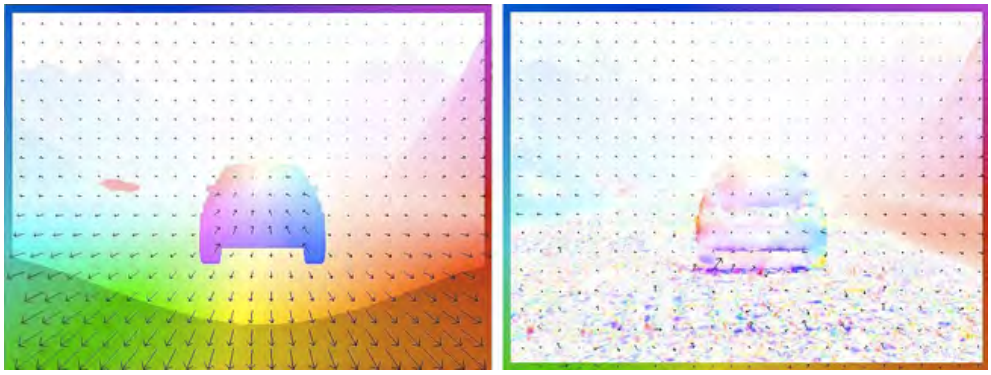

Figure 1.4 Visualization of optical flow using the colour key shown around the border of the image for assigning a direction to particular colours; the length of the flow vector is represented by saturation, where value 'White' (i.e. undefined saturation) corresponds to 'no motion'. *Left*: Ground truth for the image shown on the left of Figure 1.8. *Right*: Calculated optical flow using the original Horn-Schunck algorithm. Courtesy of Tobi Vaudrey

Dense motion analysis aims at calculating approximately-correct motion vectors for "basically" every pixel location $p = (x, y)$ in frame t; see Fig. 1.4 for an example. *Sparse*

motion analysis is designed for having accurate motion vectors at a few selected pixel locations.

Motion analysis is a difficult 2D correspondence problem, and it might become easier by having recorded high-resolution images at a higher frame rate in future. For example, motion analysis is approached by a single-module solution by *optical flow* calculation, or as a multi-module solution when combining image segmentation with subsequent estimations of motion vectors for image segments.

Distance Analysis *Stereo vision* is the dominant approach in computer vision for calculating distances. *Corresponding pixels* are here defined by projections of the same surface point in the scene into the left and right image of a stereo pair. After having recorded stereo pairs rectified into canonical stereo geometry, 1-dimensional (1D) correspondence search can be limited to identical image rows.

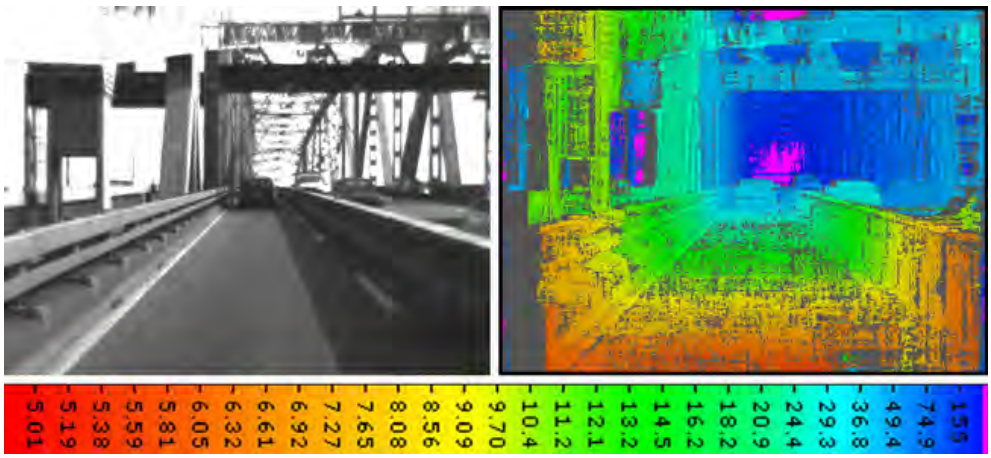

Figure 1.5 *Left*: Image of a stereo pair. *Right*: Visualization of a depth map using the colour key shown at the bottom for assigning distances in metres to particular colours. A pixel is shown in grey if there was low confidence for the calculated disparity value at this pixel. Courtesy of Simon Hermann

Corresponding pixels define a *disparity*, which is mapped based on camera parameters into *distance* or *depth*. There are already very accurate solutions for stereo matching, but challenging input data (rain, snow, dust, sunstroke, running wipers, and so forth) still pose unsolved problems. See Fig. 1.5 for an example of a depth map.

Laser range-finders are increasingly used for estimating distance basically based on the time-of-flight principle. Assuming sensor arrays of larger density in the near future, laser range-finders will become another option for cost-efficient accurate distance calculations. Combining stereo vision with distance data provided by laser range-finders is a promising multi-module approach towards distance calculations.

Environment Analysis There are static (i.e. fixed with respect to the Earth) or dynamic objects in a scenario which need to be detected, understood, and possibly further analysed.

A flying helicopter (or just multi-copter) should be able to detect power lines or other potential objects defining a hazard. Detecting traffic signs or traffic lights, or understanding

lane borders of highways or suburban roads are examples for driving vehicles. Boats need to detect buoys and beacons.

Pedestrian detection became a common subject for road-analysis projects. After detecting a pedestrian on a pathway next to an inner-city road, it would be helpful to understand whether this pedestrian intends to step onto the road in the next few seconds.

After detecting more and more objects, we may have the opportunity to model and understand a given environment.

Tracking Object tracking is an important task for understanding the motion of a mobile platform, or of other objects in a dynamic environment. The mobile platform with the installed system is also called the *ego-vehicle* whose *ego-motion* needs to be calculated for understanding the movement of the installed sensors in the 3-dimensional (3D) world.

Calculated features in subsequent frames $I(.,.,t)$ can be tracked (e.g. by using RANSAC for identifying an affine transform between feature points) and then used for estimating ego-motion based on bundle adjustment. This can also be combined with another module using non-visual sensor data such as GPS or of an inertial measurement unit (IMU).

Other moving objects in the scene can be tracked using repeated detections, or by following a detected object in frame $I(.,.,t)$ to frame $I(.,.,t+1)$. A *Kalman filter* (e.g. linear, general, or unscented) can be used for building a model for the motion as well as for involved noise. A *particle filter* can also be used based on extracted weights for potential moves of a particle in particle space.

Figure 1.6 Face detection, eye detection, and face tracking results under challenging lighting conditions. Courtesy of Mahdi Rezaei

Driver Monitoring Besides (e.g.) measurements for understanding the steadiness of driver's movement of the steering wheel, cameras are also an appropriate tool for understanding the state of the driver (e.g. drowsiness detection, or eye gaze).

Face and eye detection (Viola and Jones 2001), or head-pose analysis (Murphy-Chutorian and Trivedi 2009) are basic tasks in this area. Challenging lighting conditions still define unsatisfactorily solved scenarios; for example, see (Rezaei and Klette 2012) for such scenarios.

Driver awareness can be defined by relating driver monitoring results to environment analysis for the given traffic scenario. The driver not only needs to pay attention to driving; eye gaze or head pose (Rezaei and Klette 2011) should also correspond (for some time) to those outside regions where safety-related events occur.

Environment Reconstruction 3D road-side visualization or 3D environment modelling are applications where a 3D reconstruction from a moving platform can be used (Xiao et al. 2009), possibly in combination with 3D reconstructions from a flying platform such as a multi-copter.

There are unresolved issues in the required very high accuracy of ego-motion analysis for mapping 3D results obtained at time t into a uniform world-coordinate system. This is in particular apparent when trying to unify results from different runs through the same street (Zeng and Klette 2013). See Fig. 1.7 for 3D results from a single run and Fig. 1.19 for combined results from two runs (for the same site at Tamaki campus, Auckland).

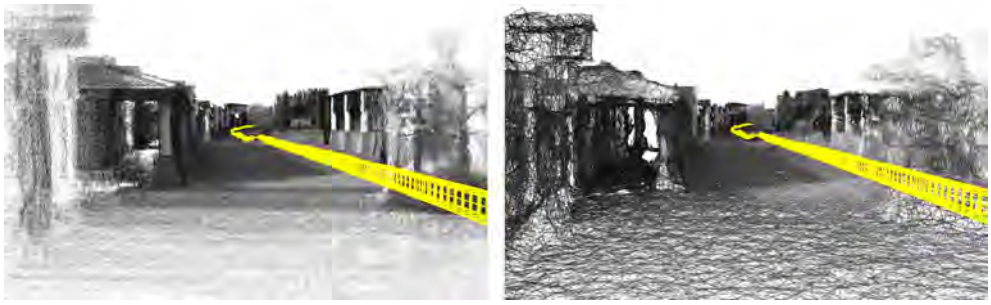

Figure 1.7 Reconstructed cloud of points and surface based on a single run of the ego-vehicle. Courtesy of Yi Zeng

Comparative Performance Evaluation An efficient way for a comparative performance analysis of solutions for one task is by having different authors testing their own programs on identical benchmark data. But we not only need to evaluate the programs, we also need to evaluate the benchmark data used (Haeusler and Klette 2010, 2012).

Benchmarks need to come with *measures* for quantifying performance such that we can compare accuracy on individual data, or robustness across a diversity of different input data.

Figure 1.8 Examples of benchmark data available for a comparative analysis of computer vision algorithms for motion and distance calculations. *Left*: Image of a synthetic image sequence provided on EISATS with accurate ground truth. *Right*: Image of a real-world sequence provided on KITTI with approximate ground truth

Figure 1.8 illustrates two possible ways for generating benchmarks, one by using computer graphics for rendering sequences with accurately-known ground truth,[1] and another one by using high-end sensors (in the illustrated case, ground truth is provided by the use of a laser range-finder).[2]

[1]For EISATS benchmark data, see `www.mi.auckland.ac.nz/EISATS`.
[2]For KITTI benchmark data, see `www.cvlibs.net/datasets/kitti/`.

But those evaluations need to be considered with care. Not everything can be compared with everything. Evaluations depend on the used benchmark data; having a few summarizing numbers for those may not be really of relevance for particular scenarios possibly occurring in the real world. For some input data we can also simply not answer how a solution performs; for example, in the middle of a large road intersection we cannot answer which lane border detection algorithm performs best for this scenario.

There is no Winner We are not so naive to expect an all-time "winner" when comparatively evaluating computer vision solutions. Vehicles operate in the real world (whether on Earth, the Moon, or on Mars), which is so diverse that not all of the possible event occurrences can be modelled in underlying constraints for a designed program. Particular solutions perform differently for different scenarios, a winning program for one scenario may fail for another one. We can only evaluate how particular solutions perform for particular scenarios. At the end, this might support an optimization strategy by adaptation to a current scenario a vehicle experiences at a time.

1.3 Examples for Single-Module Solutions

This, and the next section, illustrate components for single- or multi-module solutions for particular low- or high-level tasks, and comment briefly on tools for evaluating those solutions for real-world scenarios.

1.3.1 Optical Flow

Optical flow $\mathbf{u}(p,t) = [u(p,t), v(p,t)]^\top$ is the result of dense motion analysis. It represents motion vectors between corresponding pixels $p = (x, y)$ in frames $I(.,.,t)$ and $I(.,.,t+1)$. See Fig. 1.4 for a visualization of an optical flow map.

Optical Flow Equation and Image Constancy Assumption The derivation of the *optical flow equation* (Horn and Schunck 1981)

$$0 = u\,(p,t) \cdot I_x\,(p,t) + v\,(p,t) \cdot I_y\,(p,t) + I_t\,(p,t) \tag{1.1}$$

for $p \in \Omega$ and first-order derivatives I_x, I_y, and I_t, follows from the *image constancy assumption* (ICA), i.e. by assuming that corresponding 3D world points are represented in frame t and $t+1$ by the same intensity. This is actually not true for computer vision in vehicles. Light intensities change frequently due to lighting artefacts (e.g. driving below trees), changing angles to the sun, or simply due to sensor noise. However, the optical flow equation is often used as *data-cost term* in a Markov random-field (MRF) approach for solving the optical flow problem. See the Appendix for a brief definition of the MRF approach which applies data-cost and smoothness-cost terms.

Examples of Data and Smoothness Costs If we accept Equ. (1.1) due to Horn and Schunck (and thus the validity of the ICA) as data constraint, then we derive

$$E_{HS}(f) = \sum_{p \in \Omega} [\; u(p,t) \cdot I_x(p,t) + v(p,t) \cdot I_y(p,t) + I_t(p,t) \;]^2 \tag{1.2}$$

as a possible data-cost term, for any given time t.

The Appendix introduces the zero-mean normalized census-cost function E_{ZCEN}. The sum $E_{ZCEN}(f) = \sum_{p\in\Omega} E_{ZCEN}(p, p + f(p))$ can replace $E_{HS}(p, q)$ in an MRF approach as defined by Equ. (1.16) in the Appendix; see (Hermann and Werner 2013). This corresponds to the invalidity of the ICA for video data recorded in the real-world.

For the smoothness-error term we may use

$$E_{FO\_L_2}(f) = \sum_{p\in\Omega} \left[\frac{\partial u(p,t)}{\partial x}\right]^2 + \left[\frac{\partial u(p,t)}{\partial y}\right]^2 + \left[\frac{\partial v(p,t)}{\partial x}\right]^2 + \left[\frac{\partial v(p,t)}{\partial y}\right]^2 \tag{1.3}$$

This smoothness-error term applies squared penalties to first-order derivatives in the L_2 sense. Applying a smoothness term in an approximate L_1 sense reduces the impact of outliers (Brox et al. 2004).

Terms E_{HS} and $E_{FO\_L_2}$ define the TVL$_2$ optimization problem as originally considered by (Horn and Schunck 1981).

Performance Evaluation of Optical Flow Solutions Besides using data with provided ground truth (see EISATS and KITTI and Fig. 1.8), there is also a way for evaluating calculated flow vectors on recorded real-world video assuming that the recording speed is sufficiently large; for calculated flow vectors for frames $I(.,.,t)$ and $I(.,.,t+2)$, we calculate an image "half-way" using the mean of image values at corresponding pixels, and we compare this calculated image with frame $I(.,.,t+1)$; see (Szeliski 1999). Limitations for recording frequencies of current cameras make this technique not yet practically appropriate, but it is certainly appropriate for fundamental research.

1.3.2 Stereo Vision

We address the detection of corresponding points in a stereo image $I = (L, R)$ (as defined in the Appendix), a basic task for distance calculation in vehicles using binocular stereo.

Binocular Stereo Vision After camera calibration we have two virtually-identical cameras C^L and C^R which are perfectly aligned defining *canonical stereo geometry*. In this geometry we have an identical copy of the camera on the left translated by *base distance* b along the X_s-axis of the $X_sY_sZ_s$ camera coordinate system of the left camera. The projection centre of the left camera is at $(0, 0, 0)$ and the projection centre of the cloned right camera is at $(b, 0, 0)$. A 3D point $P = (X_s, Y_s, Z_s)$ is mapped into undistorted image points

$$p_u^L = (x_u^L, y_u^L) = \left(\frac{f \cdot X_s}{Z_s}, \frac{f \cdot Y_s}{Z_s}\right) \tag{1.4}$$

$$p_u^R = (x_u^R, y_u^R) = \left(\frac{f \cdot (X_s - b)}{Z_s}, \frac{f \cdot Y_s}{Z_s}\right) \tag{1.5}$$

in the left and right image plane, respectively. Considering p_u^L and p_u^R in homogeneous coordinates, we have that

$$[p_u^R]^\top \cdot \mathbf{F} \cdot p_u^L = 0 \tag{1.6}$$

for the 3×3 *bifocal tensor* $\mathbf{F}$, defined by the configuration of the two cameras. The dot product $\mathbf{F} \cdot p_u^L$ defines an *epipolar line* in the image plane of the right camera; any stereo-point corresponding to p_u^L needs to be on that line.

Binocular Stereo Matching Let B be the *base image* and M be the *match image*. We calculate corresponding pixels p^B and q^M in xy image coordinates of carrier Ω following the MRF approach (as expressed by Equ. (1.17) in the Appendix). A labelling function f assigns a disparity f_p to pixel location p which specifies a corresponding pixel $q = p^f$.

For example, we can use the census data-cost term $E_{ZCEN}(p, p^f)$ (as defined in Equ. (1.18) of the Appendix), and for the smoothness-cost term either the Potts model, linear truncated cost, or quadratic truncated costs; see Chapter 5 in (Klette 2014). Chapter 6 of this book discusses also different algorithms for stereo matching including *belief-propagation matching* (BPM) (Sun et al. 2003)(Guan and Klette 2008) and *dynamic-programming matching* (DPM). DPM can be based on scanning along the epipolar line only, and using either an ordering or a smoothness constraint, or on scanning along multiple scanlines and using a smoothness constraint along those lines; the latter case is known as *semi-global matching* (SGM) if multiple scanlines are used for error minimization (Hirschmüller 2005).

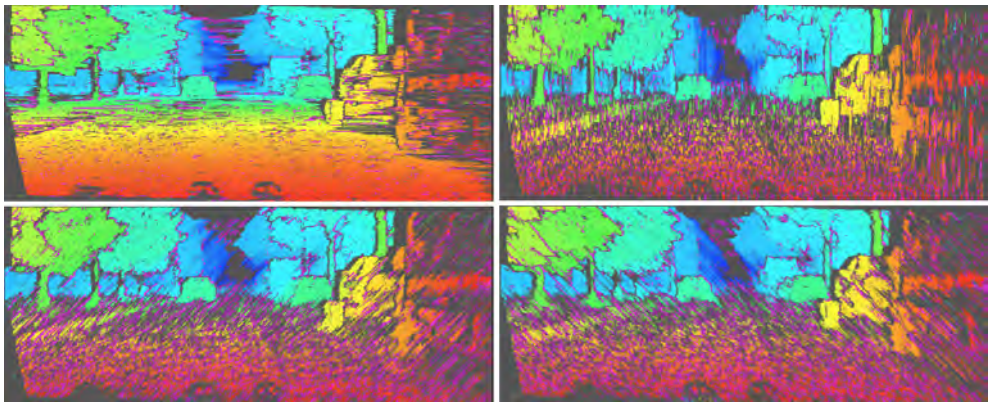

Figure 1.9 Resulting disparity maps for stereo data when using only *one* scanline for DPM with the SGM smoothness constraint and a 3×9 ZCEN data cost function. *From top to bottom*: Horizontal scanline (left to right), diagonal scanline (lower left to upper right), vertical scanline (top to bottom), and diagonal scanline (upper left to lower right). Pink pixels are for low-confidence locations (here identified by inhomogeneous disparity locations). Courtesy of Simon Hermann; the input data have been provided by Daimler A.G.

Iterative SGM (iSGM) modified SGM; for example, error-minimization along the horizontal scan line should in general contribute more to the final result than optimization along other scan lines (Hermann and Klette 2012). See Fig. 1.9 which also addresses confidence measurement; for a comparative discussion of confidence measures, see (Haeusler and Klette 2012). *Linear BPM* (linBPM) applies the ZCEN data-cost term and the linear truncated smoothness-cost term (Khan et al. 2013).

Performance Evaluation of Stereo Vision Solutions Figure 1.10 provides a comparison of iSGM versus linBPM on four frame sequences each of 400 frames length. It illustrates that iSGM performs better (with respect to the used measure; see the next paragraph for its definition) on the `bridge` sequence which is characterized by many structural details in the scene, but not as good as linBPM on the other three sequences. For sequences `dusk` and

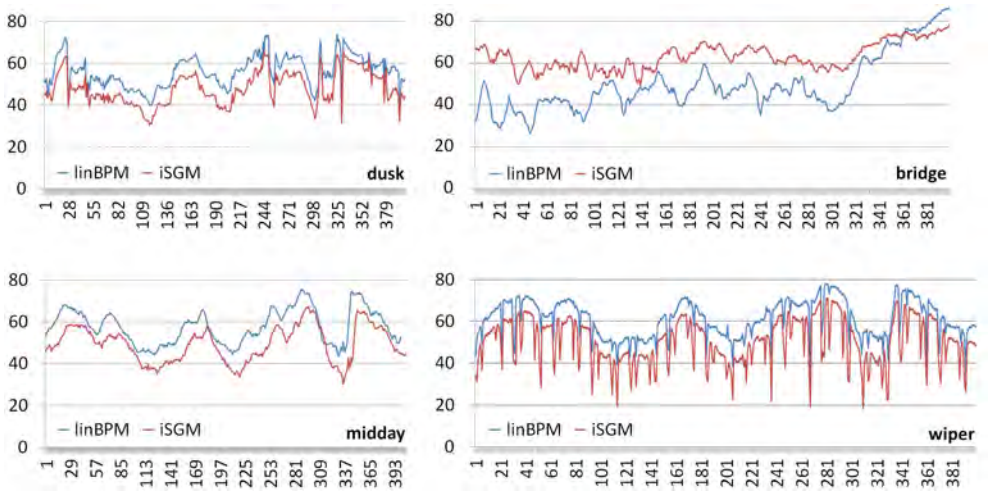

Figure 1.10 NCC results when applying the third-eye technology for stereo matchers iSGM and linBPM for four real-world trinocular sequences of Set 9 of EISATS. Courtesy of Waqar Khan, Veronica Suaste, and Diego Caudillo

`midday`, both performances are highly correlated, but not for the other two sequences. Of course, evaluating on only a few sequences of 400 frames each is insufficient for making substantial evaluations, but it does illustrate performance.

Figure 1.11 Laplacians of smoothed copies of the same image using `cv::GaussianBlur` and `cv::Laplacian` in `OpenCV`, with values 0.5, 1, 2, and 4, for parameter σ for smoothing. Linear scaling is used for better visibility of the resulting *Laplacians*. Courtesy of Sandino Morales

The diagrams in Fig. 1.10 are defined by the *normalized cross-correlation* (NCC) between a recorded third frame sequence and a virtual sequence calculated based on the stereo-matching results of two other frame sequences. This *third-eye technology* (Morales and Klette 2009) also uses masks such that only image values are compared which are close to step-edges (for example, see Fig. 1.11 for detected edges at bright pixels in LoG scale space) in the third frame. It enables us to evaluate performance on any calibrated trinocular frame sequence, recorded in the real world.

1.3.3 Lane-borders

In a general sense, a *lane* L is defined by sufficient width for driving a road vehicle; it is the space between a left and a right *lane border*, being arcs γ_L and γ_R respectively on a 2D manifold (which is, obviously, not just a plane in general). Many different models have been used for defining lanes (e.g. analytically defined curves or sequences of individual border points following some kind of systematic pattern). There is already a vide variety of solutions available for this module of a complex computer-vision system for on-road vehicle control (Bar Hillel et al. 2012; Kim 2008; McCall and Trivedi 2006; Shin *et al.* 2014).

Solved? Lane detection is sometimes characterized as being "solved". However, this is only correct for the more than, say, 90% of scenarios during driving where lane markings, lane geometry, and visibility conditions are reasonable.[3] There is still a need for studying lane-border detectors or trackers for challenging scenarios. Can we really claim that something, which is non-trivial by nature, is "solved" knowing about all the surprises (e.g. underground road intersections, unpaved roads, or very wide road intersections without any lane marking) which may occur in the real world?

There is also not yet any satisfying automatic evaluation available for quantifying the performance of a lane detector. For example, we could claim that "*lane borders are correctly detected if they are within an error of at most 5 cm to the true lane border*". Between what minimum and maximum distance to the car? What exactly is the "true lane border"? How to measure for cases as illustrated in Fig. 1.12? KITTI offers a few manually-labelled frames for evaluating lane detection.

Figure 1.12 Three examples for data provided by (Borkar et al. 2010) where a used lane detector follows its strategy for detecting lane borders due to temporal inference, but where one given frame alone be insufficient for a judgement whether the detected border is correct or not. The three images show detected lane borders composed of sequences of individual points. Courtesy of Bok-Suk Shin

[3]Percentage can be estimated as being lower for many countries.

Performance Evaluation of Lane Detectors The detection of lane borders is sometimes even a challenge for human vision. Lane borders can often not be identified in an individual frame; see Fig. 1.12. Additional knowledge such as the width of the car or the previous trajectory of the car can be used for estimating the continuation of lanes. The authors of (Shin *et al.* 2014) also write: "The localisation of a lane is not always uniquely defined in the real world; it may depend on traffic flow or driving comfort if there is no unique lane marking."

(Borkar et al. 2010) propose a semi-automatic technique for generating ground truth for lane detection. They use *time slices*, being defined by taking a specified single row with detected lane locations in subsequent frames, and fit splines to the resulting sequences of individual points in such time slices. By specifying different rows, different time slices are created. The proposed approach works reasonably well on clearly-marked roads. The involved interaction comes with the risk of human error and limited usability.

1.3.4 Road Signs

Road signs are traffic signs (stop sign, speed sign, etc.) or any form of written (directions, weather conditions, closure times of a lane etc.) or graphically expressed information (pedestrian crossing, speed bump, icons, etc.) on or near the road which are of relevance for driving a vehicle on this road. Classes of road signs can define one particular module of a complex computer-vision system for on-road vehicle control.

Traffic Sign Recognition A standard approach (Escalera et al. 2011) can be briefly sketched as follows: possibly preprocess an image (e.g. by mapping a colour image into HSV colour space), detect geometric shapes (circles or polygons) which are potential candidates for a traffic sign (possibly using colour as a guide as well), extract features (such as, e.g., SIFT; see the Appendix for a definition of features), and compare those with features of a data base of traffic signs.

Figure 1.13 *Left*: Sift features in an input image. *Middle*: Detected sign due to voting by SIFT features which passed the potential location filter. *Right*: Diversity of the appearance of the P30 sign in New Zealand. Courtesy of Feixiang Ren

Solutions can be classified in general by focusing either more on the use of colour, or more on the use of shape for the initial detection. For example, circles can be detected by using a Hough transform or a radial-symmetry approach (Barnes and Zelinsky 2004). Recorded images are subdivided into regions of interest (i.e. left and right of the road) for having

size-priors for traffic signs in those regions. See Fig. 1.13, left, for a case when detecting SIFT features uniformly, all over the image, in the middle when restricting the search to regions of interest, and on the right for illustrating the diversity of traffic signs. Traffic sign categorization is a main subject in (Escalera et al. 2011).

Performance Evaluation of Traffic Sign Detectors The authors of (Müller-Schneiders et al. 2008) suggest an evaluation methodology for traffic sign recognition by specifying measures for comparing ground truth with detected signs, following basically set-theoretical measures defined in (Smith et al. 2005); see also the Appendix for object detectors and evaluation measures. Of course, before applying this methodology the ground truth needs to be available, and so far it is provided manually. GPS and e-maps allow us to compare locations of detected traffic signs with mapped locations of signs.

1.3.5 Face Detection

For monitoring driver awareness by a camera, basic tasks can be face detection, eye detection, face and eye analysis, or head-pose identification and analysis. We consider face detection; face detection is then typically followed by eye detection and an analysis of the state of the eyes. See Fig. 1.14.

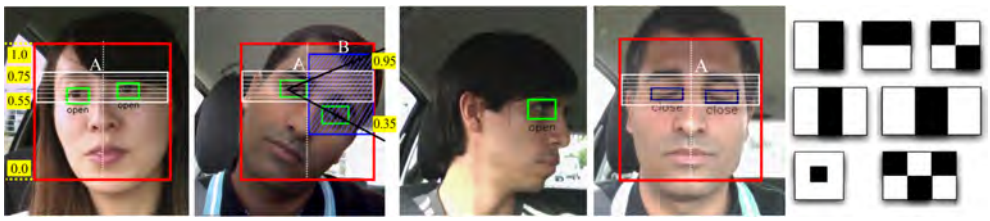

Figure 1.14 Detected eyes in a (supposed to be) driver's face based on defining a region of interest within a detected face for expected eye locations. *Right*: Examples of Haar wavelets. Courtesy of Mahdi Rezaei

The Viola-Jones Object Detection Technique. The technique (Viola and Jones 2001) was primarily motivated by face detection. A search window scans through the current input image. At each location we apply one strong classifier H which has been trained on pre-classified data by:

1. The generation of w weak classifiers h_j, $1 \leq j \leq w$, each defined by one of w given *masks* M_j; a mask contains k_j (e.g. 2 to 4) *Haar wavelets* (see Fig. 1.14, right); masks systematically scan the image, and start a new scan with a modified size of the masks.
2. The use of a statistical boosting algorithm (e.g. AdaBoost) to assemble those w weak classifiers $h_1, \ldots, h_w$ into the strong classifier H.

At the scale of each used Haar wavelet, we have weights $\omega_i > 0$ which adjust the influence of the black or white rectangles, a parity $\rho \in \{-1, +1\}$ defining whether "black-white" should match either "dark-bright" or "bright-dark", and a threshold $\theta > 0$ defining the sensitivity of the Haar wavelet for detecting a Haar feature.

At the scale of each mask M_j, $1 \leq j \leq w$, we have a threshold τ_j defining when the Haar descriptor $\mathcal{D}(M_{j,p})$ defines value $h_j = +1$; in this case the weak classifier h_j *indicates* that there might be an object (i.e. a face). The w weak classifiers are mapped into one strong classifier H.

Face Detection and Post-Processing. The w results of the weak classifiers $h_1, \ldots, h_w$ at a position of the search window are passed on to the trained strong classifier H for detecting either a face or a no-face situation. If a face is detected then its position is identified with the rectangular box of the current search window.

After classification, usually there are multiple overlapping object detections at different locations and of different sizes around a visible object. Post-processing returns a single detection per object. Methods applied for post-processing are usually heuristic (e.g. taking some kind of a mean over the detected rectangular boxes).

Performance Evaluation of Face Detection Similar to the case of traffic sign detection, it is common to use manually identified ground truth (i.e. available training or test data bases) and measures such as PR, RC, MR, or FPPI (see the Appendix) for quantifying detection success.

1.3.6 Pedestrian Detection

A standard procedure is as follows: at first a *bounding box* (a window) is detected as the region of interest (RoI) which possibly contains a pedestrian. Apply a classifier for this bounding box for detecting a pedestrian. This classifier can be based on a *histogram of gradients* (HoG) for the bounding box (Dalal and Triggs 2005); after deriving *HoG descriptors*, the classifier uses those for deciding about the presence of a pedestrian. It is also possible to use such HoG descriptors within a *random decision forest* (RDF) (Breiman 2001) for training the *split functions* of the trees of this forest.

Localizing a Bounding Box An exhaustive search, i.e. passing windows of expected bounding-box sizes through the image at all the relevant places, is time-consuming. A possible and more efficient way is to use disparity maps produced by a stereo matcher.

Disparity-based segmentation is extensively used for pedestrian detection (Gómez 2010; Zhao and Thorpe 2000) by identifying back- and foreground areas in an image before applying intensity-based segmentation. A possible approach is to evaluate in available disparity maps relationships between camera-to-object distances, height of object in the image (in pixels), and the corresponding disparity range. For example, a person, standing 10 metres away from the camera, might appear to be about 260 pixels tall in the image plane.

Split Functions A detected window can be passed down a decision tree where decisions at a non-leaf node of such a tree are made by "asking" a *split function*. Those split functions (and thus the trees, and in conclusion the forest) are generated when training the RDF on a given set of test or training data. Only very few components of a HoG descriptor are used for defining a split function at a particular node.

Leaf Nodes and Strong Classifier The leaf nodes of a tree of the trained RDF are labelled with class probabilities (e.g. just for the binary case: "pedestrian" or "not a pedestrian"). A given bounding box travels down all the w trees of the given RDF and ends up in a leaf node in each tree. Each tree defines a weak classifier, and we need to derive one (final)

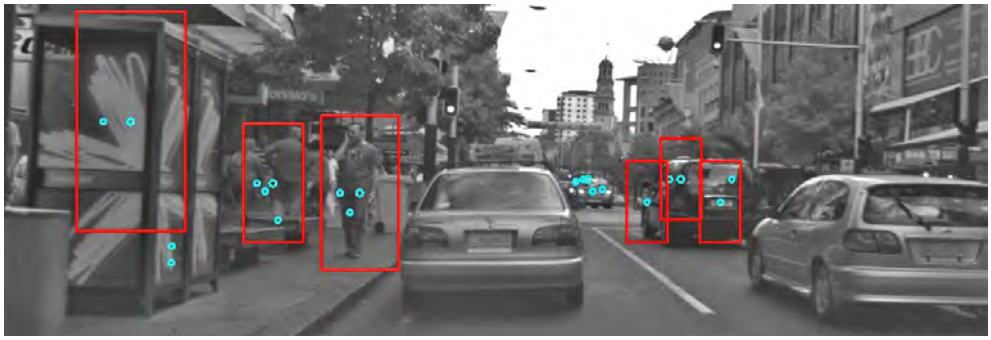

Figure 1.15 Red rectangles represent combined results; cyan circles are centres where the classifier detected a pedestrian. The few circles in the middle (at a car in a distance) are not merged. Courtesy of Junli Tao

strong classifier. For example, if the bounding box arrives at any leaf which has a probability greater than 0.5 for the class "pedestrian", then the box is classified this way (by this simple maximum-value rule). In case of overlapping bounding boxes, results may be merged into a single detection or box. See Figure 1.15.

Rejection of false-positives can be based on analysing a recorded image sequence, using failure in tracking or repeated detections for rejecting a detection.

Performance Evaluation of Pedestrian Detection Similar to the cases of traffic sign detection or face detection, it is common to use manually identified ground truth (i.e. available training or test data bases)[4] and measures such as PR, RC, MR, or FPPI (see the Appendix) for quantifying detection success.

1.4 Examples for Multi-Module Solutions

Higher-order tasks require the design of multi-module solutions. This section illustrates components of solutions (considered to be of higher order) for a few tasks, again paying particularly attention to tools for evaluating those solutions for real-world scenarios.

1.4.1 Video Segmentation

Segmentation for vehicle technology aims at *semantic segmentation* (Floros and Leibe 2012; Fröhlich et al. 2012) with temporal consistency along a recorded video sequence. The concept of super pixels (see, e.g. (Liu et al. 2012)) might be useful for achieving semantic segmentation. Temporal consistency requires tracking of segments and similarity calculations between tracked segments.

Use of Stereo Analysis and Optical Flow Calculations Modules for solving stereo matching and optical flow calculation can be used for designing a system for video

[4]For example, the *TUD Multiview Pedestrians* database is available at `www.d2.mpi-inf.mpg.de/node/428` for free download.

segmentation. For example, following (Hermann et al. 2011), stereo matching for images $L(.,.,t)$ and $R(.,.,t)$ of frame t results into a depth map which is segmented by

1. preprocessing for removing noisy (i.e. isolated) depth values and irrelevant depth values (e.g. in the sky region),
2. estimating a ground manifold using *v-disparities*; depth values identified as being in the ground manifold are also removed, and
3. performing a segmentation procedure (e.g. simple region growing) on the remaining depth values.

Resulting segments are likely to be of similar shape and location as those obtained for stereo frame $t+1$ by the same procedure. For each segment obtained for frame t, the mean optical flow vector for pixels in this segment defines the expected move of this segment into a new position in frame $t+1$. Those expected segments (of frame t after expected moves into frame $t+1$) are compared with the actual segments of frame $t+1$ for identifying correspondences, for example by applying a set-theoretical metric which represents the ratio between overlap and total area of both segments.

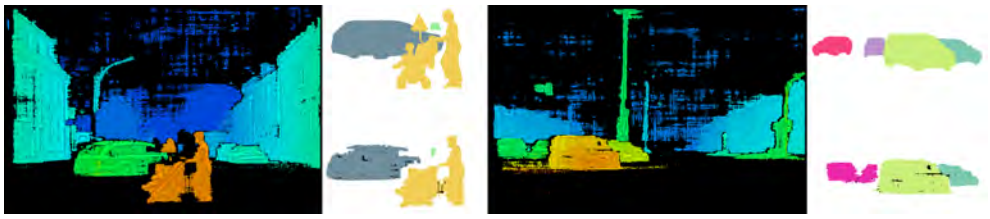

Figure 1.16 Two examples for Set 7 of EISATS illustrated by preprocessed depth maps following the described method (Steps 1 and 2). Ground truth for segments is provided by (Barth et al. 2010) and shown on top; resulting segments using the described method are shown below. Courtesy of Simon Hermann

Performance Evaluation of Video Segmentation There is a lack of provided ground truth for semantic segmentations in traffic sequences. Work reported in current publications on semantic segmentation, such as (Floros and Leibe 2012; Fröhlich et al. 2012), can be used for creating test databases.

(Barth et al. 2010) proposed a method for segmentation which is based on evaluating pixel probabilities whether they are in motion in the real world or not (using scene flow and ego-motion). (Barth et al. 2010) also provides ground truth for image segmentation in Set 7 of EISATS, illustrated by Fig. 1.16. This figure also shows resulting SGM stereo maps and segments obtained when following the multi-module approach briefly sketched above.

Modifications in the involved modules for stereo matching and optical flow calculation influence the final result. There might be dependencies between performances of contributing programs.

1.4.2 Monocular Wrong-lane Detection

Wrong-roadway related accidents lead only to a relatively small number of accidents, but with a high risk of being a heads-on crash. About 33.6 percent of the world population drives on the left-hand side.

Use of Digital Maps and GPS Lane-positioning algorithms based on e-maps and GPS (Bétaille et al. 2008; Peyret et al. 2008; Sekimoto et al. 2012; Selloum et al. 2010) are related to wrong-lane detection. A lane-positing module together with a lane-detection and tracking module allow us to design a wrong-lane detection system.

The location, provided by GPS, can be matched with an available e-map, for instance, an *openstreet map*. The number of lanes and lane direction(s) at the current location needs to be read from this map. Apart from using GPS and an e-map, further sensors such as onboard odometer or gyroscope can be used to refine the accuracy of ego-vehicle positioning on a road (Selloum et al. 2010).

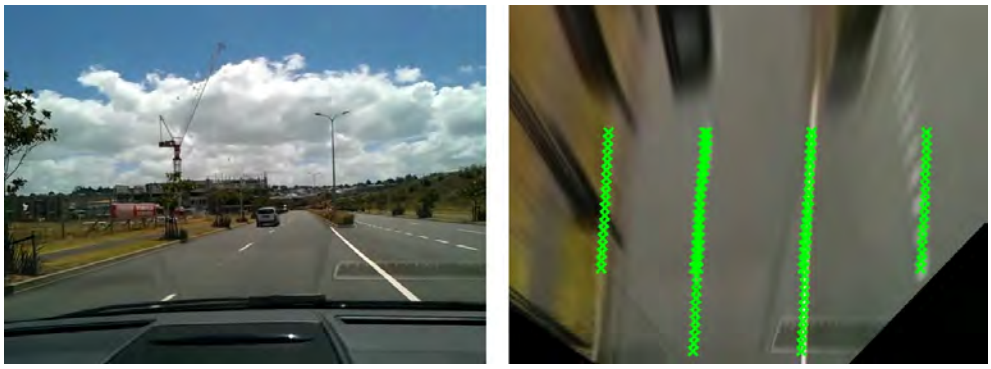

Figure 1.17 Three detected lanes in a country driving on the left-hand side. Matching with an available e-map simplifies the decision about the lane the ego-vehicle is currently driving in. Courtesy of Junli Tao

Single- and Multi-lane Detection Lane detection can be performed by using one camera only. Lane detection techniques typically only detect that lane the ego-vehicle is currently driving in, and not also adjacent lanes. Single-lane detection algorithms are discussed in (Bar Hillel et al. 2012; Shin *et al.* 2014).

A multi-lane-detection result (e.g. as shown in Fig. 1.17) is mapped onto a current lane configuration, thus supporting (besides a detection of the central marking in the middle of the road) the decision in which lane the ego-vehicle is currently driving in. Methods as described in (Deusch et al. 2012; Gupta et al. 2010; Zhao et al. 2012) address multi-lane detection. Lane confidence measures can be used to weight detected lanes for producing stable detections (Tao et al. 2013).

Wrong-lane Detection A first assistance system for the detection of driving on the wrong side of the road by reading no-entry signs of motorways is reported in (Daimler 2013),

and (Monteiro et al. 2007) analyzes motion patterns in highway traffic for understanding wrong-lane driving. (Tao et al. 2013) propose a system for wrong-lane driving detection by combining multi-lane detection results with e-map information.

Performance Evaluation of Wrong-lane Detection The performance of wrong-lane detection relies significantly on the accuracy of the used GPS. The camera-based wrong-lane detection methods can be measured by the number of false-positive lane detections per frame (FP), the number of false-negative lane detections per frame (FN), the number of false alarms (FA), and the number of *ignored frames* where only the driven lane is detected (IGN). There is a lack of video benchmark data with provided ground truth (in particular, generating realistic data while driving in the wrong lane is difficult to achieve).

1.4.3 Monocular Distance Detection under Challenging Conditions

Radar provides a standard non-computer-vision solution for measuring distance to obstacles, and stereo vision is the preferred choice in computer vision. Due to limitations to implement a stereo-vision sensor with sufficient base distance into a mobile device (such as mobile phone), monocular distance detection is also of practical relevance for driver assistance.

Two Examples of Techniques Used (Ali and Afghani 2005) propose to measure the shadow underneath a vehicle driving in front of the ego-vehicle, thus defining basically a single-module solution. (Rezaei and Terauchi 2012) suggest a data-fusion approach using a boosted classifier (based on global Haar-like features) (in conjunction with corner and line features, and virtual-symmetry of tail lights of vehicles) to effectively detect vehicles, with a particular focus on also covering challenging lighting conditions, as illustrated in Fig. 1.18. Global Haar-like features allowed the authors to deal successfully also with challenging lighting conditions; line and corner features helped to prevent false alarms; virtual-symmetry of tail-lights supported detections at close distance (Haar classifiers failed at very close distances, e.g. when a vehicle is so close that it covers 80% of image carrier Ω.

The proposed solution in (Rezaei and Terauchi 2012) uses the boosted classifier for an initial vehicle detection; for achieving more accurate results (i.e. less false-positives), these initial detections are also evaluated by applying line and edge features. A used rule is, for example, that "Vehicle regions have typically a higher density of corner points than road, sky, or other background regions". Corners are detected using the Shi-Tomasi detector (Shi and Tomasi 1994).

Figure 1.18 Monocular vehicle detection under challenging lighting conditions for subsequent distance estimation. Courtesy of Mahdi Rezaei

Performance Evaluation of Distance Detection Radar or stereo vision can be used for generating ground truth for distances between ego-vehicle and vehicles in front of the ego-vehicle. There is also a lack of online benchmark video data with provided ground-truth.

1.4.4 3D Modelling of Road Sides

The Section 1.2.2 mentioned under the subject of environment reconstruction already the task to model road sides based on video data recorded in a road vehicle; see also Fig. 1.7. Road side modelling is an example of *ground-level reconstruction*, as an alternative to *aerial reconstruction* using cameras in airplanes or multi-copters. For the latter technology, see, for example, (3D Reality Maps 2014).

Examples of Related Work Ground-level reconstruction is a subject, for example, in (Geiger et al. 2011; Huang and Klette 2012; Xiao et al. 2009), all using multiple cameras while recording road sides in a *single run* when moving the cameras in the ego-vehicle basically parallel to the road borders into one direction only (i.e. without any significant variations in the path).

A Single-Run Reconstruction (Zeng and Klette 2013) propose a stereo-based reconstruction framework for automatically merging reconstruction results from multiple single runs in different directions. For each direction the authors suggest to perform

1. stereo matching for having disparity maps for subsequent stereo frames $I(.,.,t)$, allowing to generate a cloud of 3D points on visible surfaces in the camera coordinate system at time t,
2. bundle-adjustment-based (Sünderhauf et al. 2005) visual odometry (Nister et al. 2004) based on a chosen feature detector such as SIFT (Lowe 2004), SURF (Bay et al. 2006), or ORB (Rublee et al. 2011) with good repeatability properties (i.e. features visible in frames $I(.,.,t)$ and $I(.,.,t+1)$ should be identifiable as being corresponding),
3. a map of the cloud of 3D points generated at time t into an underlying world coordinate system using the estimated changes in camera poses, and
4. a triangulation of the merged sets of 3D points in the underlying world coordinate system using an α-shape algorithm, as proposed in (Edelsbrunner and Mücke 1994), to generate a surface model.

The inclusion of 3D points obtained at time t in Step 3 into the combined set of points can be controlled by results of the third-eye technology (Morales and Klette 2009); for example, NCC values below a defined threshold may exclude the insertion totally.

Repeatability of Detected Features Drift in visual odometry (Jiang et al. 2011) often leads to a twist in the 3D model. Repeatability of detected features is critical for achieving accuracy in visual odometry. See (Song and Klette 2013) for a performance evaluation of repeatability for feature detectors offered in 2013 in `OpenCV`, including SIFT, SURF, and ORB.

Bundle Adjustment (Triggs et al. 2000) define *bundle adjustment* by refining the 3D model as well as detecting camera parameters. A set of n 3D points b_i is seen from m cameras (e.g. a camera at m different times while recording a video sequence). The cameras have

parameters a_j. Let X_{ij} be the projection of the ith point on camera j. By bundle adjustment we minimize the reprojection error with respect to 3D points b_i and camera parameters a_j. This is a non-linear minimization problem; it can be solved by using iterative methods such as Levenberg-Marquardt.

Memory and time define limitations for applying an optimization strategy to all the recorded frames of one video sequence at once; instead we need to limit ourself to processing only a subsequence of m frames $I(.,.,t)$ to $I(.,.,t+m)$ at a time.

A Multiple-Run Reconstruction Consider a special *registration problem* defined by incorporating a triangulated surface, being a result of a single-run reconstruction, into the already-given triangulated surface of the same road segment. Both triangulations use one underlying world coordinate system each. The main steps of a straightforward procedure can be as follows:

1. Identify a set of interest points (e.g. 3D keypoints, see (Salti et al. 2011)) that represent both 3D point sets.
2. Compute a feature descriptor at each interest point.
3. Estimate the correspondence between both sets of feature points, based on their descriptor similarities (e.g. using RANSAC).
4. Reject an estimated percentage of outliers based on accepting that data are noisy.
5. The identified correspondence mapping defines a transformation between both underlying world coordinate systems.
6. Use this transform as an initial estimate; refine by using an *iterative closest points technique* (ICP); see (Besl and McKay 1992).

Because such an approach is unlikely to succeed for video sequences (e.g. due to inaccuracy of visual odometry results for each single run), (Zeng and Klette 2013) suggest a semi-automatic procedure for multi-run roadside reconstructions, starting with manually selected key points for an initial alignment, breaking the point clouds or surface models into segments (e.g. representing 10 m of roadside only) and process such segments individually (following the procedure above), and refining the model by some post-processing steps, such as aiming at a uniform density of surface points or enforcing surface patch planarity where the generated data "suggest" to do so.

Figure 1.19 illustrates results for two runs through the scene illustrated in Fig. 1.7 (top-down views), the merged data set (also in top-down projection), and finally a 3D view of the merged set.

Performance Evaluation of 3D Modelling of Road Sides Laser range-finders can be used for generating fairly accurate roadside models; those could be used as ground truth for the discussed vision-based multi-run roadside reconstruction problem. There are no known benchmark data available online. It would also be necessary to provide multiple calibrated stereo video sequences for the modelled road scene, ideally recorded under different lighting conditions, with minor changes in the scene (e.g. smaller objects appear or disappear), and so forth.

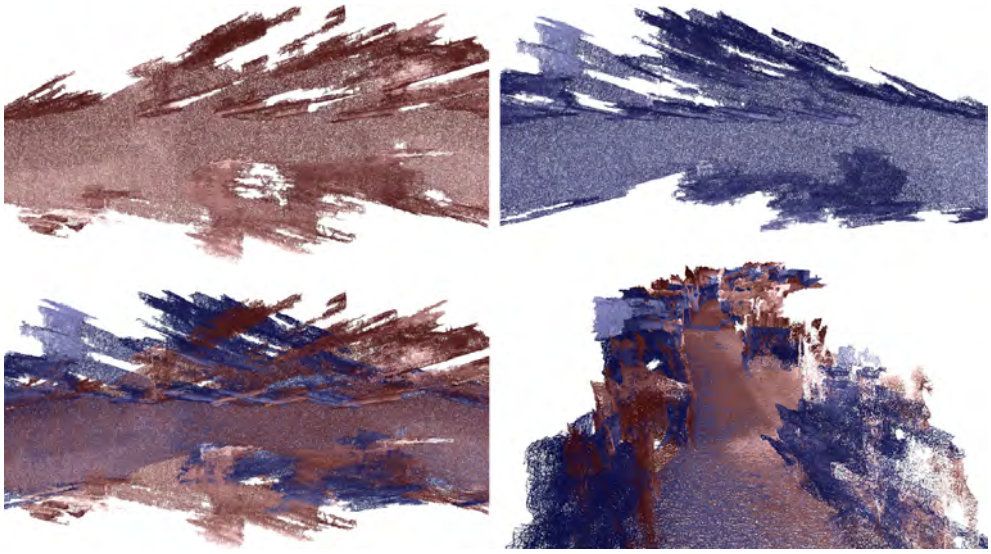

Figure 1.19 Two opposite runs through the scene illustrated by Fig. 1.7. *Top*: Top-down projections of recovered surface points, clearly illustrating a directional bias of occlusions. *Bottom, left*: Merged data also in a top-down view. *Bottom, right*: 3D view of the merged data. Courtesy of Yi Zeng

1.5 Concluding Remarks

The vehicle industry world-wide has assigned major research and development resources for offering competitive solutions for vision-based components for vehicles. Research at academic institutions needs to address future or fundamental tasks, challenges which are not of immediate interest for the vehicle industry, for being able to continue to contribute to this area.

The chapter introduced basic notation, selected tasks, and selected solutions. It reviewed work in the field of computer vision in vehicles. It pointed to existing open questions in this area, often related to

1. adding further alternatives to only a few existing robust solutions for one module,
2. a comparative evaluation of such solutions,
3. ways of analysing benchmarks for their particular challenges,
4. the design of more complex (i.e. multi-module) systems, and
5. ways how to test such multi-module systems.

Specifying and solving a higher-order task might be a good strategy to define fundamental research, ahead of currently extremely intense industrial research and development within the area of computer vision for vehicles. (The borderline between lower- and higher-order is not well defined; aiming at robustness, also including challenging scenarios, or understanding interactions in dynamic scenes between multiple moving objects, are certainly examples of ways to arrive at higher-order tasks.)

Computer vision can help to solve true problems in society or industry, thus contributing to the prevention of social harms or atrocities; it is a fundamental ethical obligation of researchers in this field not to contribute to those (e.g. by designing computer-vision solutions for the use in UAVs for killing people). Academics identify *ethics in research* often with subjects such as plagiarism, competence, or objectivity, and a main principle is also social responsibility. Computer vision in road vehicles can play, for example, a major role in reducing casualties in traffic accidents which are counted by hundreds of thousands of people worldwide each year; it is a very satisfying task for a researcher to contribute to improved road safety.

Acknowledgement

The author thanks Simon Hermann, Antonio Lopez Pena, Mahdi Rezaei, Konstantin Schauwecker, Junli Tao, and Garry Tee for comments on drafts of this report.

1.6 References

3D Reality Maps 2014 `www.realitymaps.de/en/`.

Ali A, Afghani S 2005 Shadow based on-road vehicle detection and verification using Haar wavelet packet transform. In *Proc. IEEE Int. Conf. Information Communication Technologies* pp. 346–350.

Bar Hillel A, Lerner R, Levi D, and Raz G 2012 Recent progress in road and lane detection: a survey *Machine Vision Applications* **23**, published online.

Barnes N and Zelinsky A 2004 Real-time radial symmetry for speed sign detection. In *Proc. IEEE Intelligent Vehicles Symposium* pp. 566–571.

Barth A, Siegmund J, Meissner A, Franke U, and Förstner W 2010 Probabilistic multi-class scene flow segmentation for traffic scenes. In *Proc. DAGM*, LNCS 6376, pp. 513–522.

Bay H, Tuytelaars T, Van Gool L 2006 SURF: Speeded up robust features. In *Proc. ECCV* pp. 404–417.

Besl PJ, McKay ND 1992 A method for registration of 3-D shapes *IEEE Trans. Pattern Analysis Machine Intelligence* **14**, 239–256.

Bétaille D, Toledo-Moreo R, and Laneurit J 2008 Making an enhanced map for lane location based services. In *Proc. IEEE Conf. Intelligent Transportation Systems* pp. 711–716.

Borkar A, Hayes M and Smith MT 2010 An efficient method to generate ground truth for evaluating lane detection systems. In *Proc. IEEE Int. Conf. Acoustics Speech Signal Processing* pp. 1090–1093.

Breiman L 2001 Random forests *Machine Learning* **45** 5–32.

Briassouli A and Kompatsiaris I 2011 Change detection for temporal texture in the Fourier domain. In *Proc. ACCV* LNCS 6492, pp. 149–160.

Brox T, Bruhn A, Papenberg N, and Weickert J 2004 High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV* LNCS 3024, pp. 25–36.

Camera & Imaging Products Association 2009 Multi-picture format (online translation of CIPA DC-007-2009).

Choi KH, Park SY, Kim SH, Lee KS, Park JH, Cho SI, and Park JH 2010 Methods to detect road features for video-based in-vehicle navigation systems *J. Intelligent Transportation Systems: Technology Planning Operations* **14**, 13–26.

Crowley JL and Sanderson AC 1987 Multiple resolution representation and probabilistic matching of 2-D gray-scale shape *IEEE Trans. Pattern Analysis Machine Intelligence* **9**, 113 – 121.

Daimler A.G. 2013 First assistance system in the fight against vehicles driving on the wrong side of the road. www.daimler.com/dccom/, January 21.

Dalal N and Triggs B 2005 Histograms of oriented gradients for human detection. In *Proc. CVPR* pp. 886–893.

Deusch H, Wiest J, Reuter S, Szczot M, Konrad M, Dietmayer K 2012 A random finite set approach to multiple lane detection. In *Proc. Intelligent Transportation Systems* pp. 270–275.

Edelsbrunner H, Mücke EP 1994 Three-dimensional alpha shapes. *ACM Trans. Graphics* **13**, 43–72.

Escalera S, Baró X, Pujol O, Vitrià J and Radeva P 2011 *Traffic-Sign Recognition Systems* Springer.

Floros G and Leibe B 2012 Joint 2D-3D temporally consistent semantic segmentation of street scenes. In *Proc. CVPR* pp. 2823–2830.

Fröhlich B, Rodner E, and Denzler J 2012 Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In *Proc. ACCV* LNCS 7724, pp. 218–231.

Geiger A, Ziegler J, Stiller C 2011 StereoScan: Dense 3d reconstruction in real-time. In *Proc. IEEE Intelligent Vehicles* pp. 963–968.

Gómez GD 2010 A global approach to vision-based pedestrian detection for advanced driver assistance systems. PhD thesis, Univ. Autónoma de Barcelona.

Guan S, Klette R 2008 Belief-propagation on edge images for stereo analysis of image sequences. In *Proc. Robot Vision* LNCS 4931, pp. 291–302.

Gupta RA, Snyder W, Pitts WS 2010 Concurrent visual multiple lane detection for autonomous vehicles. In *Proc. IEEE Conf. Robotics Automation* pp. 2416–2422

Haeusler R and Klette R 2010 Benchmarking stereo data (not the matching algorithms). In *Proc. DAGM* LNCS 6376, pp. 383–392.

Haeusler R and Klette R 2012 Analysis of KITTI data for stereo analysis with stereo confidence measures. In *Proc. ECCV, Workshops and Demonstrations* LNCS 7584, pp. 158–167

Harris C and and Stephens M 1988 A combined corner and edge detector. In *Proc. Alvey Vision Conference* pp. 147–151.

Hermann S, Boerner A, and Klette R 2011 Mid-level segmentation and segment tracking for long-range stereo analysis. In *Proc. PSIVT* LNCS 7087, pp. 224–235.

Hermann S and Klette R 2009 The naked truth about cost functions for stereo matching. MItech-TR-33, The University of Auckland.

Hermann S and Klette R 2012 Iterative semi-global matching for robust driver assistance systems. In *Proc. ACCV*, LNCS 7726, pp. 465–478

Hermann S and Werner R 2013 High accuracy optical flow for 3D medical image registration using the census cost function. In *Proc. PSIVT*, LNCS 8333, pp. 24 – 36.

Hirschmüller H 2005 Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proc. CVPR*, volume 2, pp. 807–814.

Hirschmüller H and Scharstein D 2009 Evaluation of stereo matching costs on images with radiometric differences *IEEE Trans. Pattern Analysis Machine Intelligence* **31**, 1582–1599.

Horn BKP and Schunck BG 1981 Determining optic flow *Artificial Intelligence* **17**, 185–203

Huang F, Klette R 2012 City-scale modeling towards street navigation applications *J. Information Convergence Communication Engineering* **10**, 411–419.

Jiang R, Klette R, Wang S 2011 Statistical modeling of long-range drift in visual odometry. In *Proc. ACCV workshop* LNCS 6469, pp. 214–224.

Khan W, Suaste V, Caudillo D, and Klette R 2013 Belief propagation stereo matching compared to

iSGM on binocular or trinocular video data. In *Proc. IEEE Intelligent Vehicles Symposium*, pp. 791–796.

Kim Z 2008 Robust lane detection and tracking in challenging scenarios *IEEE Trans. Intelligent Transportation Systems* **9**, 16–26.

Klette R 2014 *Concise Computer Vision: An Introduction into Theory and Algorithms* Springer.

Kovesi PD 1993 A dimensionless measure of edge significance from phase congruency calculated via wavelets. In *Proc. New Zealand Conf. Image Vision Computing*, pp. 87–94.

Lindeberg T 1994 *Scale-Space Theory in Computer Vision* Kluwer Academic Publishers.

Liu L, Xing J, Ai H, and Lao S 2012 Semantic superpixel based vehicle tracking. In *Proc. ICPR* pp. 2222–2225.

Lowe DG 2004 Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision* **60**, 91–110.

Marr D and Hildreth E 1980 Theory of edge detection *Proc. Royal Society London, Series B, Biological Sciences* **207**, 187 – 217.

McCall JC and Trivedi MM 2006 Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation *IEEE Trans. Intelligent Transportation Systems* **7**, 20–37.

Monteiro G, Ribeiro M, Marcos J, and Batista J 2007 Wrongway drivers detection based on optical flow. In *Proc. IEEE Int. Conf. Image Processing* volume 5, pp. 141–144.

Morales S and Klette R 2009 A third eye for performance evaluation in stereo sequence analysis. In *Proc. CAIP*, LNCS 5702, pp. 1078–1086.

Müller-Schneiders S, Nunn C and Meuter M 2008 Performance evaluation of a real time traffic sign recognition system. In *Proc. IEEE Intelligent Vehicles Symposium* pp. 79–84.

Murphy-Chutorian E and Trivedi MM 2009 Head pose estimation in computer vision: A survey *IEEE Trans. Pattern Analysis Machine Intelligence* **31**, 607–626.

Nister D, Naroditsky O, Bergen J, Visual odometry. In *Proc. CVPR* volume 1, pp. 652–659.

Peyret F, Laneurit J, and Bétaille D 2008 A novel system using enhanced digital maps and WAAS for a lane-level positioning. In *Proc. World Congress Intelligent Transport Systems* 12 pages.

Rezaei M and Klette R 2011 3D cascade of classifiers for open and closed eye detection in driver distraction monitoring. In *Proc CAIP* LNCS 6855, pp. 171–179.

Rezaei M and Klette R 2013 Novel adaptive eye detection and tracking for challenging lighting conditions. In *Proc. ACCV Workshops* LNCS 7729, pp. 427–440.

Rezaei M and Terauchi M 2013 Vehicle detection based on multi-feature clues and Dempster-Shafer fusion theory. In *Proc. PSIVT* LNCS 8333, pp. 60–72

Rublee E, Rabaud V, Konolige K, Bradski G 2011 ORB: An efficient alternative to SIFT or SURF. In *Proc. ICCV* pp. 2564–2571.

Salti S, Tombari F, Di Stefano L 2011 A performance evaluation of 3D keypoint detectors. In *Proc. 3DIMPVT* pp. 236–243.

Sekimoto Y, Matsubayashi Y, Yamada H, Imai R, Usui T, and Kanasugi H 2012 Light weight lane positioning of vehicles using a smartphone GPS by monitoring the distance from the center line. In *Proc. IEEE Conf. Intelligent Transportation Systems* pp. 1561–1565.

Selloum A, Bétaille D, Le Carpentier E, and Peyret F 2010 Robustification of a map aided location process using road direction. In *Proc. IEEE Conf. Intelligent Transportation Systems* pp. 1504–1510.

Shi J, Tomasi C 1994 Good features to track. In *Proc. CVPR* pp. 593–600.

Shin B, Xu Z and Klette R 2014 Visual lane analysis and higher-order tasks: A concise review *Machine Vision Applications* to be published.

Smith K, Gatica-Perez D, Odobez JM and Ba S 2005 Evaluating multi-object tracking. In *Proc. CVPR Workshop EEMCV*.

Sobel IE (1970) Camera models and machine perception Stanford, Stanford Univ. Press, pp. 277–284.

Song Z, Klette R 2013 Robustness of point feature detection. In *Proc. CAIP* pp. 580 – 588.

Suaste V, Caudillo D, Shin B, and Klette R 2013 Third-eye stereo analysis evaluation enhanced by data measures. In *Proc. MCPR* LNCS 7914, pp. 74–83.

Sünderhauf N, Konolige K, Lacroix S, Protzel P 2005 Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. In *Proc. Autonome Mobile Systems* pp. 157–163.

Sun J, Zheng NN and Shum HY 2003 Stereo matching using belief propagation *IEEE Trans. Pattern Analysis Machine Intelligence* **25** pp. 1–14

Szeliski R 1999 Prediction error as a quality metric for motion and stereo. In *Proc. ICCV* pp. 781–788.

Tao J, Shin B, Klette R 2013 Wrong roadway detection for multi-lane roads. In *Proc. CAIP* LNCS 8048, pp. 50–58.

Triggs B, McLauchlan P, Hartley R, Fitzgibbon A 2000 Bundle adjustment - A modern synthesis. In *Proc. Vision Algorithms Theory Practice* pp. 298–375.

Viola P and Jones M 2001 Robust real-time face detection *Int. J. Computer Vision* **57**, 137–154.

WHO 2013 Global status report on road safety (online).

Xiao J, Fang T, Zhao P, Lhuilier M and Quan L 2009 Image-based street-side city modeling. In *Proc. SIGGRAPH* article 114.

Zeng Y and Klette R 2013 Multi-run 3D streetside reconstruction from a vehicle. In *Proc. CAIP* LNCS 8047, pp. 580–588.

Zhao K, Meuter M, Nunn C, Muller D, Muller-Schneiders S, Pauli J 2012 A novel multi-lane detection and tracking system. In *Proc. IEEE Intelligent Vehicles Symposium* pp. 1084–1089

Zhao L and Thorpe C 2000 Stereo and neural network-based pedestrian detection *IEEE Trans. Int. Transportation Systems* **1**, 148–154.

Appendix: Notation and Basic Definitions

Basic notations and definitions follow (Klette 2014).

Images and Videos An *image* I is defined on a set

$$\Omega = \{(x, y) : 1 \leq x \leq N_{cols} \wedge 1 \leq y \leq N_{rows}\} \subset \mathbb{Z}^2 \tag{1.7}$$

of pairs of integers (*pixel locations*), called the image *carrier*, where N_{cols} and N_{rows} define the number of columns and rows, respectively. We assume a left-hand coordinate system with the coordinate origin in the upper-left corner of the image, the x-axis to the right, and the y-axis downward. A *pixel* of an image I combines a location $p = (x, y)$ in the carrier Ω with the value $I(p)$ of I at this location.

A *scalar image* I takes values in a set $\{0, 1, \ldots, 2^a - 1\}$; typically with $a = 8$, $a = 12$, or $a = 16$. A *vector-valued image* I has scalar values in a finite number of channels or bands. A video or image sequence consists of *frames* $I(., ., t)$, for $t = 1, 2, \ldots, T$, all being images on the same carrier Ω.

Three Examples In case of an RGB *colour image* $I = (R, G, B)$, we have pixels $(p, I(p)) = (p, R(p), G(p), B(p))$. A geometrically rectified grey-level stereo image $I = (L, R)$ consists of two channels L and R, usually called *left* and *right* image; this is implemented in the multi-picture object (`mpo`) format for images (CIPA 2009). For a sequence of grey-level stereo images, we have pixel $(p, t, L(p, t), R(p, t))$ in frame t, which is the combined representation of pixels $(p, t, L(p, t))$ and $(p, t, R(p, t))$ in $L(., ., t)$ and $R(., ., t)$, respectively, at pixel location p and time t.

Gauss Function The zero-mean *Gauss function* is defined as follows:

$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{1.8}$$

A convolution of an image I with the Gauss function produces smoothed images

$$L(p, \sigma) = [I \star G_\sigma](p) \tag{1.9}$$

also known as *Gaussians*, for $\sigma > 0$. (We stay with symbol L here as introduced by (Lindeberg 1994) for "layer"; a given context will prevent confusion with the left image L of a stereo pair.)

Edges *Step-edges* in images are detected based on first- or second order derivatives, such as, respectively, on values of the *gradient* (Sobel 1970) or *Laplacian* (Marr and Hildreth 1980) given by

$$\nabla I = \mathbf{grad}\, I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]^\top \quad \text{or} \quad \triangle I = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{1.10}$$

Local maxima of L_1- or L_2-magnitudes $||\nabla I||_1$ or $||\nabla I||_2$, or zero-crossings of values $\triangle I$ are taken as an indication for a step-edge. The gradient or Laplacian is commonly preceded by smoothing, using a convolution with the zero-mean Gauss function.

Alternatively, *Phase-congruency edges* in images are detected based on local frequency-space representations (Kovesi 1993).

Corners *Corners* in images are localized based on high curvature of intensity values, to be identified by two large eigenvalues of the *Hessian matrix*

$$\mathbf{H}(p) = \begin{bmatrix} I_{xx}(p) & I_{xy}(p) \\ I_{xy}(p) & I_{yy}(p) \end{bmatrix} \tag{1.11}$$

at a pixel location p in a scalar image I; see (Harris and Stephens 1988). See Fig. 1.1 for corners detected by FAST. Corner detection is often preceded by smoothing using a convolution with the zero-mean Gauss function.

Scale Space and Key Points *Key points* or *interest points* are commonly detected as maxima or minima in a $3 \times 3 \times 3$ subset of the *scale space* of a given image (Crowley and Sanderson 1987; Lindeberg 1994). A finite set of differences of Gaussians

$$D_{\sigma,a}(p) = L(p, \sigma) - L(p, a\sigma) \tag{1.12}$$

produces a *DoG scale space*. These differences are approximations to Laplacians of increasingly smoothed versions of an image; see Fig. 1.11 for an example of such Laplacians forming an *LoG scale space*.

Features An *image feature* is finally a *location*, defined by a key point, edge, corner and so forth, together with a *descriptor*, usually given as a data vector (e.g. in case of SIFT of length 128 representing local gradients), but possibly also in other formats such as a graph. For example, the descriptor of a step-edge can be mean and variance of gradient values along the edge, and the descriptor of a corner can be defined by the eigenvalues of the Hessian matrix.

World and Camera Coordinates We have an $X_wY_wZ_w$ *world coordinate system* which is not defined by a particular camera or other sensor, and a *camera coordinate system* $X_sY_sZ_s$ (index "s" for "sensor") which is described with respect to the chosen world coordinates by means of an affine transform, defined by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{T}$. In 3D we assume right-hand coordinate systems.

A point in 3D space is given as $P_w = (X_w, Y_w, Z_w)$ in world coordinates, or as $P_s = (X_s, Y_s, Z_s)$ in camera coordinates. Besides this *coordinate notation* for points we also use *vector notation*, such as $P_w = [X_w, Y_w, Z_w]^T$ for point P_w.

Pinhole-type Camera The Z_s-axis models the *optical axis*. Assuming an ideal pinhole-type camera, we can ignore radial distortion and have *undistorted projected points* in the image plane with coordinates x_u and y_u. The distance f between the x_uy_u image plane and the projection centre is the *focal length*.

A visible point $P = (X_s, Y_s, Z_s)$ in the world is mapped by *central projection* into pixel location $p = (x_u, y_u)$ in the undistorted image plane:

$$x_u = \frac{fX_s}{Z_s} \quad \text{and} \quad y_u = \frac{fY_s}{Z_s} \tag{1.13}$$

with the origin of x_uy_u image coordinates at the intersection point of the Z_s axis with the image plane.

The intersection point (c_x, c_y) of the optical axis with the image plane in xy coordinates is called the *principal point*. It follows that $(x, y) = (x_u + c_x, y_u + c_y)$ A pixel location (x, y) in the 2D xy image coordinate system has 3D coordinates $(x - c_x, y - c_y, f)$ in the $X_sY_sZ_s$ camera coordinate system.

Intrinsic and Extrinsic Parametres Assuming multiple cameras C_i, for some indices i (e.g. just C_L and C_R for binocular stereo), camera calibration specifies intrinsic parameters such as edge lengths e_x^i and e_y^i of camera sensor cells (defining the aspect ratio), a skew parameter s^i, coordinates of the principal point $\mathbf{c}^i = (c_x^i, c_y^i)$ where optic axis of Camera i and image plane intersect, the focal length f^i, possibly refined as f_x^i and f_y^i, and lens distortion parameters starting with κ_1^i and κ_2^i. In general it can be assumed that lens distortion has been calibrated before and does not need to be included anymore in the set of intrinsic parameters. Extrinsic parameters are defined by rotation matrices and translation vectors, e.g. matrix $\mathbf{R}^{ij}$ and vector $\mathbf{t}^{ij}$ for the affine transform between camera coordinate systems $X_s^i Y_s^i Z_s^i$ and $X_s^j Y_s^j Z_s^j$, or matrix $\mathbf{R}^i$ and vector $\mathbf{t}^i$ for the affine transform between camera coordinate system $X_s^i Y_s^i Z_s^i$ and $X_w Y_w Z_w$.

Single-camera Projection Equation The camera-projection equation in homogeneous coordinates, mapping a 3D point $P = (X_w, Y_w, Z_w)$ into image coordinates $p^i = (x^i, y^i)$ of the ith camera, is as follows:

$$k \begin{bmatrix} x^i \\ y^i \\ 1 \end{bmatrix} = \begin{bmatrix} f^i/e_x^i & s^i & c_x^i & 0 \\ 0 & f^i/e_y^i & c_y^i & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^i & -[\mathbf{R}^i]^\top \mathbf{t}^i \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1.14}$$

$$= [\mathbf{K}^i | \mathbf{0}] \cdot \mathbf{A}^i \cdot [X_w, Y_w, Z_w, 1]^\top \tag{1.15}$$

where $k \neq 0$ is a scaling factor. This defines a 3×3 matrix $\mathbf{K}^i$ of intrinsic camera parameters, and a 4×4 matrix $\mathbf{A}^i$ of extrinsic parameters (of the affine transform) of camera i. The 3×4 *camera matrix* $\mathbf{C}^i = [\mathbf{K}^i | \mathbf{0}] \cdot \mathbf{A}^i$ is defined by 11 parameters if we allow for an arbitrary scaling of parameters; otherwise it is 12.

Markov Random-Field Model In an abstract sense we assign to each pixel a *label* l (e.g. an optical flow vector $\mathbf{u}$ or a disparity d) out of a set L of possible labels (e.g. all vectors pointing from a pixel p to points in a Euclidean distance to p of less than a given threshold). Labels $(u, v) \in \mathbb{R}^2$ are thus in the 2D continuous plane.[5] Labels are assigned to all the pixels in the carrier Ω by a *labelling function* $f : \Omega \to L$. Solving a labelling problem means to identify a labelling f which approximates somehow an optimum of a defined *error* or *energy*

$$E_{total}(f) = E_{data}(f) + \lambda \cdot E_{smooth}(f) \tag{1.16}$$

where $\lambda > 0$ is a weight. Here, $E_{data}(f)$ is the *data-cost term* and $E_{smooth}(f)$ the *smoothness-cost term*. In general, we take $\lambda < 1$ for avoiding strong smoothing. We search for an optimal (i.e. of minimal total error) f in the set of al possible labellings, which defines a *total variation* (TV).

We detail Equ. (1.16) by adding costs at pixels. In the current image, label $f_p = f(p)$ is assigned by labelling f to pixel position p. Then we have that

$$E_{total}(f) = \sum_{p \in \Omega} E_{data}(p, f_p) \; + \; \lambda \cdot \sum_{p \in \Omega} \sum_{q \in A(p)} E_{smooth}(f_p, f_q) \tag{1.17}$$

[5] A. A. Markov (1856 – 1922) studied stochastic processes where the interaction of multiple random variables can be modelled by an undirected graph; this graph is here defined by interactions between adjacent pixels in Ω for specifying a local smoothness condition. The labelling f specifies the random variables.

where A is an adjacency relation between pixel locations. In optical flow or stereo vision, label f_p (i.e. optical-flow vector or disparity) defines a pixel q in another image; in this case we can also write $E_{data}(p,q)$ instead of $E_{data}(p,f_p)$.

Census Data-Cost Term The census cost function has been identified as being able to compensate successfully brightness variations in input images of a recorded video (Hermann and Klette 2009; Hirschmüller and Scharstein 2009). The *zero-mean normalized census-cost function* is defined by comparing a $(2l+1)\times(2k+1)$ window centred at pixel location p in frame $I(.,.,t)$ with a window of the same size centred at a pixel location q in frame $I(.,.,t+1)$. Let $\overline{I}_{p,t}$ be the mean of the window around p, and $\overline{I}_{q,t+1}$ be the mean around q. Then we have that

$$E_{ZCEN}(p,q)=\sum_{i=-l}^{l}\sum_{j=-k}^{k}\rho_{ij} \tag{1.18}$$

with

$$\rho_{ij}=\begin{cases}0 & I(p+(i,j),t)\perp\overline{I}_{p,t}\ \wedge\ I(q+(i,j),t+1)\perp\overline{I}_{q,t+1}\\ 1 & \text{otherwise}\end{cases} \tag{1.19}$$

with $\perp$ either $<$ or $>$ in both cases. Let $\mathbf{a}_p$ be the vector listing results $\mathrm{sgn}(I(p+(i,j),t)-\overline{I}_{p,t})$ in a left-to-right, top-to-bottom order, where sgn is the signum function; $\mathbf{b}_q$ lists values $\mathrm{sgn}(I(q+(i,j),t+1)-\overline{I}_{q,t+1})$. The zero-mean normalized census data-cost $E_{ZCEN}(p,q)$ equals the Hamming distance between vectors $\mathbf{a}_p$ and $\mathbf{b}_q$.

Object Detector and Performance Measures In general, an *object detector* is defined by applying a classifier for an object detection problem. We assume that any decision made can be evaluated as being either correct or false. Let *tp* or *fp* denote the numbers of true-positives or false-positives, respectively. Analogously we define *tn* and *fn* for the negatives; *tn* is not a common entry for performance measures.

Precision PR is the ratio of true-positives compared to all detections. *Recall* RC (or *sensitivity*) is the ratio of true-positives compared to all potentially possible detections (i.e. to the number of all visible objects):

$$\mathrm{PR}=\frac{tp}{tp+fp}\qquad\text{and}\qquad\mathrm{RC}=\frac{tp}{tp+fn} \tag{1.20}$$

The *miss rate* MR is the ratio of false-negatives compared to all objects in an image. *False-positives per image* FPPI is the ratio of false-positives compared to all detected objects:

$$\mathrm{MR}=\frac{fn}{tp+fn}\qquad\text{and}\qquad\mathrm{FPPI}=\frac{fp}{tp+fp} \tag{1.21}$$

How to decide whether a detected object is true-positive? Assume that objects in images have been locally identified manually by bounding boxes, serving as the ground truth. All detected objects are matched with these *ground-truth boxes* by calculating ratios of areas of overlapping regions

$$a_o=\frac{\mathcal{A}(D\cap T)}{\mathcal{A}(D\cup T)} \tag{1.22}$$

where $\mathcal{A}$ denotes the area of a region in an image, D is the detected bounding box of the object, and T is the area of the bounding box of the matched ground-truth box. If a_o is larger than a threshold T, say $T=0.5$, the detected object is taken as a true-positive.

INDEX