Computational Imaging and Vision 45

Mahdi Rezaei · Reinhard Klette

Computer Vision for Driver Assistance

Simultaneous Traffic and Driver Monitoring

Springer

Computational Imaging and Vision

Computational Imaging and Vision

This comprehensive book series embraces state-of-the-art expository works and advanced research monographs on any aspect of this interdisciplinary field.

Topics covered by the series fall in the following four main categories:

- Imaging Systems and Image Processing
- Computer Vision and Image Understanding
- Visualization
- Applications of Imaging Technologies

Only monographs or multi-authored books that have a distinct subject area, that is where each chapter has been invited in order to fulfill this purpose, will be considered for the series.

Volume 45

More information about this series at http://www.springer.com/series/5754

Mahdi Rezaei • Reinhard Klette

Computer Vision for Driver Assistance

Simultaneous Traffic and Driver Monitoring

Springer

Mahdi Rezaei
Department of Computer Engineering
Qazvin Islamic Azad University
Qazvin, Iran

Reinhard Klette
Department of Electrical and Electronic Engineering
Auckland University of Technology
Auckland, New Zealand

ISSN 1381-6446
Computational Imaging and Vision
ISBN 978-3-319-50549-7 ISBN 978-3-319-50551-0 (eBook)
DOI 10.1007/978-3-319-50551-0

Library of Congress Control Number: 2016962187

Mathematics Subject Classification (2010): 68Txx, 68T45, 65Dxx, 65D19

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Currently, the automotive industry is experiencing three substantial changes of a revolutionary character, the replacement of internal combustion engines by electric motors, the integration of vehicle–vehicle and vehicle–infrastructure communication, and the rise of autonomous driving. The latter is still at a stage where drivers are expected to be in control of the vehicle at all times, but provided automated control features of the vehicle already enhance safety and driver comfort. We prefer to speak about *driver assistance* when addressing current developments towards autonomous driving. Driver-assistance features of vehicles are essentially based on data provided by various sensors such as radar, LIDAR, ultrasound, GPS, inertial measurement unit (IMU), or cameras. In this book, we discuss the use of cameras for driver assistance.

Computer vision-based driver assistance is an emerging technology, in both the automotive industry and academia. Despite the existence of some commercial safety systems such as night vision, adaptive cruise control, and lane departure warning systems, we are at the beginning of a long research pathway towards a future generation of intelligent vehicles.

Challenging lighting conditions in real-world driving scenarios, simultaneous monitoring of *driver vigilance* and *road hazards*, ensuring that the system responds in *real time*, and the legal requirements to comply with a high degree of *accuracy* are the main concerns for the developers of any advanced driver-assistance system (ADAS).

This book reviews relevant studies in the past decade as well as the state of the art in the field. The book also proposes various computer vision algorithms, techniques, and methodologies to address the aforementioned challenges, which are mainly suitable to be implemented for monocular vision.

Mobile devices (such as a mobile phone or a small-sized computing device) are often equipped with one or multiple cameras. If they also come with a stereocamera option, then this is typically with a small baseline (i.e., the distance between the two cameras), which supports stereo visualization, but not accurate stereo image analysis. Purpose-built processing units for driver monitoring or traffic observation, to be an add-on to already existing cars, also benefit from solutions for monocular

vision. However, our focus on monocular vision does not exclude the potential integration of the provided solutions into a multi-sensor environment, possibly also collecting data by using stereo vision, a laser range-finder, radar, ultrasound, GPS, an inertial measurement unit, or other sensors.

The discussion of monocular vision is also of academic interest, providing answers to the question of which tasks can be solved sufficiently by "using one eye only", which defines an important research subject on its own.

The first part of the book focuses on monitoring driver vigilance, including classification, detection, and tracking of the driver's *facial features*, i.e., eye status, head pose, yawning detection, and head nodding. The second part of the book mainly contributes to methods for road perception and road hazard monitoring, by introducing novel algorithms for *vehicle detection* and *distance estimation*. In the third part of the book, we simultaneously analyse the driver's attention (in-cabin data) and road hazards (out-road data). We apply a *data fusion* approach on both data sources to measure the overall risk of driving conditions, to prevent or mitigate imminent crashes, and to assist a distracted driver in a timely and efficient manner.

For each part of our discussion, we present and analyse real-world experimental results, supported by benchmarks on a comprehensive range of datasets.

The book consists of eight chapters. Each chapter of the book addresses a major goal. All chapters follow a unique structure starting with an introduction and an overview of related work, followed by the proposed technique or algorithm(s). Then we compare the proposed methods against common existing techniques and the state of the art, followed by discussions on novelties and improvements based on empirical, analytical, or experimental results. The outline of the book is as follows:

Chapter 1: Vision-Based Driver-Assistance Systems

This chapter provides a general overview of tasks addressed when using cameras in modern cars.

Chapter 2: Driver–Environment Understanding

This chapter refines the general goals described in Chap. 1 for the specific context of interactions between driver monitoring and road environment; the chapter reviews related work published elsewhere.

Chapter 3: Computer Vision Basics

We briefly describe common computer vision concepts (theoretical and mathematical) which are of relevance for the following chapters in the book. This chapter includes a review on image notations, including integral images, colour conversion, line detection (Hough space), camera coordinates, and stereo vision analysis.

Chapter 4: Object Detection, Classification, and Tracking

This chapter provides the basics of classification and machine learning. It starts with a general introduction to object detection and then discusses supervised and unsupervised classification techniques. It ends with a brief outline of object tracking. This is material which is also of relevance for the following chapters.

Chapter 5: Driver Drowsiness Detection

We analyse the strengths and weaknesses of common face detectors such as LBP- and Haar-like-based detectors and their suitability for a DAS application. We discuss the causes of face detection failure in the context of driving conditions, followed

by our solutions to improve the results. The most important part of this chapter is the introduction of an *adaptive global* and *dynamic global* Haar-like classifier as a significant improvement on the standard Viola–Jones detector in both "training" and "application" phases. We also provide a Kalman filter-based tracking solution that can successfully track the driver's drowsiness status, especially for open and closed eye detection at night, under sharp street lights, strong shades, strong backlights, and partial occlusions.

Chapter 6: Driver Inattention Detection

Following the initial facial features detected in the previous chapter, we continue by developing robust techniques in order to detect the driver's direction of attention. The main aim of this chapter is to achieve a 6D head-pose estimation including roll, yaw, pitch, and (x, y) position of the head, at the time t. Having an accurate pose detection, as well as the eye-status detection discussed in Chap. 5, we are able to assess the driver's level of fatigue or distraction.

Chapter 7: Vehicle Detection and Distance Estimation

In this chapter, we introduce an AGHaar-based vehicle detector augmented with multi-clue feature analysis and a Dempster–Shafer fusion. The chapter provides sufficient experiments to assure the robustness of the vehicle detection algorithm under different weather and challenging conditions.

Chapter 8: Fuzzy Fusion for Collision Avoidance

We can now define a risk level for every moment of driving, based on the online data acquired from Chaps. 5, 6, and 7 (i.e., driver's state, detected vehicles on the road, and distance and angle of the ego-vehicle to the lead vehicles).

Within a fuzzy fusion platform, we correlate the in-vehicle data with the out-road hazards and identify the risk level of a potential crash driving condition based on the eight input parameters: yaw, roll, pitch (the driver's direction of attention), eye status (open or closed), yawning, head nodding, distance to the detected vehicles in front of the ego-vehicle, and angle of the detected vehicle to the ego-vehicle.

The book briefly discusses stereo vision for driver assistance, but focuses in general on the analysis of monocular video data. We do not cover all the tasks which are possibly solvable by monocular vision only. For example, there is also traffic sign recognition, lane detection, and motion analysis (e.g., optic flow calculation), which are also areas experiencing substantial progress using monocular vision only. However, by discussing driver monitoring and examples of traffic monitoring, we present the required components that illustrate the important novel contribution of merging the analysis of both in-cabin data and out-road data.

The authors acknowledge the cooperation of colleagues, visitors, and students of the *Environment Perception and Driver Assistance* (.enpeda..) project, originally at the University of Auckland and now at Auckland University of Technology. We would especially like to thank Zahra Moayed and Noor Saleem for providing the text about the KLT tracker.

Qazvin, Iran
Auckland, New Zealand
September 2016

Mahdi Rezaei
Reinhard Klette

Contents

Symbols

General Notation	Description
$\mathbb{N}$	Set of natural numbers
$\mathbb{Q}$	Set of rational numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integers
I	Image in spatial domain
N_{cols}	Image width in pixels
N_{rows}	Image height in pixels
x, y	Pixel coordinates (x, y) in a 2D image $\mathbb{R}^2$
u	Pixel intensity value
Ω	Image carrier, a set of all $N_{cols} \times N_{rows}$ pixel locations
X, Y, Z	Coordinates in a 3D space $\mathbb{R}^3$
p, q	Points in $\mathbb{R}^2$, with coordinates x and y
P, Q, R	Points in $\mathbb{R}^3$, with coordinates X, Y, and Z
W, W_p	Window in an image, windows with reference pixel location p
a, b, c	Real numbers
a, b, c	Vectors
A, B, C	Matrices
$\underset{\sim}{\mathrm{A}}, \underset{\sim}{\mathrm{B}}, \underset{\sim}{\mathrm{C}}$	Fuzzy sets
α, β, γ	Angles
$\mathscr{A}(\cdot)$	Area of a measurable set (function)
$\angle$	Angle sign
θ	Angular parameter in Hough space
ψ	Standard Haar-like feature
Φ	Global Haar-like feature
f	Camera focal length
h	Camera height from the ground
Θ	Camera viewing angle
H	Hue

S	Saturation
V	Value (in HSV colour space)
t	Time
$\mathbf{t}$	Translation vector
τ	Threshold
$\mathbf{R}$	Rotation matrix
ℓ	Length
ρ	Distance to origin in Hough space
ζ	Parity
$\wedge$	Logical AND
$\vee$	Logical OR
$\cap$	Intersections of sets
$\cup$	Union of sets
$\times$	Times (multiplication)
$\oplus$	Orthogonal sum
$\otimes$	Data fusion
$\subset$	Subset
$\approx$	Approximation
$\in$	Belonging to a set
ω	Training weights
δ	Saturation angle (in HSV colour space)
Γ	Coefficient factor
ε	Very small real greater than zero
Δ	Time or value difference between two variables
∇	Vector differential operator (e.g., for calculating the image gradient)
Π	Product
Σ	Summation

Chapter 1
Vision-Based Driver-Assistance Systems

1.1 Driver-Assistance Towards Autonomous Driving

Driver-assistance systems (DASs) still assume a driver in the *ego-vehicle*, which is the reference vehicle, opposed to the "other" vehicles in a traffic context. Fully autonomous driving, expected to dominate driving in a few years from now, will benefit incrementally from driver-assistance solutions.

Why Driver Assistance Systems? According to the World Health Organization (WHO) around 1.24 million people die each year due to traffic accidents [283] (this is on average about 2.4 people *every minute*), in addition to 50 million serious injuries. Road injury ranked number 9 in 2011 for causes of death in the world, by far ahead of all war-related deaths, for example.

In recent years, traffic accidents have annually caused a global cost of about US$500 billion which takes 1–3% of the gross domestic product from all countries in the world. The report [283] also anticipated that road crashes would rank 3rd for deaths and injuries by the year 2020, with an increased rate to 1.9 million fatalities annually.

Current *passive* and *active* safety systems [286] can reduce the impact of traffic accidents. Passive safety systems, such as air-bags, seat belts, padded dashboards, or physical structures of a vehicle, normally help to reduce the severity or the consequences of an accident. Active safety systems like adaptive cruise control (ACC), automatic braking systems (ABS), or lane departure warning systems (LDWS) are designed to prevent or decrease the chance of crash occurrences.

DASs are designed to reduce the number of traffic accidents (i.e. more safety for the driver, but also for all the passengers in the ego-vehicle, and also for any other participant in on-road traffic).

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_1

Improved comfort is the second reason for designing a DAS. For example, stop-and-go driving at slow speed can be replaced by autonomous driving, or the unevenness of the road ahead can be detected and compensated by the car [10]. Subsystems such as ACC can reduce the driver's stress for long-distance driving or on high-speed motorways [167].

1.2 Sensors

A comprehensive DAS collects information with various sensors and decides whether the current moment of driving could possibly lead to a potential accident.

VB-DASs A *vision-based DAS* (VB-DAS) uses cameras as sensors. Historically, VB-DASs started with solutions for lane-departure or blind spot supervision; for example, see [170, 243] for related comments and references. VB-DASs are now a common feature in modern vehicles; for example, see [10, 244]. A backward-camera, possibly also two cameras integrated into the side mirrors, connected to a monitor in the dash board, are relatively simple but very efficient additions to modern cars for improving safety and comfort.

Multi-sensor Data Collection Besides cameras, DASs (often also called ADASs, for *advanced* DAS) use also further sensors such as GPS, IMU (= inertial moment unit), radar, sound, or LIDAR (laser range-finders); see [15]. *Adaptive cruise control* (ACC) pioneered these approaches in the early 1990s: longitudinal distance is measured by a radar unit (e.g. behind the front grille or under the bumper), and more recently also by employing a laser range-finder or stereo vision [109]. Figure 1.1 shows a sketch for multi-sensor data collection in a car.

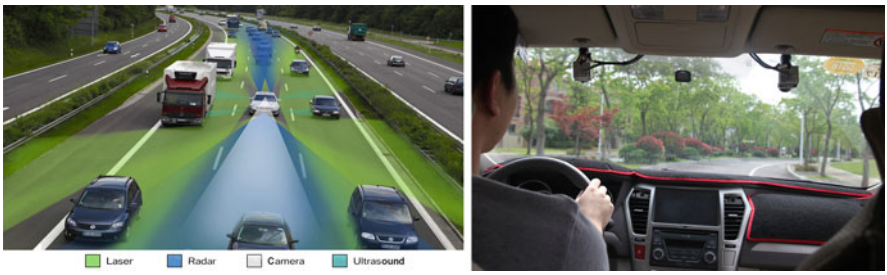

Fig. 1.1 *Left*: Graphical sketch for multi-sensor data recording (Courtesy of Clemens Dannheim). *Right*: Experimental set-up of two cameras in a DongFeng test vehicle at Wuhan University (Courtesy of Li Ming)

Data Collection via Sensors A DAS can be in communication with external wireless sensor networks (WSNs) [31], satellites, or global positioning systems (GPSs) to help the driver with alternative routes and online localization information. An ADAS can also communicate or "talk" to other vehicles via vehicle-to-vehicle (V2V) [60, 266] or vehicle to infrastructure (V2X or V2I) [129] systems. Vehicles equipped with these systems can exchange data with nearby vehicles or receive supplementary information regarding traffic jams, just-happened accidents, or other road hazards within a diameter of about 5 km [180].

Common sensors used in DASs and intelligent transportation systems (ITSs) are ultrasonic sensors [100], visions sensors or cameras [241], radar [288], LIDAR and laser range-finders [138, 228]. The sensors collect data from the road environment and supply them to a DAS as single or multiple sources of information [74].

Ultrasonic Sensors versus LIDAR Ultrasonic sensors are normally useful for detecting obstacles at short distances, e.g. to assist the driver in parallel parking or auto-park systems. LIDAR technology involves ranging and scanning of the emitted laser pulses to measure the distance to obstacles based on the concept of time-of-flight. Depending on the resolution of the laser scanner, such systems can produce a pin-point and accurate topographic map from the road scene. LIDARs can be used for short- and long-distance obstacle detection or adaptive cruise control. The performance of a LIDAR system may be degraded due to light reflections or weather conditions.

Radar versus LIDAR Radar sensors emit radio waves and analyze the bounced wave via a receiver. These sensors can detect the distance and the speed of moving objects by measuring the change in the frequency of returned signals caused by the Doppler effect [288]. Since the wavelengths of radar sensors are much longer than those of LIDARs, they can be used for faster scanning (to analyze the overall status of the road). LIDARs can provide higher resolution results.

Cameras versus LIDAR Camera sensors are cheaper than both radar and LIDAR sensors. They generally need or support more advanced post-processing (image processing, image classification, object detection, and so forth) in order to convert raw perceived images into meaningful information or interpretations of complex traffic scenes.

Data Fusion Each of the above-mentioned sensors has its strengths and limitations. Hence, researchers take multi-sensor *data fusion* approaches into account. This leads to more reliable solutions for DAsS, but also to more expensive hardware, and higher computational costs. Figure 1.2 illustrates a multi-sensor platform and applications of the discussed sensors.

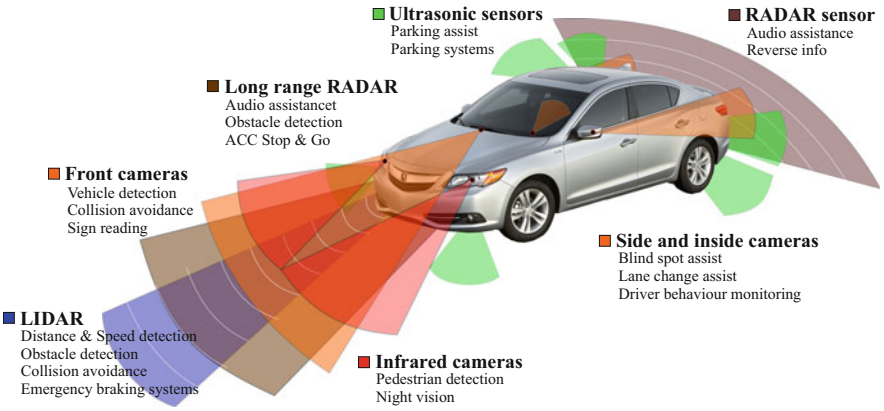

Fig. 1.2 Common types of sensors for ADAS

1.3 Vision-Based Driver Assistance

Vision-based driver-assistance systems (VB-DASs) define an important component of DASs. Computer vision addresses in general the processing and analysis of image and video data recorded in the real world. In our case, image or video data are recorded in the ego-vehicle. Traffic monitoring also involves surveillance cameras at road intersections, pedestrian crossings, and so forth. We do not discuss surveillance in this book, only computer vision tasks in relation to recordings in the ego-vehicle.

VB-DASs combine one or multiple cameras, a processing unit with implemented applications, possibly interfaces to further sensors available in the vehicle, or to vehicle components related to vehicle control (e.g. if the system detects an obstacle on the left of the vehicle then steering to the left can be blocked) or to vehicle-driver communication (e.g. using the windshield as a head-up display, a visual signal may point to a potential risk).

Computer Vision This book focuses on tasks for computer vision related to driver assistance (*How does one solve automatically visual perception tasks for driver assistance?*); it does not cover non-camera sensors or interfaces with other components in the ego-vehicle.

Computer vision [119] often classifies approaches into *low-level vision* (e.g. image processing, stereo vision, or optic flow), *medium-level vision* (e.g. semantic segmentation of images, object detection, object tracking, or object clustering), and *high-level vision* (e.g. the complex understanding of perceived scenes, such as current and possible future interactions between various objects in a scene).

This classification does not correspond to the complexity of the studied problems; for example, optic flow calculation (a low-level vision approach towards motion analysis) is in general more challenging than lane analysis for situations involving well-marked lanes and reasonable lighting conditions, and often also more challeng-

ing than the detection and recognition of a particular traffic sign (e.g. a stop sign or speed-limit sign).

Proofs of Existence The visual system of human beings provides a proof of existence that vision alone can deliver nearly all of the information required for moving around safely in the 3-dimensional (3D) world [36].

Visual odometry supported by our "built-in IMU" (i.e. accelerometer and gyroscope, with related sensors in the ears) defines human navigation. Thus, in principle, multiple cameras and an IMU are potentially sufficient for solving navigation tasks in the real world.

The Mars rovers "Curiosity" and "Opportunity" operate based on computer vision; "Opportunity" has done so since 2004.

Cameras and Frames Cameras in the ego-vehicle record frames at different time slots t, typically at 25 Hz or higher frequency. We speak about *frame t* if these image data are recorded at time $t \cdot \triangle t$. Formally, frame t is denoted by $I(.,.,t)$, assuming a single camera for recording (i.e. the *monocular* case). The camera records grey-level or colour values $I(x, y, t)$ at a pixel position (x, y) at time slot t.

A frame, recorded at time slot t, can also be a time-synchronized stereo-pair of images. In such a *binocular* case (i.e. like human stereo vision), we have two video streams; a frame is then composed of two images $L(.,.,t)$ and $R(.,.,t)$ for a left and a right channel, formally $I(.,.,t) = (L(.,.,t), R(.,.,t))$.

Outward-recorded (or *looking-out*) frames should have a large bit depth (e.g. at least 10 bits per pixel in each recorded channel), a high dynamic range (i.e. to be able to deal with "sudden" changes of light intensities between frames, or within one frame) and a high pixel resolution (e.g. significantly larger than just 640×480 VGA image resolution) in order to support accurate vision algorithms considering a wide horizontal angle (a wide vertical angle is less important).

Inward-recorded (or *looking-in*) frames for monitoring the driver, or the ego-vehicle occupants in general [260], might be of lower pixel resolution than for outward-recording.

Outward-recording should aim at covering as much as possible of the full 360° panorama around a vehicle. To do so, multiple time-synchronized cameras are installed "around" the ego-vehicle, for example stereo cameras looking forward or backward. This extends binocular vision to *multi-ocular* vision, and a recorded frame at time t is then composed of multiple images.

Performance Requirements and Evaluation Ideally, VB-DASs have to operate in any occurring scenario, whether sunshine, rain in the night, driving in a tunnel or on serpentines, inner-city or highway traffic, and so forth. While there are crash tests for evaluating the physical components of a vehicle; to evaluate VB-DAS solutions it is necessary to run implemented applications on a large diversity of possibly occurring scenarios. Test data (i.e. a *video data benchmark*) can be recorded in the real world, or generated by computer-graphics programs (e.g. for simulating particular changes in processed frames).

The evaluation of solutions in relation to particular real-world scenarios has been discussed in [120]. Solutions can be characterized as being *accurate* or *robust*. *Accuracy* means correctness for a given scenario. *Robustness* means "sufficient" correctness for a set of scenarios, which may also include cases of challenging scenarios. Ideally, robustness should address *any* possible scenario in the real world for a given task.

Accuracy (or robustness) may be expressed by a single number, for example by a real number in the interval [0, 1], where 1 means "ideal accuracy" (e.g. identified by a perfect match with ground truth data), and 0 indicates a measurement that is outside a pre-defined margin of error compared to ground truth values. *Performance* of a DAS may be quantitatively defined based on common classification rates (e.g. using numbers of true detections or false detections; to be discussed later in detail).

Used benchmarks should be of high diversity and complexity; used video data need to be evaluated to understand their complexity. For example, *changes* in recorded video data can be characterized by using quantitative measures such as video descriptors [25] or data measures [236]. Sets of benchmark data should represent hours or days of driving in a wide diversity of possible scenarios.

Currently there are only very limited sets of data publicly available for comparative VB-DAS evaluations. Figure 1.3 illustrates two possible ways of generating benchmarks, one by using computer graphics for rendering sequences with accurately-known ground truth, as done for one data set on [56], and a second way by using high-end sensors (in the illustrated case for [118]; approximate depth ground truth is provided by the use of a laser range-finder).

Adaptive Solutions We cannot expect to have all-time "winners" when comparatively evaluating computer vision solutions for VB-DAS applications. Vehicles operate in the real world, which is so diverse that not all of the possible event occurrences can be modelled within the underlying constraints of 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, possibly defining an optimization strategy for designing VB-DASs which is adaptive to the current scenario.

Fig. 1.3 Examples of benchmark data available for a comparative analysis of computer vision algorithms in a traffic context. *Left*: Image of a real-world sequence provided on KITTI (with approximate ground truth about distances on [118]). *Right*: Image of a synthetic image sequence provided on EISATS (including accurate ground truth about distances and movements)

Premier Journals and Major Conferences Research contributions in VB-DASs typically appear in premier journals such as *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Intelligent Transportation Systems Magazine*, *IEEE Transactions on Vehicular Systems*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, and *Computer Vision and Image Understanding*. Major conferences with contributions on VB-DAS are the annual *IEEE Intelligent Vehicles Symposium*, and the annual *IEEE Intelligent Transportation Systems Conference*.

1.4 Safety and Comfort Functionalities

Before discussing computer vision tasks, we briefly point to functionalities where recorded video data or graphical visualisations are simply used to enhance the driver's visual perception of the environment, for safety or driver comfort.

Avoidance of Blind Spots The *blind spot* is the total area around the ego-vehicle which cannot be seen by the driver. This is typically composed of an area behind the vehicle and two areas on the left and right of the vehicle. A simple VB-DAS solution is to show video data of those areas to the driver on a screen at relevant times. The law in the USA requests that vehicles built from May 1, 2018 onward need to have a back-up camera [39].

Night Vision A VB-DAS may also support a driver's visual perception in the night or during otherwise limited viewing conditions (i.e. rain, snow, or fog), thus increasing the seeing distance and improving object recognition. This is typically implemented by showing improved video data on a screen, but this can also be achieved by using a head-up display. Fog detection (for driver warning), see [190], is an example for distinguishing weather conditions.

The automotive industry designs active (i.e. use of a near-infrared light source built into the vehicle, which is invisible to the human eye but visible to a standard digital camera) or passive (i.e. no special illumination of the scene but capturing thermal radiation) recording systems to provide enhanced images for the driver.

Virtual Windshield The head-up display, or *virtual windshield*, is an efficient way of representing information to the driver without creating a need to change the head pose (see Fig. 1.4). Indoor-recording with face detection may be used for an accurate understanding of the driver's head pose.

The virtual windshield may be used, for example, to provide information about speed, distance to destination, or navigation data. No computer vision needs to be involved in these cases.

The virtual windshield may also be used to provide information about currently applying traffic signs (e.g. speed limit), to give an enhanced view of lane borders in the night, or to show a flashing light indicating the location of a potential hazard (e.g. a detected pedestrian in low-light conditions), or to label visible buildings (e.g. hotel chain).

Fig. 1.4 Two examples of a virtual windshield in a BMW. The "7" in the *image on the left* identifies the current gear (Courtesy of Clemens Dannheim)

For the mentioned examples, the shown information is mostly derived from particular VB-DAS applications (e.g. for traffic sign detection and recognition, visual lane-border analysis, or an advanced traffic hazard detection system; see related material later in this chapter).

Intelligent Headlamp Control Adaptive LED headlamps adjust the individual light beams according to the visible traffic scene [222]. Each light beam (of each addressable LED emitter) may vary between a low-aimed low beam and a high-aimed high beam. The beam is adjusted to maximize the seeing range for the driver in the ego-vehicle, but with the constraint to avoid dazzling drivers in other vehicles or pedestrians.

Cameras in the ego-vehicle are used for vehicle and pedestrian detection (for these topics, see later in this chapter), continually changing a *glare-free high beam* pattern of the headlamps.

1.5 VB-DAS Examples

A traffic scene is composed of the road (with lane markings, pedestrian crossings, speed bumps, cavities, and so forth), *road furniture* (e.g. traffic signs, handrails, or construction site blocks), various types of obstacles (driving vehicles, pedestrians, rocks, parked vehicles, and so forth), and also by traffic-related buildings, tunnels, bridges, and so forth.

A complex scene analysis needs to understand the current traffic-related components, their motion, and their possible near-future interaction with the ego-vehicle, or even the possible impacts on other traffic participants. Before discussing traffic scene-analysis tasks for cameras which point outwards from the ego-vehicle, we first consider a few tasks when pointing a camera towards the driver (to monitor driver awareness).

Driver Monitoring Cameras are not the only way for driver awareness monitoring. For example, see [108] for a tactile solution using an embedded sensor in the

steering wheel. Cameras are not only useful for understanding the state of the driver (e.g. drowsiness detection) but, in particular, they are also appropriate for analyzing the viewing direction.

Face and eye detection [270], or head-pose analysis [174], are basic tasks in this area. The viewing direction can be estimated via head pose analysis; eye gaze analysis [251] is an alternative way to estimate viewing direction which also covers eye state analysis (i.e. a percentage estimate of being open or closed). Challenging lighting conditions still present unsatisfactorily-solved scenarios; for example, see [210] for such scenarios. Foot gesture or visual hand motion patterns are further possible indicators for driver monitoring [182, 258].

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 [211] should also correspond (for some time) to those outside regions where safety-related events occur. Here, head pose or face detection is typically followed by eye detection and an analysis of the state of the eyes or eye-gaze detection. See Fig. 1.5.

In the applied face or eye detection technique [270], a search window scans through the current input image comparing intensities with *local Haar wavelets*; see Fig. 1.6, right, for examples of such wavelets. Rezaei and Klette [210] also introduces *global Haar wavelets* (to be described later in this book), motivated by challenging lighting conditions (as illustrated in Fig. 1.5). Figure 1.6 also illustrates the use of a head model for identifying eye regions.

For a more detailed introduction into driver monitoring, see Sect. 2.2.

Fig. 1.5 Face detection, eye detection, and face tracking results under challenging lighting conditions

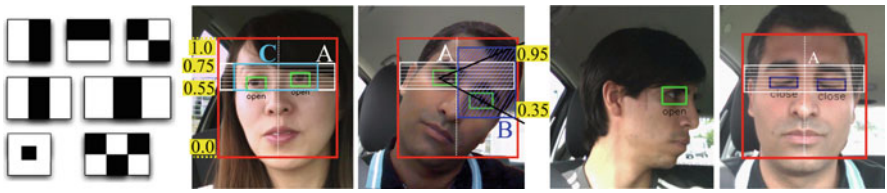

Fig. 1.6 *Left*: Examples of local Haar wavelets. *Other images*: Detected eyes in a (purported) driver's face based on defining a region of interest within a detected face for expected eye locations

Speed Adaptation Now we mention a first task for outward-recording cameras. *Intelligent speed adaptation* (ISA) can be based on knowing the currently-applying speed limit, road conditions, the distance to the vehicle in front, and further traffic-related events (e.g. children playing close to the road often require a speed reduction).

Basic information for ISA is available in e-maps via GPS. Vision technologies are adequate for collecting on-site information. The detection and interpretation of speed-signs [55] or road-markings [114] are examples of traffic-sign analysis, and the evaluation of road conditions is part of road environment analysis (see the related paragraphs below).

Platoons, Queuing An *automated queue assistant* (AQuA) applies in congested traffic situations on highways; see Volvo's program for AQuA design in [80]. For example, it is of interest for a truck convoy to maintain constant speed and distances between trucks. But it is also of importance in any congested traffic situation.

An AQuA application should ideally combine longitudinal distance control (to the preceding vehicle) for adjusting speed and lateral control for steering supervision (i.e. distance to side vehicles). Driver monitoring (see the paragraph above) is also of significance for understanding driver awareness (drowsiness or inattentiveness). Lane detection and analysis (e.g. of lane curvature; see the paragraph above) is of importance for proper positioning control of the vehicle.

For example, a truck convoy may be grouped for automated driving into a *platoon*, with the goal of reducing inter-truck distances to increase the capacity of roads.

Parking Automated parking requires in general a wider field of view than blind spot surveillance, and a full 360° recording supports both applications [274, 304].

Autonomous parking systems typically use multiple sensors such as ultrasonic or close-range radars, or laser scanners (implemented in the front and rear bumpers of the car), and vision sensors. The system needs to detect a possible parking space, and needs to guide the vehicle autonomously into this space. Note that bumper-implemented sensors have a limited field of view (i.e. close to the ground).

Automated parking (as designed, tested, and used in Asia and Europe in the past 20 years) has now been extended to solutions for parking autonomously in a more general parking house, with the drop-off and collection point at the entrance of the parking house. Vision sensors play an essential role in such an application.

Blind Spot Supervision At the beginning of the chapter we already mentioned blind spot visualization. More advanced applications analyze the video data recorded for blind spots, and communicate only the necessary information (e.g. there is another vehicle on the left) to the driver [201]. For example, the *Blind Spot Information System* (BLIS), introduced by Volvo in 2005, produced just an alert if another vehicle was detected to the left or right of the ego-vehicle (by analyzing a recorded video); see [253].

Lane Departure Warning Lane analysis (as discussed above) aims in particular to provide information about lane changes, if a driver is interested in this type of support (e.g. truck or long-distance bus drivers).

Wrong Lane Detection Wrong-roadway related accidents lead only to a relatively small number of accidents, but with a high risk of being a head-on crash. About 33.6% of the world's population drives on the left-hand side.

Lane-positioning algorithms based on e-maps and GPS [17, 193, 231, 232] are related to wrong-lane detection; this *map matching* approach is not yet accurate due to existing variance in (standard) GPS data. 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 an onboard odometer or gyroscope can be used to refine the accuracy of ego-vehicle positioning on a road [232].

A multi-lane-detection result (e.g. as shown in Fig. 1.7) 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 [47, 82, 314] address multi-lane detection. Lane confidence measures can be used to weight detected lanes to produce stable detections [250].

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 [40], and [166] analyzes motion patterns in highway traffic for understanding wrong-lane driving. Tao et al. [250] proposes a system for wrong-lane driving detection by combining multi-lane detection results with e-map information.

Sections 2.3 and 2.4 report in more detail a few subjects concerning environment analysis.

Fig. 1.7 Three detected lanes in a country driving on the *left-hand side*. Matching with an available e-map simplifies the decision about which lane the ego-vehicle is currently driving in (Courtesy of Junli Tao)

1.6 Current Developments

Vision technologies, in combination with further technological developments, offer various new opportunities to improve traffic safety and comfort.

Active Safety Systems Above we briefly reviewed subject areas of various active safety systems and algorithms. Computer-vision techniques [92, 308], in particular stereo-vision-based techniques [268], contributed to the current fast progress towards autonomous driving. Accuracy issues, as still apparent in the 2012 "Robust Vision Challenge" [87], are currently solved for an increasing number of challenging situations. LIDAR [202, 281] can provide accurate range information (costs go down with increases in numbers of produced systems); overcoming the sparse data collection still appears to be the critical issue in this field. The fusion of multiple sensors such as radar and vision [74, 86, 160] combines the strength of individual sensors.

Monocular vision-based solutions are a strategy-of-choice if a fusion of stereo vision, LIDAR, or radar is not possible or not cost-effective; for example, for solutions using smart phones or other mobile devices [317].

Among computer vision-based methodologies, current research addresses issues such as vehicle detection based on analyzing shade underneath a vehicle [3, 84], stereo vision to estimate distances between ego-vehicle and obstacles [257], optical flow-based methods to detect moving objects and vehicles [32], the use of *local binary patterns* (LBP) [184, 203], or Haar-like features [79, 171, 299].

Particle and Kalman Filters Time sequences of image data (video) support the use of temporal filters. Applications of various types of Kalman or particle filters are today of common use when designing VB-DASs. For example, stereo-vision and particle-filter-supported tracking is used by Danescu and Nedevschi [42] for lane tracking. They compare a linear Kalman filter (KF) with particle filtering with the conclusion of a better performance when applying particle filtering, especially in the case of road discontinuity, lane forking, lane joining, or sharp curves, where the linear KF may fail or diverge. But the particle-filter-based method also underperforms in terms of accuracy in dense city road zones. Discussions of iconic, extended, or unscented KFs are common practice when analyzing video data.

Three Revolutionary Processes In the year 2016, we have experienced three essentially inter-related revolutionary developments in the automotive industry:

- Electric vehicles (EVs) are replacing internal combustion engine vehicles,
- autonomous driving is incrementally replacing human drivers; driver assistance (in particular VB-DAS) is providing practical solutions during the transition phase,
- vehicle to vehicle (V2V), vehicle to infrastructure (V2I), and vehicle to roadside (V2R) communication is developing into a common technology.

We finish this chapter by briefly mentioning a few developments in this context, just for illustration, without aiming to cover these complex developments in any detail.

Driver-Environment Understanding The sections above discussed the understanding of the driver (i.e. inward-recording for analyzing awareness, eye gaze, and so forth), and also various modules of a VB-DAS for outward-recording. The obvious next step is to correlate driver understanding with traffic scene understanding, for example to warn about a detected issue for which it appears that the driver has not yet paid attention [80, 211]. Two chapters of this book are especially dedicated to the subject of understanding a driver in a given traffic scenario.

For example, the virtual windshield (i.e. a head-up display) appears to be a good implementation for issuing a warning to a driver if a particular issue is detected outside in traffic, to be indicated in relation to the head pose of the driver.

Approaches for VB-DASs which combine looking-in and looking-out techniques provide significant benefits for realizing active safety functionalities; see [51, 157, 181, 252, 261] and Fig. 1.8.

Inter-Car Communication Inter-car communication (a non-vision technology) supports the better understanding of large-scale road environments, not apprehendable from the perspective of a single car. This approach is also known as *car-to-car* (C2C). It is expected that these communication networks will contain vehicles in a local neighbourhood but also roadside sources as nodes (C2I = *car-to-infrastructure*).

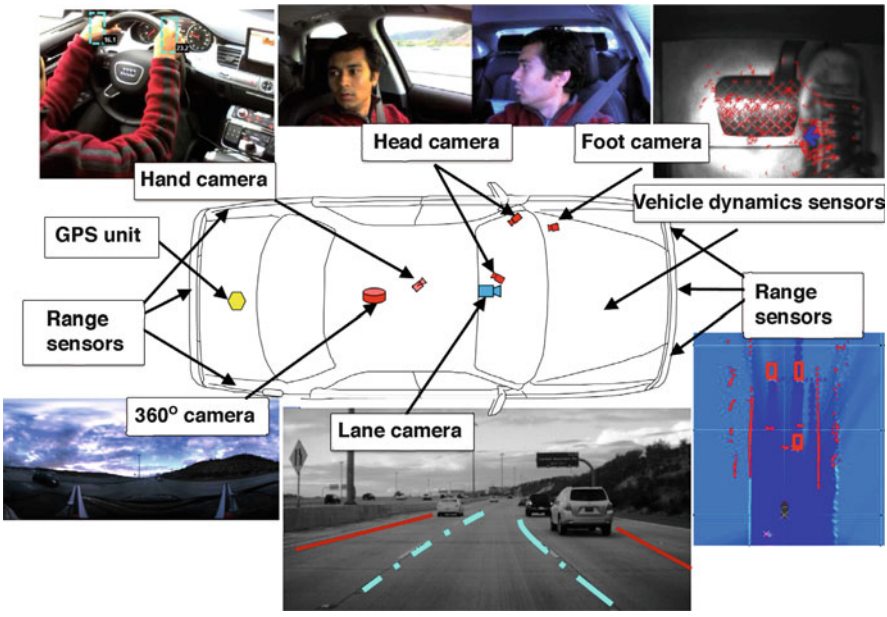

Fig. 1.8 Holistic scene understanding and driver monitoring using multiple sensors for inward and outward recording. Driver awareness can be modelled based on a combined analysis of these data (Courtesy of Mohan Trivedi)

Communicated information will include data collected via VB-DASs, and also data collected via stationary cameras along the road. Inter-car communication will be part of expected *intelligent transport systems* (ITS), defining the general context of future transportation.

Autonomous Driving Autonomous driving is often identified as being the ultimate goal when designing DAS, or VB-DAS in particular [218, 219]. There are already various vehicles available demonstrating that autonomous driving is possible today using, for example, stereo vision [68] integrated into a car, or a laser range-finder [44] mounted on top of a car (together with very accurate e-map and GPS data). In both cases, environment and weather conditions have to satisfy particular constraints to guarantee that the systems will work accurately, and the driver still has the responsibility to be aware about the given situation.

The vehicle industry world-wide (traditional companies as well as newcomers to the field) has assigned major research and development resources for offering competitive solutions for DASs, or autonomous driving. Computer vision in road vehicles can play here 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. VB-DASs also contribute to better driving comfort in modern cars.

Surround Maps Gandhi and Trivedi [72] propose a driver-assistance system by developing a novel method to create a "surround map", using two omni-directional cameras. This type of camera can capture a panoramic view around the ego-vehicle. Due to the large field of view (FOV), the resolution of the recorded videos by an omni-camera is considerably lower than those captured by standard cameras. This needs to be added to distortion errors as well. Therefore, the method could be more suitable for close-by obstacle and vehicle detections. A robust ADAS needs to perform under high-speed driving conditions, therefore a timely detection of hazards from both close and far distances is an essential requirement for any realistic ADAS.

Road Environment When analyzing the road environment, we are typically interested in detecting and interpreting road furniture, pedestrian crossings, curbs, speed bumps, or large-scale objects such as an entrance into a tunnel, or a bridge. Accurate e-maps and GPS provide information about the expected environment; cameras and computer vision can be used to detect unexpected changes in such an environment.

Ground-level recording and 3D reconstruction (i.e. only using cameras in vehicles, not also aerial recording) is a subject of interest. For example, [76, 93, 296] all use multiple cameras while recording road sides in a *single run* when moving the cameras in the ego-vehicle essentially parallel to the road borders in one direction only (i.e. without any significant variations in the path).

3D road-side visualization or 3D environment modelling are applications which still lie beyond the current interest in VB-DASs. A 3D reconstruction from a moving platform [296, 309], possibly in combination with 3D reconstructions from a flying

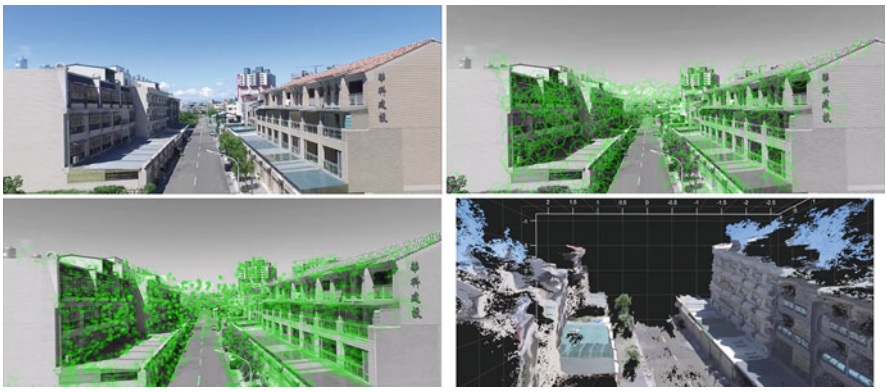

Fig. 1.9 Monocular video data, recorded in a multi-copter (*upper left*) in Kaohsiung, can be used to create 3D models of a given road environment (*lower right*). SURF features (*upper right*) are analyzed for corresponding features in the next frame (*lower left* shows vectors connecting corresponding features); this allows us to estimate ego-motion of the multi-copter as well as the fundamental matrix for performing stereo reconstruction based on two subsequent frames (Courtesy of Hsiang-Jen (Johnny) Chien)

platform such as a multi-copter (see Fig. 1.9), can be used for an even more accurate environment model compared to today's planar e-maps.

Testbeds Crash tests have a long history in the automative industry. Cars are tested under very precisely defined conditions to evaluate their mechanical safety parameters. DASs and autonomous vehicles contain various sensors and control modules which also need to be tested. Those sensors and modules have to measure and understand various traffic scenarios, and the diversity of possible scenarios requires new ideas about the representative design of testbeds for modern vehicles. Testbeds[1] can be public roads (with adequate sensors for measuring a vehicle's performance), or specially designed local traffic areas with opportunities to create especially challenging scenarios, and to make extensive use of sensors integrated into the testbed.

1.7 Scope of the Book

As highlighted earlier, in this book we restrict ourselves to camera sensors and computer vision-based techniques. In the field of computer vision, there is still only limited work done on simultaneous *driver behaviour* and *traffic scene* monitoring. Generally, these topics are treated as individual or independent research directions. Researchers who work on driver's facial or behavioural monitoring normally do not

[1]For an example, see www.n3t.kiwi for the N3T testbed near Whangarei, New Zealand.

work on traffic scene monitoring, and vice versa. Thus, a proper evaluation of the effectiveness of the proposed methods for real-world ADASs is hardly possible.

Helping to fill this research gap, we start in Chap. 3 by recalling the required theoretical fundamentals, and then we continue by developing in detail a practical ADAS application to simultaneously monitor the driver and road scenes. We also aim to validate the proposed methodologies by providing experimental results in *real-time*[2] and for scenes recorded in *real-world* driving scenarios.

Object Detection and Classification Focus In our presentations we will focus on *object detection* and *classification* methodologies: We detect faces, eyes, vehicles and so forth, and evaluate the developed classifiers by using common classification measures.

Motivation We are motivated to work on driver's behaviour monitoring (detecting a driver's direction of attention, eye closure, yawning, and head-pose monitoring), road hazard analysis (especially detection of the vehicles in front, and their distance to the ego-vehicle), finding the correlation between the "driver's direction of attention" and the "road hazards", and a simultaneous analysis of "driver" and "road" data to determine high-risk driving situations. Our ultimate objective is to prevent imminent crashes due to driver fatigue, drowsiness, or distraction.

Monocular Vision Algorithms Besides also briefly discussing stereo vision, we focus in this book on opportunities given by monocular vision (e.g. when using a mobile device only allowing us to use one camera in one viewing direction). We are especially interested in monocular-vision algorithms for target classification and tracking for both in-vehicle (i.e. driver) and out-vehicle (i.e. road) monitoring. Low computational cost of monocular vision techniques has been a motivation for us that may facilitate the implementation of our research outcome in the form of cost-efficient chips, integrated circuits and systems, or even real-world apps for smartphones or other mobile devices.

Face Detection As a prerequisite for driver-behaviour monitoring, and before going for driver's *facial feature* analysis and driver's *head-pose estimation*, we need a very accurate and robust face detection algorithm.

There is a widespread understanding of the acceptable accuracy of existing face detection methods such as [8, 224, 271] under ideal conditions. Some researchers work on pose variation [187] and rotation-invariant [28] face detection techniques. There are also recent research attempts to improve the speed and detection accuracy of current face detection algorithms [188, 226].

A literature review of existing methods shows that the Viola–Jones (V-J) method based on *Haar-like features* and a cascade of classifiers [269, 271] rates the highest worldwide among other proposed face detection methods. Since the introduction

[2]In this book any processing speed of more than 15 frames per second (fps) is considered real-time, as it enables us to provide timely information to assist a driver, even in high-speed driving scenarios.

of the method almost 10 years ago, it has received several thousands of citations in the literature. Recent research also uses deformable part models (DPMs) [63] to classify an object as a joint detection of different parts; nevertheless, their speed would be up to 20 times slower than that of the standard classification methods [61].

Accuracy and Low Computational Costs The considered high-dimensional DAS needs to monitor both inside and outside the vehicle. It has to perform fast enough to ensure real-time processing, real-time decision making, and real-time responding to the driver. So, any proposed method in this research needs to be both accurate and have low computational costs to comply with the real-time requirements, and also must use only common existing hardware at the time of the research.

Lighting Conditions Today's digital cameras and many other commercial devices widely use the V-J method for face detection [310]. However, there are serious issues with the V-J method, even for frontal face detection if there are noisy images, or under difficult lighting conditions, as discussed in some very recent publications; see [107, 206], or [246].

The general success of the V-J method for real-time face detection, and, on the other hand, the existing gap to be closed by coping with its weakness in difficult lighting situations [206], have been strong motivations behind attempts to improve the V-J method, as discussed in this book.

Contributions in the Book Considering the discussed motivations, requirements, and the existing research gaps, the book contributes as follows:

We first refine the training step, including refinements of parameters of classifier and dataset, to create a more efficient cascade of classifiers using the same Haar-like features as originally proposed for the V-J method.

Our next contribution is the development of two novel classifiers based on *adaptive global Haar-like features* (AGHaar) and *dynamic global Haar-like features* (DGHaar). This improves the original methodology and makes it a significantly more robust detector by ensuring a higher rate of true detections, a lower rate of false positives, and faster performance. We use the proposed classifiers for both facial feature analysis and *vehicle detection* under various outdoor or weather conditions, and for noisy images.

In order to define a correlation between the driver's looking direction (i.e. the driver's direction of attention) and road hazards, we define a 6-dimensional driver's head pose including head tilt, yaw, pitch, and its (x, y) location at time t. Not limited to a driver's face, many symmetric objects may appear as being asymmetric, depending on the location and angle of the light source, or due to an unbiased lighting condition. This is the source of many object misinterpretations and misdetections. For example, under real-world driving conditions, one part of the driver's face may receive more light from the side window than the other half of the face. Under such conditions, half of the face (or object) may provide completely different feature-points, and considerably less information due to falling into a darker space. We develop an asymmetric active appearance modelling method (ASAAM) and a novel technique called the *Fermat-point transform* to resolve the

issues of intensity asymmetry, as well as gaining a highly-accurate 3D head-pose estimation from a 2D image.

In the second part of the book, we contribute by proposing algorithms for multiple vehicle detection and inter-vehicle distance estimation in road scenes. We propose various techniques such as *virtual symmetry detection*, horizontal edge detection, corner-point analysis, as well as a multi-clue data fusion technique, based on Dempster–Shafer theory. The introduced techniques altogether enable successful vehicle detection under various lighting (from very bright to very dark) and noisy conditions. We also develop a hybrid distance-estimation technique based on bird's-eye views and camera-pose trigonometry, using a monocular camera only.

Another gap in the field is the absence of publicly available road datasets, especially for situations such as night, rain, fog, or other challenging driving scenarios. Wherever applicable, we use existing datasets provided by researchers, to benchmark our proposed methods. To tackle the absence of challenging road datasets, we also develop our own iROADS dataset [214] for a diversity of lighting and weather conditions. We perform experimental results using both publicly available datasets and our developed dataset to evaluate the robustness of the proposed methods.

Ultimately we define a fuzzy-logic-based fusion platform that processes and correlates the obtained data from driver's behaviour with the location and distance of the detected vehicles on the road. The fusion module acts as a decision layer that can effectively identify the risk-level of a given driving situation, and is expected to provide a timely warning to prevent an imminent crash. We also expect that the entire developed ADAS (including driver's head-pose analysis, drowsiness detection, vehicle detection, distance estimation, and data fusion) can perform reasonably well in real-time.

Chapter 2
Driver-Environment Understanding

2.1 Driver and Environment

As in cases of computers where the cause of a problem is in general sitting in front of the machine, traffic accidents are typically caused by the driver.

Inattentive Driving Studies in the United States by the National Highway Traffic Safety Administration [265] and the Virginia Tech Transportation Institute [272] pointed out that almost 80% of all types of vehicle accidents involve driver fatigue, driver drowsiness, or driver distraction, in general summarized as *inattentive driving*, within the last 3 s prior to a catastrophic accident [141]. Driver inattention is a contributing factor for 93% of all rear-end collisions in this study [272]. This contradicted some prior studies which showed only a low contribution rate of 25% for driver distraction.

Studies in small developed countries such as New Zealand (with a population of 4.5 million) also show similar statistics. The 2010 statistics by the New Zealand Ministry of Transport reports 384 fatal crashes, 14,541 serious injuries, and a $5.3 billion national cost; 52% of all road deaths have been caused by drivers falling asleep at the wheel, driver fatigue, or driver distraction [29, 176, 179].

Simultaneous Evaluation of Driver and Road Hazards These statistics highlight the importance of research and development of *advanced driver-assistance systems* (ADASs) focusing on "driver behaviour". Therefore, introducing a proactive safety system that is able to simultaneously evaluate a driver's level of alertness as well as the road hazards, is of high interest. This is also a main subject in this book.

Three Classes of Parameters Generally, an ADAS can include a set of active safety systems that work together to increase the safety of the driver, passengers, pedestrians, and other road users [120]. The objective is to recognize critical driving situations by perception of the vehicle or driver, characterized by *internal parameters*, or of road hazards characterized by *external parameters*. Additionally,

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_2

often less highlighted in this context, we also need to consider the weather and lighting conditions, characterized by *circumferential parameters*.

Internal parameters may include the ego-vehicle's speed, steering angle, or the driver's level of awareness. We recall that the *ego-vehicle* is the vehicle where the considered system operates in; it is equipped with different sensors to monitor the road or the driver status.

The external parameters may describe stationary or moving vehicles around the ego-vehicle, road boundaries, traffic signs, and vulnerable road users such as pedestrians and cyclists.

Lighting (day or night) or weather conditions can be modelled by circumferential parameters. These parameters can highly affect the accuracy and performance of the ADAS.

Driver or Road Monitoring Chapter 1 reviewed the various subject areas related to traffic safety and driver assistance systems. Subject areas such as traffic sign recognition [159], lane detection [236], pedestrian detection [49], vehicle detection [241], and driver behaviour monitoring [175], including driver fatigue, drowsiness and distraction detection, already have numerous publications devoted to them. In this chapter we review selected research results classified into two main categories, either related to driver or road monitoring. Later in this book we will connect both categories by discussing integrated solutions.

2.2 Driver Monitoring

In our selective review we comment on work in the field by discussing briefly the strengths, weaknesses, or limitations of work done in the past decade, up to recent state-of-the-art research in academia and industry.

We expand on the initial comments regarding driver monitoring already given in Sect. 1.5. Driver monitoring comes with several difficulties associated with the challenging nature of this application. Issues can be variations in scale, pose, direction, facial expression, lighting conditions, or occlusions.

Face Detection Starting with face detection, there exist different approaches that can be classified into four categories [301, 310]:

Knowledge-based methods that use some predefined (thus "knowledge-based") rules to assess some common *facial features* that exist in any "typical" face.

Feature-invariant methods that use some *structural features* for a face, so in case of success they can detect faces in different poses or rotations.

Template-matching methods that use a few selected patterns of the whole face, patterns of facial parts, and individual facial feature patterns (e.g. eyes, nose, or mouth) as the reference for face detection; then the correlation of an input image with the stored patterns is examined to confirm or reject an image as a face. Since face models may vary for different people, the locating results are

heavily affected by face modelling and image contrast. High computational costs also prevent a wide application of this method.

Appearance-based methods that use a face dataset to learn the variation of face appearances. The learning process is on the basis of common photometric features of the object, and is based on a collective set of face images in different poses.

Face and Eye Detection Kasinski and Adam propose a hierarchical architecture for face and eye detection using an ordinary *cascade of Haar classifiers* (CHCs) augmented by two simple rules [111]. The method applies Haar-like features for eye detection within an already detected face, and it rejects a face candidate if ϑ, the angle of detected eye-pairs, is larger than 20°, or, if no eye-pair is detected within a face. The paper claims outperformance of elsewhere proposed classifiers with respect to both accuracy and training speed. However, the method is tested on an unknown dataset, and the defined rules cannot always be true. In many cases a face might be in a (semi-)profile pose, so one of the eyes could be invisible or occluded due to the nose-bridge. In such cases, the proposed method simply fails by rejecting an existing face as a non-face. We expect many false negatives in realistic situations; especially under driving conditions where the driver needs to constantly look over to the side mirrors. On these occasions, the driver's face may vary in terms of pose and position. Furthermore, the images in the dataset used for our experimental results have a very high resolution of 2,048 × 1,536 pixels, under controlled ideal lighting conditions, with no specific challenges, noise, or detection difficulty. In real-world driving scenarios we have to expect more difficult and noisy conditions.

Wilson and Fernandez [289] propose a regionalized search technique. The method tries to reduce the search area of face detection aiming at minimized false positive detections. They simply consider the upper $\frac{5}{8}$ of a face for eye detection, and the lower part for mouth and nose detection. No statistical analysis is provided.

Facial Features Batista [14] offers a framework that combines the location of facial features with a skin-colour-based face detection approach. He considers manually defined proportional distances for eye, eyebrow, mouth, and nose to find facial features after image binarization. The defined parameters are not necessarily correct for all types of faces, races, and genders. Furthermore, the shown experimental results are performed under indoor conditions with high-quality images. No experiments were done under actual driving conditions.

Lighting Conditions Under the Lambertian reflection assumption [13] and having multiple light sources, a face (or an object) can form a very large polyhedral illumination cone [130], in which the cone contains all the variations of a fixed pose face for different angles of lights and shades. This makes a large number of extreme rays for every single face pose. Not dealing with every single pixel, even if we judge a face detection based on facial feature blocks of 20×20 pixels, then there will be an illumination cone with $\mathcal{O}([\frac{n}{20 \cdot 20}]^2) = \mathcal{O}(n^2)$ extreme rays, where n is the number of pixels in the given image.

Fig. 2.1 Lighting conditions may cause a significant loss in edge or facial feature information. The same person, the same pose, and only a 20° change in light-source angle (See the public-domain Yale dataset [130])

This is one of the most important technical concerns which is often neglected or ignored in many publications such as [107, 204, 224]. For the same reason, many datasets used for driver behaviour monitoring and subsequent performance analyzes are incomplete or inaccurate.

Figure 2.1 exemplifies the effects of changes in light source directions. It shows dramatic effects on feature and edge information, especially for symmetric objects (e.g. a face). This is a very common case that happens under driving conditions when half of the driver's face receives more sunlight from the side window, and the other half is in a darker environment. As shown in Fig. 2.1, bottom right, no meaningful edges or features can be perceived in the left part of the given face. Many of the current face detection algorithms may easily fail under such conditions.

We consider this concern as one of the important difficulties for driver-behaviour monitoring, depending on the sun's position, or based on artificial street-lights in evenings or nights.

Yale B Face Database To the best of our knowledge, the *Yale B face database* [300] is the only publicly available dataset that provides more than 5,000 face images recorded under different light-source angles. The images are recorded under controlled indoor conditions using a single light-source. Later we demonstrate the use of this dataset within our experiments.

2D Cascade Training Different to the Viola–Jones (V-J) method that uses cascade bootstrapping to reject negative images at every stage, Niu et al. [315] introduce an eye detection system based on bootstrapping of both positive and negative images. Therefore, the method deals with larger training datasets more quickly by rejecting some redundant positive images at every stage of training. The experimental results from high-resolution datasets such as *BioID* [101] or *JAFFE* [146] show improvements due to the ability to learn a bigger training dataset in a given time, compared to the standard V-J method. Despite the improvements, the method is not robust under unconstrained conditions, or for *noisy images* that are affected by motion blur, out of focus images, lighting artifacts, or facial expression factors.

Driver-Fatigue Analysis Some researchers specifically address driver-fatigue analysis. The work presented by Wang et al. [275] uses a colour correlogram [106] and the AdaBoost [70] machine learning strategy for eye-state detection. The authors believe that a colour correlogram represents global and rotation-invariant features of eye images; therefore they express that the insignificant differences between left and right eyes could be ignored.

Without explicitly detecting the eye status, Sigari [239] considers spatio-temporal information in the upper half of the driver's face for driver fatigue detection. The method measures the average horizontal projection in the first 100 frames of driving as the base information, and then compares the correlation of horizontal projection of the current frame with the base data. If the correlation is lower than a threshold, the system considers as the eyes being closed. The method is basic as it ignores many existing parameters that impact on the performance of the detection. Sudden changes in lighting conditions due to street lights, driving in a tunnel, or on-off intensity changes due to shades of street trees are the possible parameters that can influence the accuracy of the system.

A cognitive fatigue detection method is provided by Miyaji et al. [163] using stereo vision to detect a subject's head and eye movement, and ECG-based heart rate monitoring. The author compares the standard deviations of head and gaze angle for two states of "normal driving" and "driving with cognitive loads". The research concludes that in the case of mental loads, the gaze becomes more concentrated with smaller standard deviations for the gaze angle. The result is interesting and can be useful for determining the overall state of a driver's vigilance. However,

calculating the standard deviation of eye gaze requires a measurement during the past few minutes of driving, which causes the method to always lag behind the actual moment of a distraction. This could be very late to prevent a crash. The method is only implemented in a simulation testbed.

Percentage of Eye Closure Since 2010, the characterization of the percentage of eye closure (PERCLOS) [284] has also became one of the common approaches in computer vision for sleep-deprivation studies and driver-drowsiness detection [50, 75, 233]. This approach assesses the driver's alertness based on the residual increase in percentage of eye closure, over a given period of time. If the eye-closure time becomes more than 80%, it is considered to be a sign of sleepiness. Despite the popularity of the method, PERCLOS alone is now known to be insufficient for preventing a crash due to driver unawareness, especially in high-speed driving scenarios when a microsleep [148] can cause a catastrophic accident. A microsleep may suddenly happen with any driver, without any pre-warning or early signal that could be measured via a computer-vision approach.

Mouth and Yawning Analysis Wang et al. develop a mouth movement tracking method using a dashboard-mounted CCD camera to detect and record the yawning frequency of a driver [276]. The method determines the face region based on skin colour analysis, followed by lip pixel segmentation using a Fisher classifier and connected component analysis (CCA). The main contribution of the work is the application of a belief-propagation artificial neural network BP ANN to detect three mouth-states, including closed, open, and widely open (yawning). The paper only considers mouth status as a fatigue sign of a driver, while many drivers may fall asleep at the wheel without yawning.

Driver's Vigilance The influential paper by Bergasa and Nuevo [16] uses an infrared vision system to track a driver's vigilance via analyzing six parameters from a driver's face: percentage of eye closure, duration of eye closure, blinking frequency, frequency of head nodding, face position, and fixed gaze detection. Due to using an infrared camera, the method could be robust for pupil detection for night driving, but the performance could dramatically decrease in daylight, especially on sunny days. The paper determines the position of the face based on the location of pupils and the nose tip. This can give a rough estimation of a head or face; but, it could easily fail or lead to wrong results in case of any false pupil and nose detections. In Chaps. 5 and 6 we further discuss these two challenges by contributing two robust approaches to eye status monitoring under day or night conditions, as well as highly accurate head-pose detection.

In another recent work, Miyaji et al. [164] also combine means of psychosomatic and computer vision approaches to assess the driver's level of vigilance. The authors introduce a hybrid method using SVM and an AdaBoost learner for classification of *heart rate* beat-to-beat intervals (known as R-R intervals) using the EEG waveform and visual information such as pupil diameter. The main deficiency of this approach is the requirement of data normalization and time alignment due to significant differences between the assessed data-types (e.g. computational time difference of

computer vision-based techniques compared to heart-rate analysis). In an offline processing, the final results of the method are expected to be more accurate than individual solutions; nevertheless, the method is only applied on a driving simulator, hence, the feasibility of the system and EEG wiring in actual driving conditions requires further assessment.

Driver-distraction Analysis Using a commercially available gaze tracker, Doshi and Trivedi [52] have worked on the parameters that may lead to distraction. The study is done on a subject's gaze shift while listening to simulated lecturing, and the outcome of the study is directly generalized for driver-distraction analysis. The method itself is carefully evaluated; however, we believe the obtained results for simulated lecturing could hardly fit into the context of distraction in a driving scenario.

Other researchers such as Peiris et al. [191, 192], Portouli et al. [200] and Furman et al. [71] proposed methods for measuring distraction by considering physiological phenomena such as EEG, ECG, skin voltage, heart rate change and respiration. The results are expected to be more accurate than vision-based techniques but the implementation and application of such systems in real-world driving situations need wiring on the driver's body, which is impractical, or even more distracting. In addition, a connection of such extra items to a driver's body may raise unexpected distractions even for a normally alert driver. This is one of the reasons that we prefer to focus only on non-intrusive vision-based approaches in this book.

Some researchers work on driver distraction [175] based on driver's head-pose estimation. Techniques such as pose detection from orthography and scaling (POS) [45], POS with iteration (POSIT) [153], or using random forest techniques [238] are widely considered. Some researchers like Fletcher and Zelinsky [66] used stereo-vision data for pose analysis; each method suffers either from high computational costs, low accuracy, or insufficient robustness.

In the next chapters we propose solutions for coping with some of the discussed challenges for driver behaviour monitoring, also called *in-vehicle* monitoring.

2.3 Basic Environment Monitoring

The basic traffic environment consists of the ego-vehicle, other vehicles or pedestrians, traffic-relevant obstacles or signs, the *ground manifold* (i.e. the geometric ground-level surface; often it can be assumed that the ground manifold is approximately a ground plane, at least in a local neighbourhood of the ego-vehicle), the road, and the lanes. Section 1.5 already presented a few brief comments about work on environment monitoring.

Ego-motion describes the ego-vehicle's motion in the real world. Vision can help to control ego-motion according to the given obstacles or planned manoeuvres. For *basic navigation support*, only the existence of obstacles needs to be detected without understanding their type or movement, or the possible implications of those movements.

Out-vehicle (i.e. road) monitoring is different to *in-vehicle* (i.e. driver) monitoring in many aspects. For an actual course of driving we may expect different situations (e.g. driving straight ahead, turning, overtaking or avoiding lead vehicles, obstacles, or pedestrians) where each case can define different kinds of challenges [207].

Obstacle Detection Monocular or stereo vision, often together with further sensors, provides input data for detecting vehicles, pedestrians, or further *obstacles* on the road [240].

For example, when applying stereo vision, detected points in the 3D scene need to be analyzed for being just noise or actually obstacles on the road. A detected local cluster of points at some height above the road define a *stixel* [7], which is a straight cuboid standing on an assumed ground plane and limited in height by the detected local cluster of points. *Regular stixels* are formed when assuming cuboids whose lower faces define a regular grid on the ground plane. See Fig. 2.2. Stixel classification can then aim at identifying basic object shapes like *car*, *bus*, *traffic sign*, or *construction cone*. Stereo vision also supports object detections on non-planar roads [137]. Generic object detection is studied in [143, 278], showing a good performance on [118].

Monocular object detection [198] has also been intensively studied for cases of monocular video recording (e.g. if attaching a mobile device to the windshield of a vehicle). Ali and Afghani [3] infer a vehicle (driving in front of the ego-vehicle) from the shadow underneath. Rezaei and Klette [211] 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. See Fig. 2.3. The book [78] discusses the detection

Fig. 2.2 *Left*: Use of a colour key (different to the one shown in Fig. 2.4) for showing depth data calculated by stereo matching. *Right*: Illustration of calculated stixels (based on the depth data illustrated above, forming an *occupancy grid*), groupings of stixels, and of estimated motion for such stixel groups (Courtesy of Uwe Franke)

Fig. 2.3 *Top*: Monocular vehicle detection under challenging lighting conditions, to be discussed later in this book in detail; detected vehicles are also labelled by monocular distance estimates. *Bottom*: A transform from perspective view into bird's-eye view is used for monocular distance estimates (Courtesy of Ruyi Jiang)

of pedestrians in a traffic context; see also [150] and the database [53]. For a survey paper on pedestrian protection, see [73].

Vehicle Detection Vargas et al. [267] provide a vehicle detection system using sigma-delta-based background subtraction to separate moving vehicles (foreground) from the road (background). The recording camera is fixed (on a non-mobile platform). The method is simple and computationally cost effective. It appears to be well-suited for monitoring traffic density. The method is not able to identify individual vehicles.

Haselhoff et al. [86] use a radar sensor to minimize the *region of interest* (ROI) for a computer-vision based approach that uses standard Haar-like features for vehicle detection. Reducing the search region can lead to fewer false-positives; still, there appear to be many weaknesses such as time synchronization issues for radar and vision sensors, and the increasing cost for the system.

O'Malley et al. [185] propose vehicle detection based on taillight segmentation. The authors hypothesize that tail and brake lights in darkness tend to appear as white spots surrounded by a red halo region. This assumption may not be necessarily true, as current cameras have auto-exposure control, and do not capture a white central spot for a red taillight. A second weakness is that this approach only works for night-time conditions, and a third weakness is that the method only works for the detection of vehicles which are at the same level as the ego-vehicle. A tilted vehicle

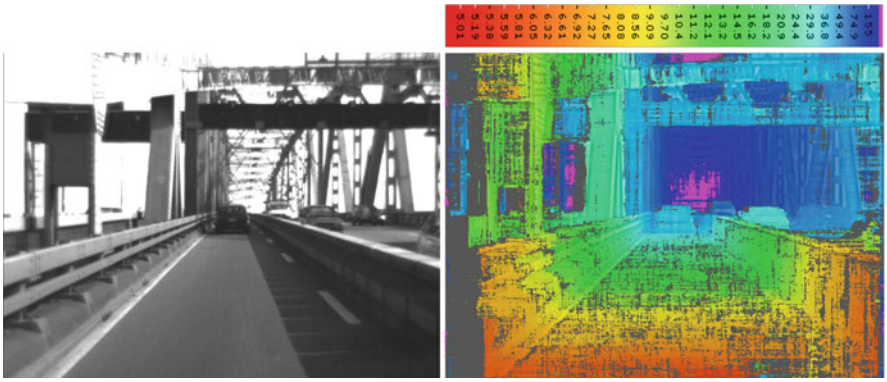

Fig. 2.4 *Left*: One image of a stereo pair recorded on Auckland's harbour bridge. *Right*: Visualization of a depth map using the *colour key* shown at the *top* for assigning distances in metres to particular colours. A *grey pixel* indicates low confidence for the calculated depth value at this pixel (Courtesy of Simon Hermann)

(e.g. due to a road ramp, road surface at a curve, or when turning at a round-about) cannot be detected by the proposed method.

Distance Computation *Stereo vision* is the dominant approach in computer vision for calculating distances. *Corresponding pixels* are here defined by projections of a surface point in the scene into images of multiple cameras. The applied vision system knows about the calibration data of those cameras and rectifies the recorded images into canonical stereo geometry so that a 1-dimensional (1D) correspondence search can be constrained to identical image rows.

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, sunglare, running wipers, and so forth) still may pose problems. See Fig. 2.4 for an example of a depth map. For example, stereo vision, combined with motion analysis (called *6D vision*, see [1]), provides basic information used in Daimler's "Intelligent Drive" system.

The *third-eye performance evaluation* [168, 236] provides a way to control the accuracy of an applied stereo matcher (i.e. of calculated disparity values). A measure based on *normalized cross-correlation* (NCC) is used for evaluating disparities frame by frame, thus identifying situations where a selected stereo matcher fails (and should be replaced by another matcher; see the section on adaptive solutions above).

Combining stereo vision with distance data provided by laser range-finders is a promising future multi-modal approach towards distance calculations (recall that this book discusses vision sensors only). There are also ways to estimate distances in monocular video data [211], and they will be discussed in detail later in this book.

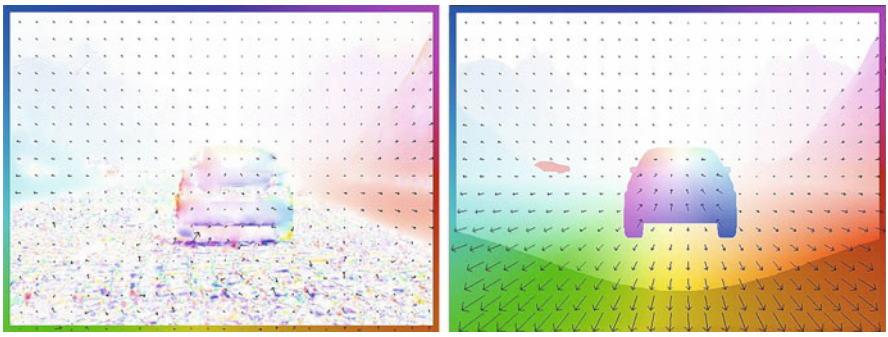

Fig. 2.5 Visualization of optical flow using the colour key shown around the border of the image for assigning a direction to particular colours; the length of the flow vector is represented by saturation, where the value 'White' (i.e. undefined saturation) corresponds to 'no motion'. *Left*: Calculated optical flow using the Horn–Schunck algorithm published in 1981 for the image shown *on the right* of Fig. 1.3. *Right*: Ground truth provided by EISATS (Courtesy of Tobi Vaudrey)

Motion Computation *Dense motion analysis* aims at calculating approximately-correct motion vectors for "basically" every pixel position $p = (x, y)$ in a frame taken at time slot t [119]; see Fig. 2.5 for an example. *Sparse motion analysis* is designed to have accurate motion vectors at a few selected pixel locations. Dense motion analysis is suitable for detecting short displacements (known as *optical flow* calculation) [105], and sparse motion analysis can also be designed to detect large displacements [161]. Motion analysis is a difficult 2-dimensional correspondence problem, and solutions might become easier by having recorded high-resolution images at a higher frame rate in future.

Moving objects in a traffic scene can be tracked by using repeated detections, or by following an object detected in a frame recorded at time t to a frame recorded at time $t + 1$. A *Kalman filter* (e.g. linear, general, or unscented) can be used for building a model for the tracked motion as well as for involved noise [199]. A *particle filter* can also be used based on extracted weights for potential moves of a particle in particle space.

Ego-Motion Object tracking is an important task for understanding the motion of the ego-vehicle, or of other dynamic objects in a traffic scene. *Ego-motion* needs to be calculated to understand the movement of the sensors installed in the ego-vehicle. For example, an *inertial moment unit* (IMU) in the ego-vehicle provides a non-vision approach to ego-motion analysis.

Visual odometry uses recorded video data to calculate ego-motion; see Fig. 2.6. Possible approaches are characterized by *feature* tracking (a feature is a *key point*, i.e. a pixel, in one frame together with a *descriptor* characterizing image data around this key point; see [119]), *bundle adjustment* [259, 313] (i.e. the combined analysis of camera movement and of detected 3D points in the traffic scene), or by direct motion estimation, e.g. by simply applying an optical flow algorithm combined with

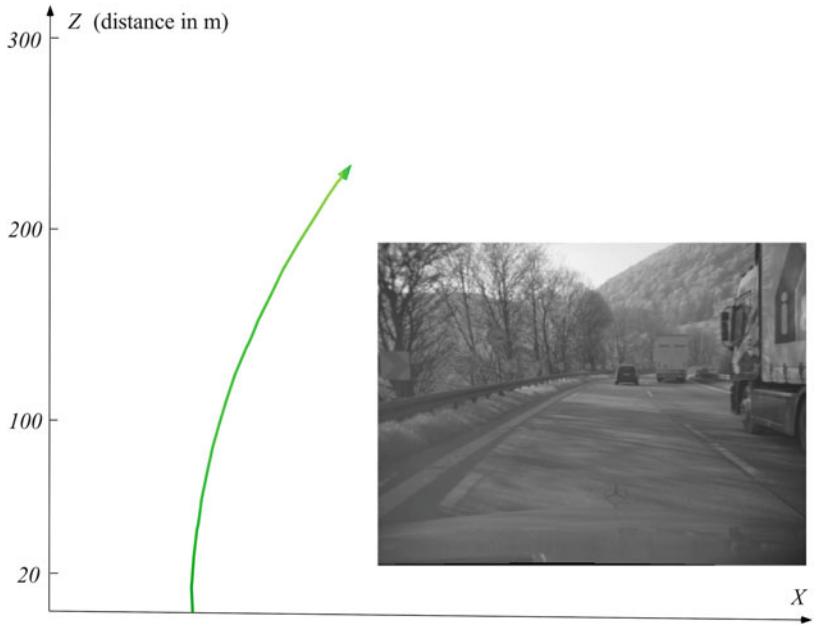

Fig. 2.6 Calculated trajectory for the ego-vehicle of sequence 3 in Data-set 1 of EISATS (Courtesy of Ali Al-Sarraf)

non-visual sensor data such as GPS or of an inertial measurement unit (IMU) [123], or, more advanced, by applying 6D vision [1].

Triggs et al. [259] defines *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 the Levenberg–Marquardt algorithm.

2.4 Midlevel Environment Perception

Object analysis (following a detection), object tracking, object clustering, and object recognition are examples of medium-level vision, which must be understood before attempting to approach complex dynamic scenes in high-level vision.

Detection and Tracking There are static (i.e. fixed with respect to the Earth) or dynamic objects in a traffic scene which need to be detected, understood, and

possibly further analyzed. Typically, those objects are either the ego-vehicle itself, other on-road vehicles (including bicycles, for example), or pedestrians.

Vehicle Tracking Vehicle tracking is an important component of *collision avoidance systems*. By analyzing the trajectories of visible vehicles, in comparison to the trajectory of the ego-vehicle, it is possible to understand the danger of an imminent crash (e.g. it may be used to trigger autonomous braking).

Tracking by repeated detection can use techniques as mentioned above in the section on vehicle detection. In general, it is beneficial to use stereo vision results (i.e. disparity or depth values) in a vehicle tracking procedure [12], and not only monocular data.

Vehicle tracking is typically easier to perform than pedestrian detection and tracking; the shape and appearance of vehicles is easier to model (e.g. by the appearance of lights, bumpers, horizontal line segments, density of detected corners, or visual symmetry; see [211]). Vehicle tracking is difficult due to occlusions, difficult lighting (e.g. light artefacts due to trees and intense sunshine), "ghost appearances" (e.g. reflected car headlamps on a wet road), and many more possible issues. Learning from ensembles of models has been proposed in [183], using the data of [118] for training and testing. Supervised learning enhances the creation of a *discriminative part-based model* (DPM) from recorded video data [62, 303].

Pedestrian Tracking Pedestrian detection, tracking, and understanding are in general still very challenging subjects. The task simplifies if one only considers pedestrians crossing the road, and not also pedestrians who are close to the road (e.g. for determining whether a pedestrian will step on to the road in the next moment, or whether a child might possibly throw a toy onto the road). A straightforward approach to tracking is by repeated detection, possibly refined by taking previous detection results into account (up to Frame t) when analyzing Frame $t + 1$.

A standard procedure for detection 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 oriented gradients* (HOG) for the bounding box [41]; after deriving *HOG descriptors*, the classifier uses those to make a decision about the presence of a pedestrian. It is also possible to use such HOG descriptors within a *random decision forest* (RDF) [24] to perform the classification task.

For example, if the bounding box arrives at any leaf in the used forest which has a probability greater than 0.5 for the class "pedestrian", then the box may be classified in this way (using this simple maximum-value rule). In case of overlapping bounding boxes, results may be merged into a single detection or box. See Fig. 2.7.

Performance evaluation of pedestrian detection or tracking can be based on image data with manually identified ground truth; see, for example, the *Caltech Pedestrian*

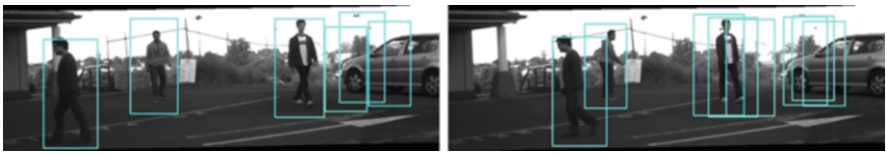

Fig. 2.7 Two frames of a sequence of detected pedestrians using an RDF. There are a few false-positives (on the right in both frames), and also a few overlapping true-positive *bounding boxes* (in the frame on the right for the person on the right). Further processing needs to eliminate false-positives, and to unify *overlapping boxes* (Courtesy of Junli Tao)

Fig. 2.8 *Left*: Unpaved road near Salta in Argentina. *Right*: Roads in tunnels (below the historic centre of Guanajuato) (Courtesy of the authors of [237])

Detection Benchmark.[1] The *TUD Multiview Pedestrian* and the *CCV Pedestrian* databases can be used for body direction classification.[2]

Detection of Infrastructure Key Elements The road, marked lanes, and traffic signs define the key elements of the traffic-related infrastructure of the environment.

Road Detection Road detection is often considered as a pre-processing module prior to lane analysis [5], especially in cases of non-highway driving. Wedel et al. [279, 280] discuss ways to model the visible road surface in front of the ego-vehicle.

The road might be identified by curbs, a particular surface texture, or by a space between parked cars on both sides of the road, but also by very specific properties. See Fig. 2.8 for two extreme cases. In the case of roads in tunnels, the walls and ground-manifold may have the same texture, but differ by their surface gradients. In the case of an unpaved road in a desert, the texture of the road surface may continue on the left or right, and only traces of previous traffic may indicate the actual location of the road.

[1] www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/

[2] They are available at www.d2.mpi-inf.mpg.de/node/428 and ccv.wordpress.fos.auckland.ac.nz/data/object-detection/ for free download.

Lane Analysis In a general sense, a *lane* is defined by a sufficient width for driving a vehicle; it is the space between a left and a right *lane border*. Many different mathematical models have been used for defining lanes (e.g. analytically defined curves or sequences of individual border points following some kind of systematic pattern). In the simplest case, straight segments are used to describe zero-curvature lane borders, and second order curves or clothoids for non-zero-curvature lane borders.

There is already a wide variety of solutions available for lane analysis; see [9, 115, 144, 156, 237].

Lane detection is basically "solved" for scenarios during driving where lane markings, lane geometry, and visibility conditions are reasonable, but there is still a need to study lane-border detectors or trackers in challenging scenarios (e.g. underground road intersections, unpaved roads, or very wide road intersections without any lane marking).

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 from the true lane border*". What exactly is the "true lane border"? How do we measure it in cases such as the one illustrated in Fig. 2.9? The KITTI vision benchmark suite [118] offers a few manually-labelled frames for evaluating lane detection. Synthetic data for evaluating lane border detectors are available on [248].

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. 2.9. Additional knowledge such as the width of the car or the previous trajectory of the car can be used to estimate the continuation of lanes.

Borkar et al. [20] proposes a semi-automatic technique for generating ground truth for lane detection. They use *time slices*, defined by taking a specified single row with detected lane locations in subsequent frames, and fitting 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

Fig. 2.9 Three examples of data provided by [20] where a used lane detector follows its strategy for detecting lane borders by temporal inference, but where one given frame alone would be insufficient for a judgement of 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)

clearly-marked roads. The involved interaction comes with the risk of human error and limited usability.

Curb Detection Wijesoma et al. [285] develop a technique to detect road curbs using a 2D LIDAR scanner. The proposed method is simpler and faster than millimetre-wave radar-based measurements. The method does not suffer from still existing limitations of video-based approaches; however, the method may fail in detecting curbs having a height smaller than 25 cm due to natural limitations of LIDAR resolution. In other words, the methodology is not very accurate for non-standard roads, curb-less roads, damaged roads, or roads with ramps.

Traffic Sign Detection and Recognition *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 it. Classes of road signs can define one particular module of a complex computer-vision system for ego-vehicle control. For a survey on traffic sign detection, see [165].

Wen et al. [282] propose a two-step heuristic method for text and sign detection from road-video recordings. The method firstly tries to localize the traffic signs, and then aims at text detection within an already detected traffic sign. The main novelty of the work is text detection by integrating 2D image features with 3D geometric structures extracted from the video sequences. The method uses a heuristic approach based on a combination of feature-point detection (possible horizontal text lines), the Gaussian mixture model (GMM), and the minimum-bounding rectangle (MBR) that defines a rectangular region for a sign-candidate. The method is not able to detect circular or triangular signs. The method may also fail by false detection of street advertisement boards, or printed advertisements on the back of buses.

A standard approach [58] can be briefly sketched as follows: possibly preprocess an image (e.g. by mapping a colour image into HSV colour space [119]), detect geometric shapes (circles or polygons) which are potential candidates for a traffic sign (possibly using colour as a guide as well), extract features, and compare those with features of a database of traffic signs.

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 [119] or a radial-symmetry approach [11]. Recorded images are subdivided into regions of interest (i.e. left or right of the road, or on top of the road) according to size-priors for traffic signs in those regions. See Fig. 2.10, left, for a case when detecting image features uniformly, all over the image, in the middle when restricting the search to regions of interest, and on the right, illustrating the diversity of traffic signs. Traffic sign categorization is a central subject in [58].

Fig. 2.10 *Left, top*: Detected relevant features in an input image. *Left, bottom*: 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)

The authors of [173] suggest an evaluation methodology for traffic sign recognition by specifying measures for comparing ground truth with detected signs. Of course, before applying this methodology the ground truth needs to be available, and so far it is provided manually. GPS and digital maps (also called *e-maps*) allow us to compare locations of detected traffic signs with mapped locations of signs.

Free Space Detection and Corridor *Free space* is the area where the ego-vehicle may evolve safely. Crisman and Thorpe [38] is an example of early work to detect free space based on colour analysis. Free space can be conveniently illustrated by using an occupancy grid.

More recent solutions in VB-DASs use stereo vision to calculate occupancy grids [230, 279, 280]. See Fig. 2.2, bottom, for a stixel-illustration of an occupancy grid.

The space where the ego-vehicle is assumed to be driving through in the next few seconds may be called the *corridor*. The corridor has approximately the same width as the ego-vehicle, and it should be a subset of the free space. See Fig. 2.11. The corridor can be estimated based on previous motions of the ego-vehicle, detected lane borders, and calculated free space.

Fig. 2.11 *Left*: Detected lane borders. *Right*: Expected area the ego-vehicle will drive in (i.e. the corridor), adjusted to the lane border after crossing lanes (Courtesy of Ruyi Jiang)

In Chaps. 4 and 7 we discuss solutions for out-vehicle monitoring, in particular for vehicle detection, vehicle tracking, and distance estimation. In Chaps. 5 and 6 we report on methods for driver drowsiness and inattention detection. Before going into these detailed discussions of particular techniques, we briefly recall some concepts from the area of computer vision in the next chapter.

Chapter 3
Computer Vision Basics

3.1 Image Notations

In order to maintain a consistent approach throughout the book, hereafter we refer to digital images I as being defined on a 2-dimensional (2D) grid.

Pixels, Pixel Locations, and Intensity Values An image consists of a rectangular array of pixels (x, y, u), each represented by a location $p = (x, y) \in \mathbb{Z}^2$ and (assuming a grey-level image for the time being) an integral intensity value u, with $0 \leq u \leq G_{\max}$, where, for example, $G_{\max} = 2^8$ or $G_{\max} = 2^{16}$. The image domain, also called the *carrier*,

$$\Omega = \{(x, y) : 1 \leq x \leq N_{cols} \wedge 1 \leq x \leq N_{rows}\} \subset \mathbb{Z}^2 \tag{3.1}$$

contains all the pixel locations, for $N_{cols} \geq 1$ and $N_{rows} \geq 1$.

We assume a left-hand coordinate system as shown in Fig. 3.1. Row y contains pixel locations $\{(1, y), (2, y), ..., (N_{cols}, y)\}$, for $1 \leq y \leq N_{rows}$, and column x contains pixel locations $\{(x, 1), (x, 2), ..., (x, N_{rows})\}$, for $1 \leq x \leq N_{cols}$.

Let $W_p^{m,n}(I)$ be an image window or sub-image of image I of size $m \times n$ with an ij coordinate system positioned with its ij-location $(1, 1)$ at the reference location $p = (x, y) \in \Omega$, being the upper-left corner of the window.

In the following chapters we mainly work with grey-level images only. In the case of colour images we have a vector-valued image, and we use pixels $(x, y, u_1, \ldots, u_n)$ to indicate the image value for each of the n channels. See Fig. 3.1 for $n = 3$ and pixels (x, y, R, G, B).

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_3

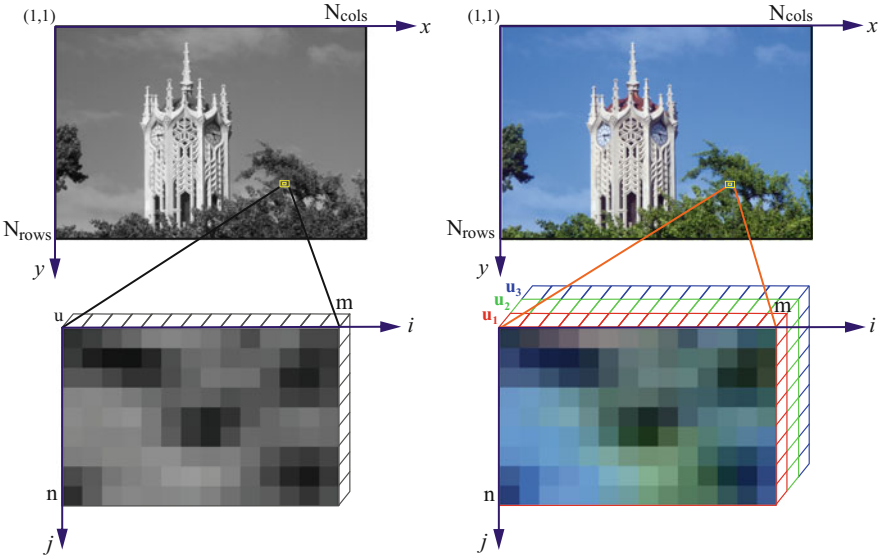

Fig. 3.1 A left-hand coordinate system, and a 14 × 9 sub-image (window) in the image `ClockTower`, shown in *grey-level* or *colour value* format, in the latter case with three indicated arrays for *Red, Green* and *Blue values*. *Top row*: Images in *xy* coordinates. *Bottom row*: Windows in *ij* coordinates

Mean and Variance Considering a grey-level image, we define the *mean* μ_I (i.e. the average grey-level) of image I as follows:

$$\mu_I = \frac{1}{|\Omega|} \sum_{(x,y)\in\Omega} I(x,y), \tag{3.2}$$

where $|\Omega| = N_{cols} \cdot N_{rows}$ is the cardinality of the carrier Ω of all pixel locations. We define the *variance* σ_I^2 of image I as

$$\sigma_I^2 = \left[\frac{1}{|\Omega|} \sum_{(x,y)\in\Omega} I(x,y)^2 \right] - \mu_I^2 \tag{3.3}$$

and σ_I as the *standard deviation* for the pixels of the given image.

Mode In a discrete distribution, such as for values of image pixels, the *mode* is the u-value with the highest number of counts on the carrier Ω, or the "most likely" value $I(p)$ in an image I.

The mean, variance, standard deviation, and mode can be defined analogously for a given region of interest or window $W_p^{m,n}(I)$ of an image I.

3.2 The Integral Image

For a given image I, the *integral image* I_{int}, popularised by the work of Viola and Jones in computer vision [269], is the sum of all pixel values in the image (or in a window) up to a given pixel location.

The Isothetic Case With reference to the assumed left-hand coordinate system in the carrier Ω, we define the value in I_{int} at pixel location $p = (x, y)$ by

$$I_{int}(p) = \sum_{(1 \le i \le x) \wedge (1 \le j \le y)} I(i,j), \tag{3.4}$$

i.e. as the summation of all pixel values in I to the left and above (x, y). See Fig. 3.2, left. Let $W_1, \ldots, W_4$ be the sum of intensity values in corresponding rectangular windows. This means that we have

$$\begin{aligned} I_{int}(p) &= W_1, \\ I_{int}(q) &= W_1 + W_2, \\ I_{int}(s) &= W_1 + W_4, \\ I_{int}(r) &= W_1 + W_2 + W_3 + W_4. \end{aligned}$$

Thus, the sum W_3 of all intensity values in the rectangle $pqrs$ equals

$$\begin{aligned} W_3 &= I_{int}(p) + I_{int}(r) - I_{int}(q) - I_{int}(s) \\ &= \sum_{\substack{1 \le i \le 15 \\ 1 \le j \le 6}} I(i,j) + \sum_{\substack{1 \le i \le 26 \\ 1 \le j \le 14}} I(i,j) - \sum_{\substack{1 \le i \le 26 \\ 1 \le j \le 6}} I(i,j) - \sum_{\substack{1 \le i \le 15 \\ 1 \le j \le 14}} I(i,j) \end{aligned} \tag{3.5}$$

in general, and for the special coordinates as illustrated in Fig. 3.2, left.

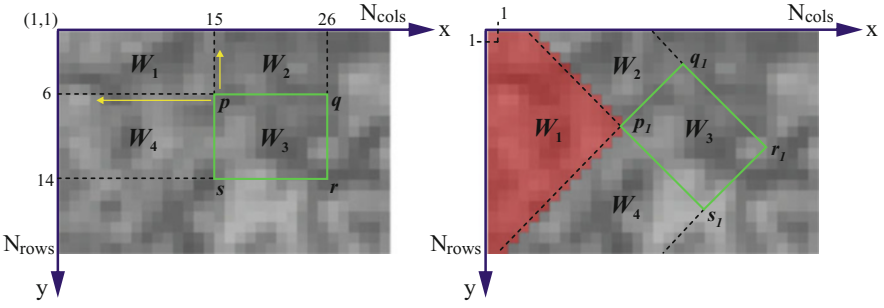

Fig. 3.2 Illustration of integral images and of the calculation of the sum W_3 of intensity values in *rectangular subimages*

Rotated Case Figure 3.2, right, also illustrates the case of an integral image defined for a rotation of I by $\frac{\pi}{4}$. In this case we have that

$$I_{\frac{\pi}{4}}(p) = \sum_{|x-i| \le y-j \ \wedge \ 1 \le j \le y} I(i,j) \tag{3.6}$$

for pixel location $p = (x, y)$; see [119]. The figure shows subimages (and the corresponding sums $W_1, \ldots, W_4$ of intensity values in those subimages) defined by pixel locations p_1, q_1, r_1, and s_1. Again, we have that $W_3 = I_{int}(p_1) + I_{int}(r_1) - I_{int}(q_1) - I_{int}(s_1)$, thus just a result of three basic arithmetic operations, independent of the actual size of the rectangular subimage.

The general principle applies analogously for rotations of I by an angle other than $\frac{\pi}{4}$, but the formulas for calculating values in the rotated integral images, replacing Eq. (3.6), will also make use of trigonometric functions in general.

Time Complexity Gain Having integral values at each pixel calculated in a preprocessing step in time $\mathcal{O}(N_{cols} N_{rows})$, and saved into an array of size $N_{cols} \times N_{rows}$, we can calculate the sum of intensities in a rectangular subimage of image I in constant time, independent of the actual size of the rectangular subimage, just by applying one addition and two subtractions. This is a very fast and cost-efficient operation needed repeatedly for real-time feature-based classification algorithms.

3.3 RGB to HSV Conversion

Occasionally we will make use of this colour space conversion. Figure 3.3 illustrates how an HSV cone (actually an idealized abstraction of layers of polygons) can be extracted from the common RGB cubic space of Red, Green, and Blue colour values. Let $0 \le R, G, B \le G_{\max}$.

Hue, Saturation, and Intensity Value Let hue H be scaled in the interval $[0, 2\pi)$, saturation S in $[0, 1]$, and the value (brightness or intensity) V in the common scale $[0, G_{\max}]$. Then we can convert an RGB colour space into the HSV space by calculating hue, saturation, and value as follows:

$$V = \frac{R + G + B}{3}, \tag{3.7}$$

$$H = \begin{cases} \delta & \text{if } B \le G \\ 2\pi - \delta & \text{if } B > G \end{cases} \quad \text{with} \tag{3.8}$$

$$\delta = \arccos \frac{(R - G) + (R - B)}{2\sqrt{(R - G)^2 + (R - B)(G - B)}} \quad \text{in } [0, \pi), \tag{3.9}$$

$$S = 1 - 3 \cdot \frac{\min\{R, G, B\}}{R + G + B}. \tag{3.10}$$

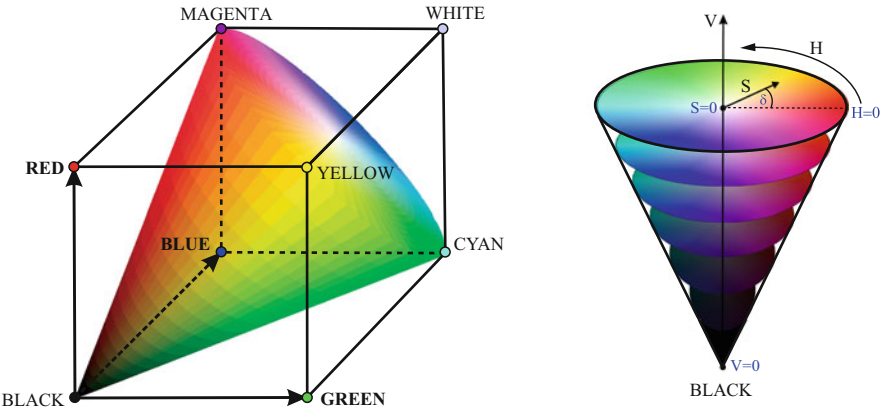

Fig. 3.3 RGB cubic colour space (*left*) and HSV conic colour space (*right*)

3.4 Line Detection by Hough Transform

Any line in a 2D Euclidean space can be represented with respect to xy Cartesian coordinates by using two parameters a and b as follows:

$$y = ax + b. \tag{3.11}$$

Using ρ and θ polar coordinates, as, for example, applied by Duda and Hart [54], a line can be represented by

$$\rho = x \cdot \cos\theta + y \cdot \sin\theta. \tag{3.12}$$

See Fig. 3.4, top.

Hough or Accumulator Space Hence, all the straight lines incident with any pixel location (x, y) in the carrier Ω can be represented either in the form of a straight line in ab parameter space or in the form of a sine-cosine curve in the $\rho\theta$ parameter space (also known as *Hough space*), thus increasing all the counters (accumulator values) in intersecting Hough space cells by 1. See Fig. 3.4, bottom. The ab parameter space would be unbounded in order to analyze lines in an image, thus preference is given to the $\rho\theta$ parameter space. For the $\rho\theta$ parameter space, bounds can be defined by $\rho \in [-d_{max}, d_{max}]$, for

$$d_{max} = \sqrt{N_{cols}^2 + N_{rows}^2} \tag{3.13}$$

and $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$.

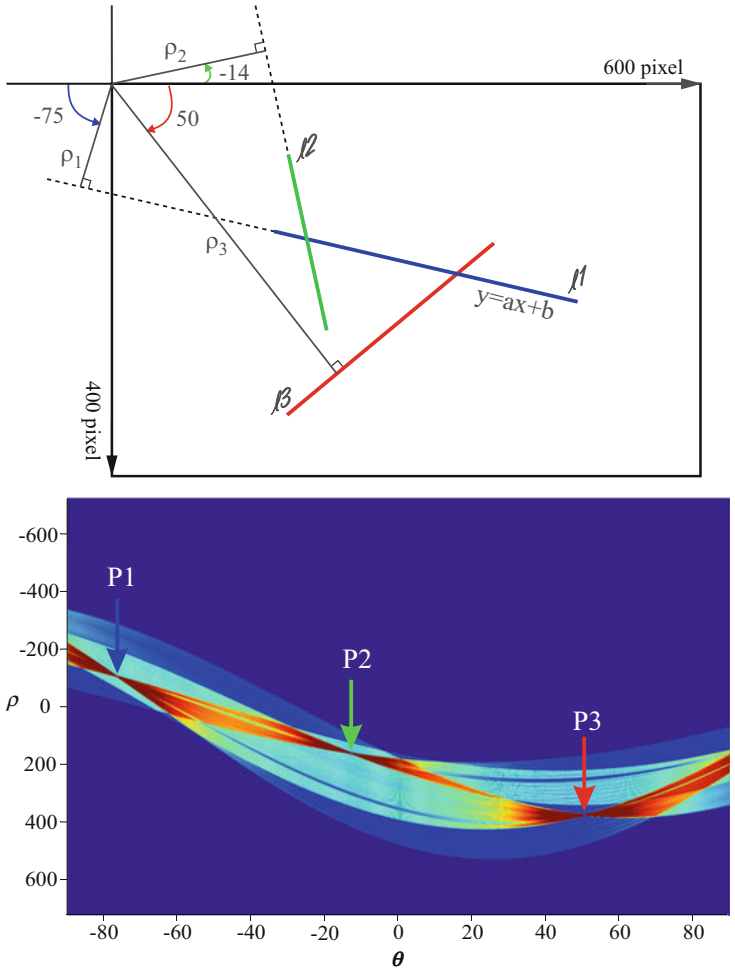

Fig. 3.4 *Top*: Line representation in polar coordinates. *Bottom*: Line detection in $\rho\theta$ Hough space; *blue* values illustrate low accumulator values, and *red values* show large accumulator values, with points P1, P2, and P3 being centres of defined clusters, corresponding to lines l_1, l_2, and l_3

Cluster Analysis in Hough Space We can conclude that set of all curves (each defined by one pixel location in the carrier Ω) which are incident with the same ρ and θ, identify one straight line in the image. The representation shown in Fig. 3.4, bottom, illustrates a Hough space after insertion of sine-cosine curves as defined by all the pixel locations on the three straight-line segments. Such a mapping of pixel locations into a parameter space is known as *Hough transform*. Straight lines are then finally detected by cluster analysis in Hough space.

3.5 Cameras

We assume to have an $X_wY_wZ_w$ *world coordinate system* which is not defined by a particular camera or other sensor, but rather by the position of a sensor when starting a process of sensor motion, a selected static reference in the scene, the position of a calibration object at a selected time, or a similar choice.

We also have a *camera coordinate system* $X_sY_sZ_s$ (index "s" for "sensor") defined for each of the used sensors. Here, Z_s is identified with the optic axis, and X_s and Y_s are assumed to be parallel to the x and y axes of the recorded image, respectively.

Euclidean Transform and Notations A camera coordinate system can be mapped by a *Euclidean transform* (i.e. an affine transform defined by rotation and translation) into the selected world coordinate system. A rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ need to be calculated for each time slot (i.e. each frame) while moving a camera through the world coordinate system of a 3D scene. A point in 3D space is given as $P_w = (X_w, Y_w, Z_w)$ in world coordinates, or (at a particular time slot) as $P_s = (X_s, Y_s, Z_s)$ in camera coordinates. Besides the coordinate notation for points we also sometimes use the vector notation, for example $P_w = [X_w, Y_w, Z_w]^T$ for point P_w.

Pinhole-type Camera The Z_s-axis models the *optic 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* f.

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:

$$x_u = \frac{f\,X_s}{Z_s} \quad \text{and} \quad y_u = \frac{f\,Y_s}{Z_s} \tag{3.14}$$

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

This intersection point (c_x, c_y) of the optical axis with the image plane (in xy image 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 Parameters 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 the 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 the optic axis of camera i and the 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 previously calibrated and does not need to be included 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 the 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 the camera coordinate system $X_s^i Y_s^i Z_s^i$ and the world coordinate system $X_w Y_w Z_w$.

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:

$$k \begin{bmatrix} x^i \\ y^i \\ 1 \end{bmatrix} = \begin{bmatrix} f^i/e_x^i & s^i & c_x^i & 0 \\ 0 & f^i/e_y^i & c_y^i & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^i & -[\mathbf{R}^i]^\top \mathbf{t}^i \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.15}$$

$$= [\mathbf{K}^i | \mathbf{0}] \cdot \mathbf{A}^i \cdot [X_w, Y_w, Z_w, 1]^\top , \tag{3.16}$$

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.

3.6 Stereo Vision and Energy Optimization

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, a 1-dimensional (1D) correspondence search can be limited to identical image rows.

Stereo Vision We address the detection of corresponding points in a stereo image pair $I = (L, R)$, which is a basic task for distance calculation in vehicles using binocular stereo.

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, sun strike, running wipers, and so forth) still pose unsolved problems. See Fig. 3.5 for an example of a depth map.

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* [119]. 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 $X_s Y_s Z_s$ camera coordinate system of the

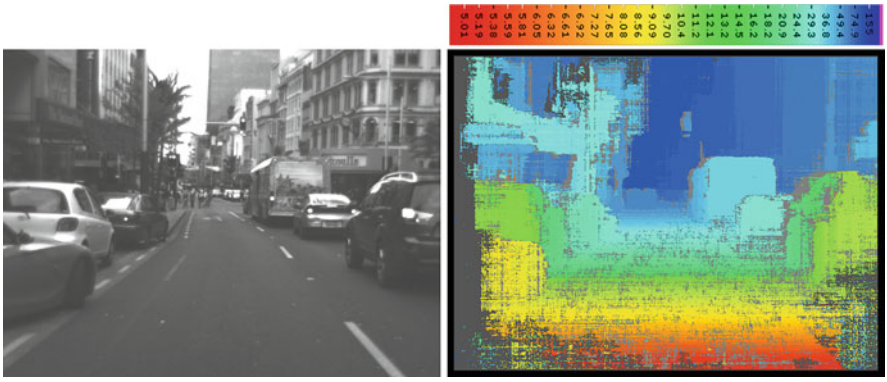

Fig. 3.5 *Left*: Image of a stereo pair (from a test sequence available on EISATS). *Right*: Visualization of a depth map using the *colour key* shown at the *top* for assigning distances in metres to particular colours. A pixel is shown in *grey* if there was low confidence for the calculated disparity value at this pixel (Courtesy of Simon Hermann)

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

$$p_u^L = (x_u^L, y_u^L) = \left(\frac{f \cdot X_s}{Z_s}, \frac{f \cdot Y_s}{Z_s} \right) \tag{3.17}$$

$$p_u^R = (x_u^R, y_u^R) = \left(\frac{f \cdot (X_s - b)}{Z_s}, \frac{f \cdot Y_s}{Z_s} \right) \tag{3.18}$$

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

$$[p_u^R]^\top \cdot \mathbf{F} \cdot p_u^L = 0 \tag{3.19}$$

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.

Optimizing a Labelling Function We specify one popular optimization strategy which has various applications in computer vision. In an abstract sense we assign to each pixel a *label* l (e.g. an optical flow vector $\mathbf{u}$, a disparity d, a segment identifier, or a surface gradient) out of a set L of possible labels (e.g. all vectors pointing from a pixel p to points with 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. Labels are assigned to all the pixels in the carrier Ω by a *labelling function* $f : \Omega \to L$.

Unfortunately, both the focal length and the labelling function are commonly denoted by the letter f. Because the context of a given text clearly identifies focal length or labelling, we decided to keep the symbol f in both cases.

Solving a labelling problem means to identify a labelling f which approximates somehow an optimum of a defined *error* or *energy*

$$E_{total}(f) = E_{data}(f) + \lambda \cdot E_{smooth}(f), \tag{3.20}$$

where $\lambda > 0$ is a weight. Here, $E_{data}(f)$ is the *data-cost term* and $E_{smooth}(f)$ the *smoothness-cost term*. A decrease of λ works towards a reduced smoothing of calculated labels. Ideally we search for an optimal (i.e. of minimal total error) f in the set of all possible labellings, which defines a *total variation* (TV).

We detail Eq. (3.20) by adding costs at pixels. In a current image, label $f_p = f(p)$ is assigned by the value of the labelling function f at pixel position p. Then we have that

$$E_{total}(f) = \sum_{p \in \Omega} E_{data}(p, f_p) + \lambda \cdot \sum_{p \in \Omega} \sum_{q \in A(p)} E_{smooth}(f_p, f_q), \tag{3.21}$$

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 (i.e. in the following image, or in the left or right image of a stereo pair); in this case we can also write $E_{data}(p, q)$ instead of $E_{data}(p, f_p)$.

Invalidity of the Intensity Constancy Assumption Data-cost terms are defined for windows which are centered at the considered pixel locations. The data in both windows, around the start pixel location p and around the pixel location q in the other image, are compared to assess "data similarity".

For example, in the case of stereo matching, we have $p = (x, y)$ in the right image R, and $q = (x + d, y)$ in the left image L, for disparity $d \geq 0$, and the data in both $(2k + 1) \times (2k + 1)$ windows are identical if, and only if, the data cost measure

$$E_{SSD}(p, d) = \sum_{i=-l}^{l} \sum_{j=-k}^{k} [R(x + i, y + j) - L(x + d + i, y + j)]^2 \tag{3.22}$$

results in value 0, where SSD stands for *sum of squared differences*.

The use of such a data-cost term would be based on the *intensity constancy assumption* (ICA), i.e. that intensity values around corresponding pixel locations p and q are (essentially) identical within a window of specified size. However, the ICA is unsuitable for real-world recording. Intensity values at corresponding pixels, and in their neighbourhoods, are typically impacted by lighting variations, or just by image noise. There are also impacts of local surface reflectance differences, differences in cameras if comparing images recorded by different cameras, or effects of perspective distortion (the local neighbourhood around a surface point

is differently projected into different cameras). Thus, energy optimization needs to apply better data measures compared to SSD, or other measures based on the ICA.

The Census Data-Cost Term The census cost function has been identified as being able to successfully compensate for brightness variations in input images of a recorded video [88, 91]. The *mean-normalized census-cost function* is defined by comparing a $(2l+1) \times (2k+1)$ window centred at pixel location p in frame I_1 with a window of the same size centred at a pixel location q in frame I_2. Let $\overline{I}_i(p)$ be the mean of the window around p, for $i = 1$ or $i = 2$. Then we have that

$$E_{MCEN}(p,q) = \sum_{i=-l}^{l} \sum_{j=-k}^{k} \rho_{ij} \tag{3.23}$$

with

$$\rho_{ij} = \begin{cases} 0 & I_1(p+(i,j)) < \overline{I}_1(p) \text{ and } I_2(q+(i,j)) < \overline{I}_2(q) \\ & \text{or } I_1(p+(i,j)) > \overline{I}_1(p) \text{ and } I_2(q+(i,j)) > \overline{I}_2(q), \\ 1 & \text{otherwise.} \end{cases} \tag{3.24}$$

Note that the value 0 corresponds to consistency in both comparisons. If the comparisons are performed with respect to values $I_1(p)$ and $I_2(q)$, rather than the means $\overline{I}_1(p)$ and $\overline{I}_2(q)$, then we have the *census-cost function* $E_{CEN}(p,q)$ as a candidate for a data-cost term.

Let $\mathbf{a}_p$ be the vector listing results $\text{sgn}(I_1(p+(i,j)) - \overline{I}_1(p))$ in a left-to-right, top-to-bottom order (with respect to the applied $(2l+1) \times (2k+1)$ window), where sgn is the signum function and $\mathbf{b}_q$ lists the values $\text{sgn}(I_2(q+(i,j)) - \overline{I}_2(q))$. The mean-normalized census data-cost $E_{MCEN}(p,q)$ equals the Hamming distance between vectors $\mathbf{a}_p$ and $\mathbf{b}_q$.

3.7 Stereo Matching

Stereo matching provides a procedure aiming at solving the optimization problem defined by Eq. (3.21) and the chosen specifications for the data and smoothness term. Due to time limitations, a particular stereo matcher, as used in practice, will only aim at providing a sub-optimal solution.

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 the xy image coordinates of the carrier Ω following the optimization approach as expressed by Eq. (3.21). 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 $E_{MCEN}(p,p^f)$ as defined in Eq. (3.23), and for the smoothness-cost term either the Potts model, linear truncated

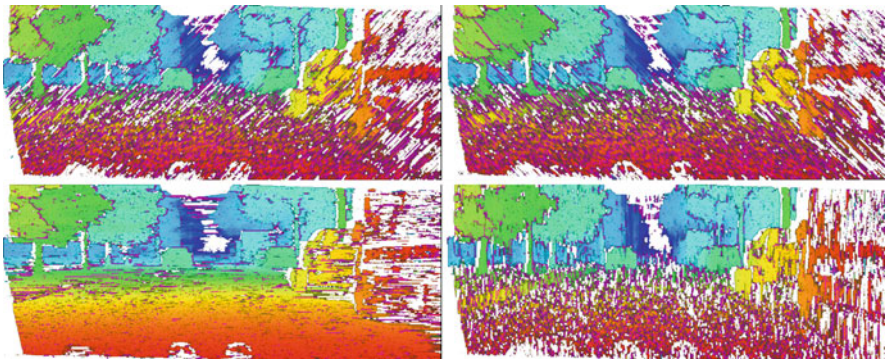

Fig. 3.6 Resulting disparity maps for stereo data when using only *one* scanline for DPSM with the SGM smoothness constraint and a 3×9 MCEN data cost function. *Bottom, left*; Left-to-right horizontal scanline. *Top, left*: Lower-left-to-upper-right diagonal scanline. *Bottom, right*: Top-to-bottom vertical scanline. *Top, right*: Upper-left-to-lower-right diagonal scanline. *White 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.)

cost, or quadratic truncated costs; see Chapter 5 in [119]. Chapter 6 of [119] also discusses different algorithms for stereo matching including *belief-propagation matching* (BPM) [247] and *dynamic-programming stereo matching* (DPSM). DPSM 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 [90].

A variant of SGM is used in Daimler's stereo-vision system, available since March 2013 in their Mercedes cars.

Iterative SGM (iSGM) is an example of a modification of a baseline 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 [89]. See Fig. 3.6, which also addresses confidence measurement; for a comparative discussion of confidence measures, see [83]. *Linear BPM* (linBPM) applies the MCEN data-cost term and the linear truncated smoothness-cost term [112].

Performance Evaluation of Stereo Vision Solutions Figure 3.7 provides a comparison of iSGM versus linBPM on four frame sequences each of length 400 frames. It illustrates that iSGM performs better (with respect to the used NCC measure) 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 `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, and it also highlights that vision systems in vehicles should be adaptive in general to ensure the best possible results under given circumstances.

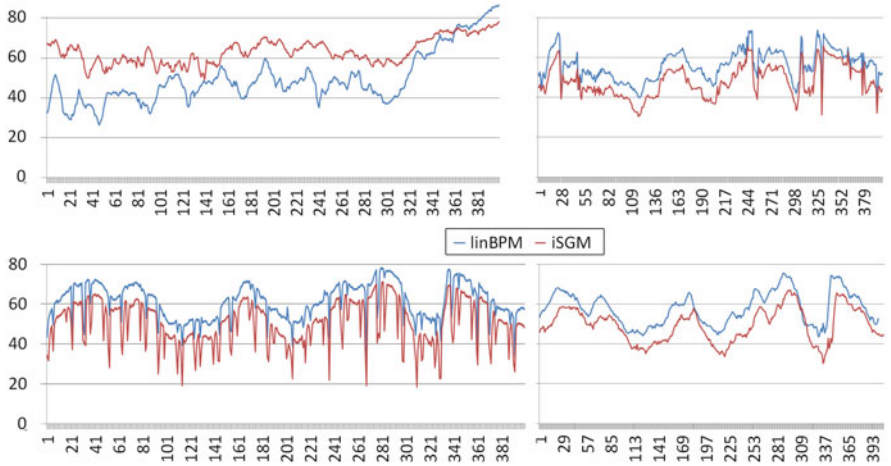

Fig. 3.7 NCC results when applying the third-eye technology for stereo matchers iSGM and linBPM for four real-world trinocular sequences of set 9 of EISATS (Courtesy of Waqar Khan, Veronica Suaste, and Diego Caudillo)

The diagrams in Fig. 3.7 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* [168] also uses masks so that only image values are compared which are close to step-edges in the third frame. It enables us to evaluate performance on any calibrated trinocular frame sequence (of any length), recorded in the real world.

Chapter 4
Object Detection, Classification, and Tracking

4.1 Object Detection and Classification

Object detection is the process of finding a specific object from a digital image or video based on prior knowledge represented by an object model, using a set of *features*, also called properties or parameters. For example, features can be a combination of edges, corners, or appearance templates. While object detection can be done instantly by a human, this can be a complicated task in computer vision [98].

The term *object classification* is sometimes mixed or confused with object detection, but classification is actually used in the case of two or multiple objects in a scene, where, for example, we need to distinguish the class of vehicles in a road scene from non-vehicle objects such as the classes of trees, buildings, or pedestrians.

There are various feature extraction methods for object detection as well as classification methods. In this chapter we mainly focus on classification techniques.

Combining Computer Vision with Machine Learning There are two kinds of classification techniques in the context of computer vision and pattern recognition, supervised classification and unsupervised classification techniques. Combining computer vision and machine learning is becoming more popular and one can hardly define a proper boundary for this as the current methods normally integrate both techniques to get the best results. We will perform a similar approach when needed in this book.

Supervised Techniques *Supervised classification* uses a supervised learning algorithm to create strong classifiers for a given object based on given training datasets, or from labelled classes of objects. The training dataset is prepared by an expert and it is in general a time consuming process to create and mark (label) an object in a large dataset.

After a training phase, a classifier accepts input images to detect the objects, or the classes of objects. Face detection, vehicle detection, and *optical character recognition* (OCR) are some of the common applications of supervised classification.

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_4

In this chapter we discuss three common supervised classification techniques, the *support vector machine* (SVM), the *histogram of oriented gradients* (HOG), and Haar-like feature-based classification. Decision trees, *k*-nearest neighbours (*k*NN), neural networks, and naive Bayes classifiers are other examples of supervised classification techniques.

Unsupervised Techniques In contrast to supervised techniques, where a training dataset has already been labelled by an expert, an unsupervised classification scheme may use segmentation or clustering techniques to build a model structure and relationships among the unlabelled data. In this chapter we also discuss two common unsupervised classification techniques, namely *k*-means clustering and Gaussian mixture models.

Performance Analysis Regardless of the applied technique, the performance of an object detector can be evaluated based on detection efficiency. Let TP denote the number of *true-positives*, or of *hits* when a classifier correctly detects an actual object. Also let FP be the number of *false-positives* or *misses*, i.e. when a classifier wrongly identifies a non-object as being an object. Similarly, we define *true-negatives* (TN) and *false-negatives* (FN) to describe the number of correct classifications of non-objects as being a "non-object", and the number of failures to detect objects, respectively.

Although we can always measure (i.e. count) the number FN, we cannot easily define the number TN for an application such as vehicle detection in a road scene. This is because the background of an image in essentially uncountable. Therefore, for performance evaluation of a classifier, we mainly rely on evaluations using TP, FP and FN.

Precision Rate and Recall Rate We define the *precision rate* (PR) and *recall rate* (RR) as follows:

$$PR = \frac{TP}{TP + FP} \quad \text{and} \quad RR = \frac{TP}{TP + FN}, \tag{4.1}$$

where *PR* is the ratio of true-positives to all detections by the detector, and RR is the ratio of true-positives to the total number of actual existing objects (i.e. the ground truth objects) in the analyzed frames.

ROC Curves Another way to visualize the performance of a classifier is a 2D *receiver operating characteristic curve* (*ROC-curve*) with FP and PR rates on its axes. In this book we use ROC-curves to compare the performance of our developed classifiers with other standard classifiers, under different conditions.

When Detected? By representing ground-truth locations as predefined *bounding boxes* around the objects, we consider a detection to be TP if the detected region has a sufficient overlap with the ground-truth bounding box, i.e. $a_0 \geq \tau$ for

$$a_0 = \frac{\mathscr{A}(D \cap T)}{\mathscr{A}(D \cup T)}. \tag{4.2}$$

The parameter a_0 is the overlap ratio, $\mathscr{A}$ denotes the area of the region that we consider for comparison, D is the detection bounding box, T is the ground truth bounding box, and τ is a threshold specifying the minimum for the expected matching rate. In this book, we consider $\tau = 0.80$ as a strict measure to confirm a detection D as contributing to TP.

4.2 Supervised Classification Techniques

In this section we discuss *supervised classification* techniques, which means that an expert defines the classes of objects (e.g. face, eye, vehicles), and also provides a set of sample objects for a given class, called a *training set*.

Regardless of the chosen classification technique (e.g. neural networks, decision trees, or nearest neighbour rule), there are two phases to the construction of a classifier: a *training phase* and an *application phase*.

Based on the provided training dataset, a classifier learns to use which sets of features, parameters, and weights are to be combined together in order to distinguish objects from non-objects. In the application phase, the classifier applies the already trained features and weights to an unknown query image to detect similar objects in the query image, based on what it has already learned from the training set.

4.2.1 The Support Vector Machine

The *support vector machine* (SVM) is a very common technique for object classification and pattern recognition, and generally can be categorized among supervised machine learning algorithms.

The method was proposed by Cortes and Vapnik in 1995; see [37]. It has shown a robust performance in solving various type of classification problems. The method is also considered to be a successor of previously defined neural network systems. We consider a binary classifier, with classification outputs $y = 1$ or $y = -1$, depending on an assigned class. We start by providing some basic definitions.

Separation by one Hyperplane Assume that we have various data items from a given dataset. The data items are shown as individual points in an n-dimensional space, where n is the number of features that we have. Figure 4.1 shows a 2D space and multiple data points, here "clearly" located in two separated clusters. Each data item is a point in $\mathbb{R}^n$ which can also be seen as a vector which starts at the coordinate origin. Figure 4.2 visualizes this concept for a 2D and 3D space. The goal of an SVM is to properly classify the data items (i.e. points or vectors) into specified classes.

A *hyperplane* is an $(n-1)$-dimensional subspace of $\mathbb{R}^n$. For the considered two-class scenario, an *ideal* hyperplane separates the data points (the vectors) into the intended classes so that the number of misclassifications is minimized, which can also be regarded as a *support* for the given vectors.

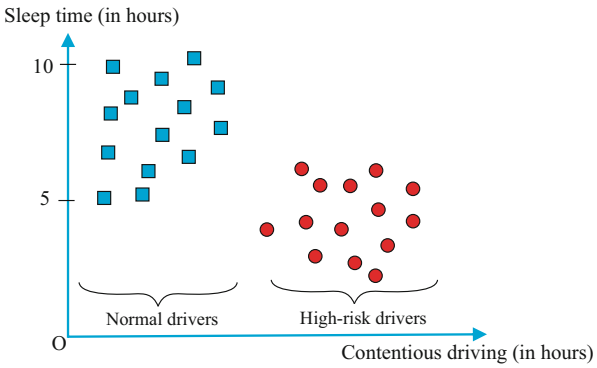

Fig. 4.1 2D data items (for two types of drivers) from a dataset shown as points

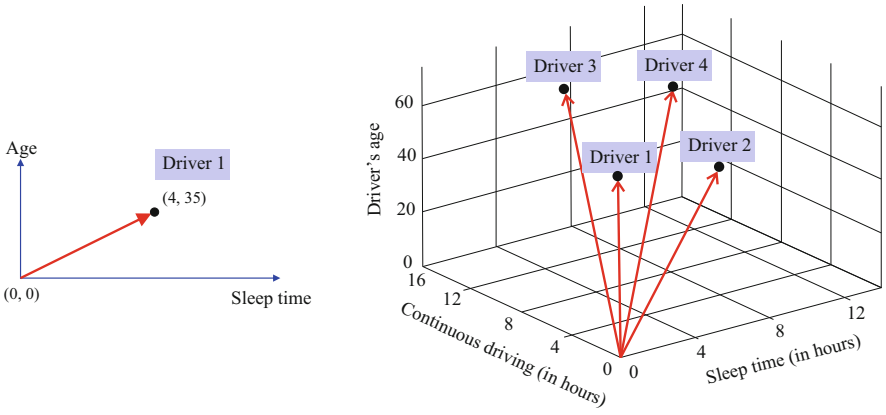

Fig. 4.2 2D (*left*) and 3D (*right*) data items (objects) from a dataset shown as vectors

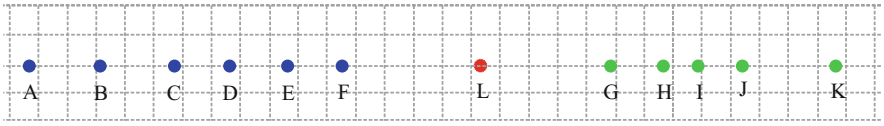

Fig. 4.3 A separating point as an ideal hyperplane for a 1D problem

For $n = 1$, a hyperplane is a point. In Fig. 4.3, point L defines an ideal hyperplane separating class $\{A, \ldots, F\}$ from class $\{G, \ldots, K\}$ (with zero misclassifications).

Margin For $n = 2$, a hyperplane is a line. In Fig. 4.4 all red, green, and blue lines are correctly classifying the blue and red data points. The red line may be judged as being the *best* of those three. Why? Because it provides the largest *margin* for both classes. The margin is the space between the two dashed lines, being symmetric to the considered red line. In such a situation, if more data points are introduced

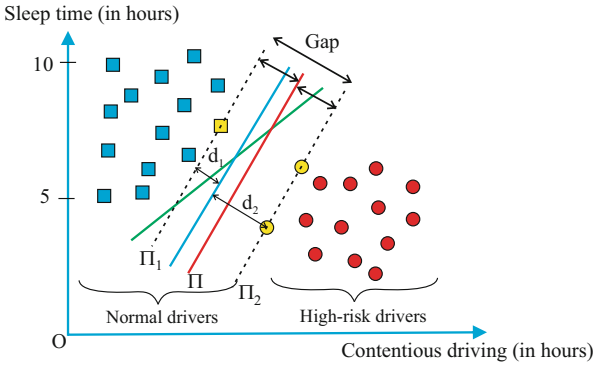

Fig. 4.4 Three different hyperplanes (*three lines*) to classify normal from high-risk drivers; the *blue* and the *red line* are assumed to be parallel, thus defining the same gap (i.e. space between both *dashed lines*); the *red line* is at a minimum distance from both *dashed lines* (since it is in the middle)

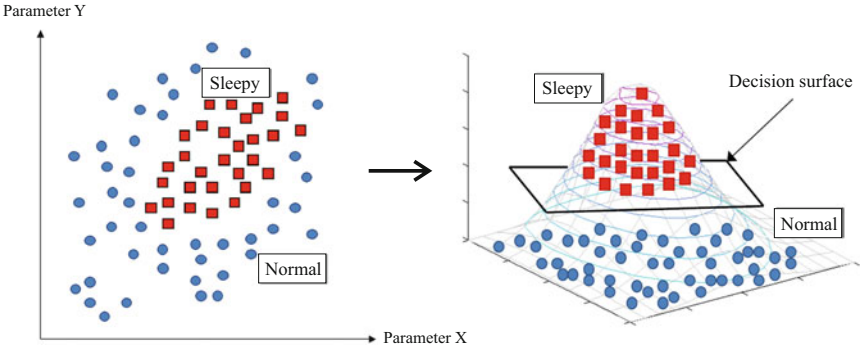

Fig. 4.5 Defining a separating plane (called the *decision surface* in the figure) by moving from $\mathbb{R}^2$ to $\mathbb{R}^3$

later, the risk of misclassification of blue and red data points is considered to be minimised if a separating hyperplane provides the largest margin.

Any hyperplane Π defining a linear separation of two classes also defines two parallel hyperplanes Π_1 and Π_2 which are the closest to Π and incident with at least one point of one of the two classes. The blue line in Fig. 4.4 has a small distance d_1 to the normal-driver data points, but a large distance d_2 to the high-risk driver data points. The margin, defined by two hyperplanes symmetric to the considered hyperplane and being between both clusters, would be of width $2d_1$ for the blue line, which is not as big as the margin by the red line. The red line has the largest margin of any ideal separation line since it is in the middle of the shown gap, and parallel to (say) line Π_2, which passes through two of the high-risk driver data points.

Adding an Augmenting Feature In Fig. 4.5, left, we cannot define a meaningful separation by one line to classify rectangular and circular data points. In such a case

we can consider adding a new feature (i.e. a third dimension) z, e.g. $z = x^2 + y^2$, or the relative distance to the centroid of all the given data points. In this way, the data points are mapped into the 3D space, and an SVM might be able to provide a reasonable (i.e. with respect to accuracy and margin) separation for classifying the data points. Fig. 4.5, right, illustrates a separating plane.

Accuracy versus Margin In general, ideal linear separation (i.e. zero error) of two classes is not possible by one hyperplane.

In Fig. 4.6 line B has a larger margin than line A; however, A is the best possible ideal separator (i.e. with zero error). Figure 4.7 shows a case where one of the stars (as an *outlier*) lies in the area which is supposed to be the red-dot region. In such a case, an SVM ignores the outlier and aims for a hyperplane that properly separates the majority of the data-points.

Any hyperplane separating the n-dimensional feature space into two half-spaces divides the data points into two clusters (being either correctly classified or not). Besides the margin, defined by the space between both clusters, now we also have to consider penalties for misclassification. To discuss this more in detail, we require some formal specifications.

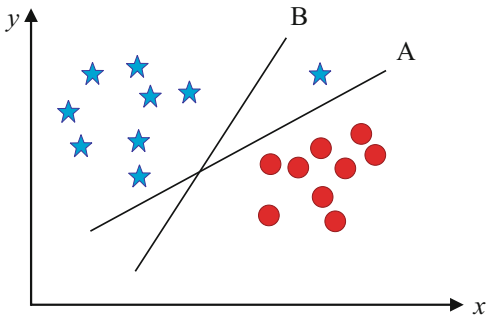

Fig. 4.6 Higher priority of accuracy versus maximizing the margin

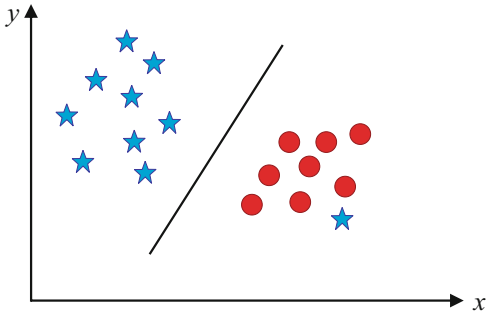

Fig. 4.7 Zero error is impossible in this case for one linear separator

Hyperplane Equations A line is given by $w_1x_1 + w_2x_2 + b = 0$ and a plane by $w_1x_1 + w_2x_2 + w_3x_3 + b = 0$. In general, a hyperplane in n-dimensional space is given by

$$\sum_{i=1}^{n} w_i x_i + b = 0 \tag{4.3}$$

or, by using vector notation, by

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]^\top, \quad \mathbf{x} = [x_1, x_2, \ldots, x_n]^\top, \quad \mathbf{w}^\top \cdot \mathbf{x} + b = 0. \tag{4.4}$$

The vector $\mathbf{w}$ defines the *normal* of the hyperplane, and the real number b is the hyperplane's offset from its origin (along the normal $\mathbf{w}$).

Any hyperplane divides $\mathbb{R}^n$ into two open half-spaces; a point $\mathbf{x} \in \mathbb{R}^n$ is either on the hyperplane or in one of the two half-spaces with either $\mathbf{w}^\top \cdot \mathbf{x} + b < 0$ (defining one of the two open half-spaces) or $\mathbf{w}^\top \cdot \mathbf{x} + b > 0$ (defining the other open half-space).

Maximizing the Margin We assume that the parameters in $\mathbf{w}$ and the value b are such that the hyperplane $\mathbf{w}^\top \cdot \mathbf{x} + b = 0$ defines the maximum margin, bordered by symmetric hyperplanes $\mathbf{w}^\top \cdot \mathbf{x} + b = +1$ and $\mathbf{w}^\top \cdot \mathbf{x} + b = -1$. The distance between two such symmetric hyperplanes equals $\frac{2}{||\mathbf{w}||_2}$, where

$$||\mathbf{w}||_2 = \sqrt{w_1^2 + \ldots + w_n^2}. \tag{4.5}$$

See Fig. 4.8. The separating line $\mathbf{w}^\top \cdot \mathbf{x} + b = 0$ is at distance $d_1 = d_2$ to both symmetrically positioned lines defined by the values -1 and $+1$, and we have

$$d_1 = d_2 = \frac{1}{\sqrt{w_1^2 + w_2^2}} \tag{4.6}$$

in this case where $n = 2$.

We note that maximizing the margin is defined in general, for any $n \geq 1$, by minimizing $||\mathbf{w}||_2$.

Penalties for Misclassifications Besides maximizing the margin, we also need to consider penalties for misclassifications. Let $\mathbf{w}^\top \cdot \mathbf{x} + b = 0$ be the considered hyperplane. Given data items $\mathbf{x}_i$, with $1 \leq i \leq m$, belong to one of the two assumed classes by being assigned either $y_i = +1$ or $y_i = -1$ (note: this is a case of supervised learning).

A correct classification of data item $\mathbf{x}_i$ by the chosen hyperplane $\mathbf{w}^\top \cdot \mathbf{x} + b = 0$ is given if, and only if,

$$y_i \left(\mathbf{w}^\top \cdot \mathbf{x}_i + b\right) \geq 1, \tag{4.7}$$

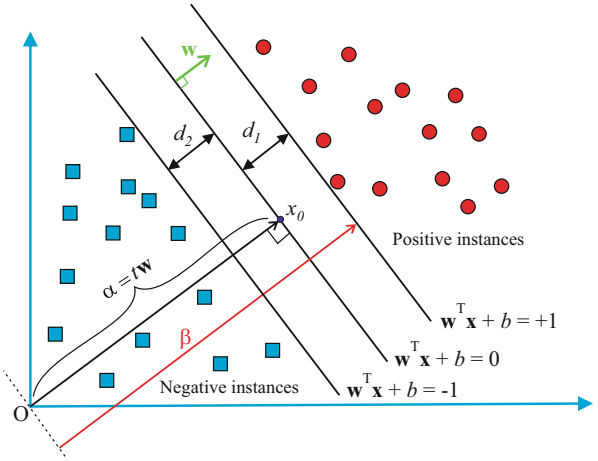

Fig. 4.8 Margin definition in case of a 2D feature space; note that $d_1 = d_2$ for the two symmetric lines defined by values -1 and $+1$

where we assumed that $y_i = -1$ if $\mathbf{x}_i$ is in the left half-space ($\mathbf{w}^\top \cdot \mathbf{x} + b = -1$), and $y_i = +1$ if it is in the right half-space ($\mathbf{w}^\top \cdot \mathbf{x} + b = +1$).

A penalty for a misclassification can now be defined by

$$1 - y_i \left(\mathbf{w}^\top \cdot \mathbf{x}_i + b\right). \tag{4.8}$$

Informally speaking, the further away a data item $\mathbf{x}_i$ lies from the considered hyperplane $\mathbf{w}^\top \cdot \mathbf{x} + b = -1$ or $\mathbf{w}^\top \cdot \mathbf{x} + b = +1$, the larger the value of $\mathbf{w}^\top \cdot \mathbf{x}_i + b$.

The Error Function Altogether we can define an error function $E(\mathbf{w}, b)$, to be minimized to find an optimum hyperplane for a given sample $[\mathbf{x}_i, y_i]$, for $i = 1, \ldots, m$, of pre-classified data items as follows:

$$E(\mathbf{w}, b) = \left[\frac{1}{m}\sum_{i=1}^{m} \max\left\{0, 1 - y_i\left(\mathbf{w}^\top \cdot \mathbf{x}_i + b\right)\right\}\right] + \lambda ||\mathbf{w}||_2, \tag{4.9}$$

where the parameter $\lambda > 0$ defines the weight given to the margin, compared to the sum of penalties for misclassifications.

Optimization Finding an optimum hyperplane means that we have to minimize the error function $E(\mathbf{w}, b)$. This is in general a non-trivial mathematical problem, and there are various solutions available, characterized by accuracy and time complexity (e.g. by using quadratic programming).

4.2.2 The Histogram of Oriented Gradients

The *histogram of oriented gradients* (HOG) is a robust feature descriptor proposed in 2005 by Dalal and Triggs [41]. Although the method was originally proposed for human detection, it is widely used for the detection of other objects as well.

A feature descriptor tries to find some common and distinct features in an object plus a model among the selected features such that these features, and the relationships between them, can be found in any "similar" images of the same object, where "similar" means changes in scale, rotation, or an affine transform.

The HOG feature descriptor partitions a window of assumed standard size into *cells*, clusters those into *blocks*, and calculates a descriptor (a vector) based on local gradient approximations for all the blocks. For a detected *candidate window*, it is scaled into the assumed standard size and a HOG descriptor is calculated. The created descriptors are then passed on to an SVM, or any other classifier, which classifies every window either as being, for example, "a human" or "not a human".

In order to support detection at various scales, the image will be sub-sampled into multiple sizes, therefore multiple classifications will be applied.

In the original paper, the training windows for human detection are of size 64×128 pixels, and partitioning cells are of size 8×8 (as shown for a sample image in Fig. 4.9). For all 64 pixels within an 8×8 cell, we estimate a *gradient vector*. We briefly recall gradient estimation.

Gradient Estimation For every pixel, we measure the intensity change in the x- and y-direction, and combine both into a vector called the *gradient vector*. (This approximation imitates mathematical approaches in continuous Euclidean space;

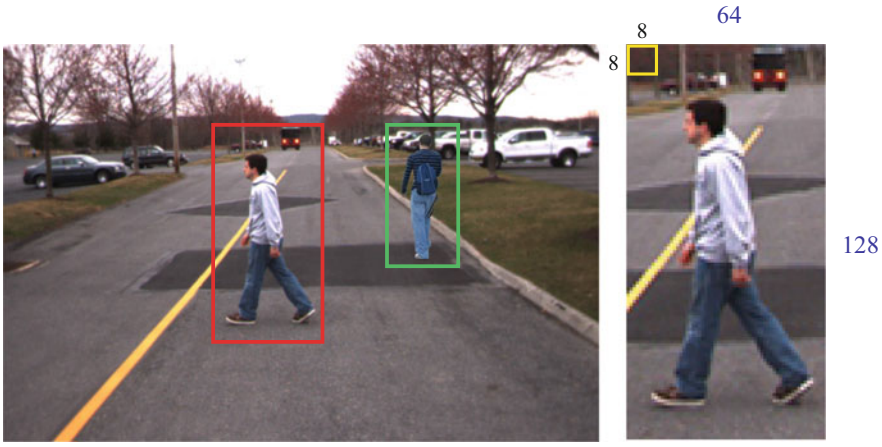

Fig. 4.9 *Left*: Two examples of *bounding boxes* for human detection in a traffic scene. *Right*: Sample of a training *bounding box* for human detection, scaled to the assumed standard size of 64×128 pixels. Such a window is then partitioned into *cells* of size 8×8 pixels (illustrated by one cell only in the *upper left*)

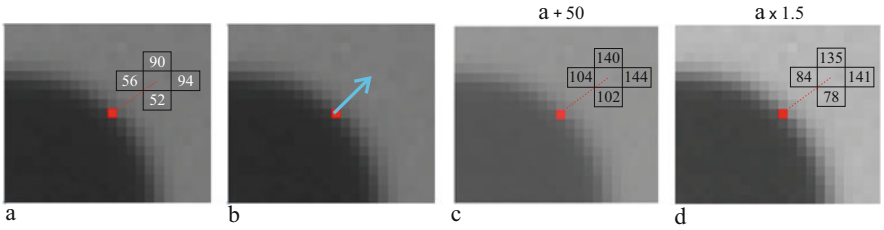

Fig. 4.10 Calculating a gradient vector and gradient magnitude for a single pixel in a scalar image (usually of image intensities). See the text for explanations

see, e.g. [119].) For example, for the given pixel location $p = (x, y)$ in Fig. 4.10a, we may calculate a gradient vector as follows, by only comparing adjacent pixel values $I(x-1, y)$ and $I(x+1, y)$, and $I(x, y-1)$ and $I(x, y+1)$:

$$\text{Gradient vector}: \quad \nabla I(p) = \begin{bmatrix} |94 - 56| = 38 \\ |90 - 52| = 38 \end{bmatrix}. \tag{4.10}$$

This is a simple method, thus time-efficient but easily influenced by image noise. (It can be enhanced by using more complex gradient estimators.) We can then calculate the magnitude and the angle of the gradient vector:

$$\text{Magnitude}: \quad |\nabla I(p)| = \sqrt{38^2 + 38^2} = 53.74. \tag{4.11}$$

$$\text{Angle}: \quad \arctan\left(\frac{38}{38}\right) = 0.78 \text{ rad } = 45°. \tag{4.12}$$

Figure 4.10b shows the gradient vector in the form of an arrow which has a length (magnitude) and a direction (angle).

Comments on Step-Edge Detection As an important property of the gradient vector at pixel location p, if there is a *step-edge* (i.e. an increase in intensity values along a path of pixels) at p in the image then this vector is perpendicular to the step-edge. A larger gradient magnitude corresponds to a "stronger" edge.

An interesting property of the gradient vector is that in the case of a uniform intensity change in the image, the intensity changes along the x- and y-axes are still the same as before. Figure 4.10c is the result when uniformly increasing (by adding 50) all intensity values shown in Fig. 4.10a. The gradient vector, and the magnitude of the gradient vector at the red pixel in Fig. 4.10c still remain the same. Figure 4.10d illustrates an increase in contrast. Here, the gradient vector still points in the same direction, but with an increase in magnitude. These examples illustrate that a gradient-based step-edge detection is basically intensity-invariant, with improved detection rate for higher-contrast edges.

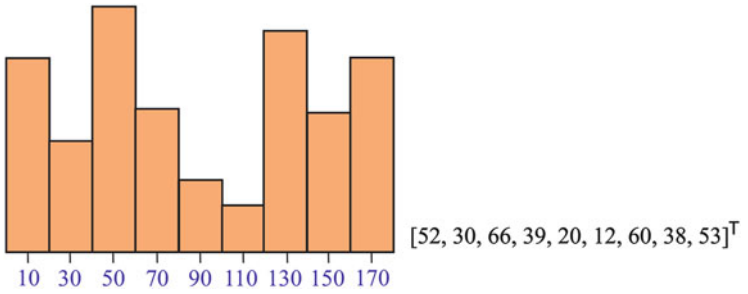

Fig. 4.11 *Left*: Sketch of a 9-bin histogram (unsigned case) for accumulated gradient magnitudes in one cell. *Right*: Corresponding vector for this histogram prior to any normalization (e.g. mapping into a unit vector)

Histogram of Gradient Magnitudes When calculating the 64 gradient vectors for the pixels in one cell, the magnitudes of these vectors are added in a 9-bin histogram based on their directions. The nine bins are created by partitioning the interval from 0° to 180° of unsigned gradient directions (or from 0° to 360° of signed gradient directions) uniformly into nine subintervals (e.g. intervals of 20° for each bin for the unsigned case). Figure 4.11, left, illustrates a histogram of accumulated gradient magnitudes (in nine bins) for one cell. Each calculated gradient vector *votes* with its direction for one of the nine bins, and then adds its magnitude to the previous sum of magnitudes in this bin. Figure 4.11, right, shows the vector of accumulated magnitudes (magnitudes of gradient vectors are counted in number of pixels) in those nine bins.

Each bin has one *defining direction*. In the unsigned case, these are directions 10°, 30°, ..., 170°; a calculated gradient vector may have any direction between 0° and 180°. To decrease aliasing, votes are generated by bilinear interpolation between defining directions of neighbouring bins. As a simple example, if we have 12 gradient vectors all calculated as having angle 25°, we give 75% of their total contribution in magnitudes (i.e. 9 votes) to the 30-degrees bin, and 25% (i.e. 3 votes) to the 10-degrees bin.

Use of Unit Gradient Vectors When discussing Fig. 4.10a, c, d, we saw that changing brightness or contrast around a pixel will not change the direction of the gradient vector, but may change the magnitude of this vector. By dividing the x- and y-components of a gradient vector by its magnitude, we obtain a vector of length one (i.e. a unit vector) without modifying the direction.

The use of unit vectors aims at invariance with respect to contrast in given images. This is a common choice when generating HOG descriptors. By this kind of vector normalization, each gradient vector contributes the same unit magnitude to the accumulated magnitudes in the nine bins of the generated cell histogram.

Use of Blocks of Cells In the HOG method as originally proposed by Dalal and Triggs, rather than using histograms of individual cells, the authors suggested

groupings of cells into blocks, and the corresponding concatenation of histograms of cells contained in one block. These blocks are not a partition of the image; they may overlap as illustrated in Fig. 4.12. Here each block consists of 2×2 cells with a 50% overlap with its adjacent block.

In the illustrated case of 2×2 cells in one block, histogram concatenation for four cells leads to a vector of 36 components (four times the values in nine bins each). It was suggested to first calculate individual cell histograms without normalization (i.e. using actual gradient vector magnitudes, not just unit vectors), and then to normalize by dividing all 36 components by the magnitude of the generated vector. Thus, this generates a unit vector with 36 components.

Figure 4.13 visualizes HOGs for two images. HOG values in each block are represented by a star-diagram of nine centred lines pointing in the defining direction of each of the nine bins, and having a length corresponding to the accumulated gradient magnitudes for its bin.

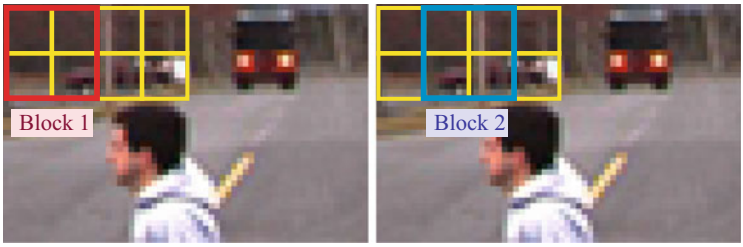

Fig. 4.12 Blocks of cells

Fig. 4.13 Two sample images and a visualization of their corresponding HOG outputs, assuming a subdivision of a standard-size window into 5 blocks horizontally, and 10 blocks vertically

HOG Descriptor After identifying a bounding box, being a candidate for the detection of an object (e.g. of a human), the bounding box is scaled into a window of the assumed standard size as used during learning, say of 64×128 pixels. In this case, a window is divided into 105 blocks (7 blocks horizontally and 15 blocks vertically). Each block has 4 cells, each with a 9-bin histogram, so 36 values for each block for the concatenated histograms. The HOG descriptor is now a vector having 3,780 components, defined by 105 blocks times 36 values in each concatenated histogram.

Note that the standard size of used windows, the organization of cells into blocks, and therefore the resulting HOG descriptor may differ for different classes of objects. Available bounding boxes, to be used as a training dataset, need to be scaled into the standard size of used windows.

Benchmark Databases In order to train a classifier based on HOG descriptors, a linear SVM classifier (as discussed before) can be generated for a given (e.g. human versus non-human) training dataset. For example, there are online datasets available for human versus non-human; see the MIT pedestrian database [189] or the challenging INRIA data set [96]. Single-hyperplane SVM classifiers have already proved to provide very good classifications for these two databases.

4.2.3 Haar-Like Features

Resembling local binary patterns as known from 1D, 2D, or higher-dimensional Haar-transforms,[1] Viola and Jones [269] introduced *Haar-like features* as a general tool for object detection in image analysis.

Haar Wavelets The following function is an example of a 1D *Haar wavelet* (see Fig. 4.14, left) defined for the interval $[0, 1]$, and extended with value 0 outside of this interval:

$$\psi(t) = \begin{cases} +1 & 0 \le t < 1/2, \\ -1 & 1/2 \le t < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

By visualizing, for example, the value -1 in black and the value $+1$ in white, and the value 0 not at all, this example would simply result in a white-black line of length 1. Figure 4.14, right, shows a black-white visualization of a 2D Haar wavelet defined on a square domain.

Haar-wavelets are local approximations of global basis functions of the Haar transform, similar to how local Gabor functions are local approximations of global basis functions of the Fourier transform.

[1] Named after the Hungarian mathematician Alfréd Haar (1885–1933).

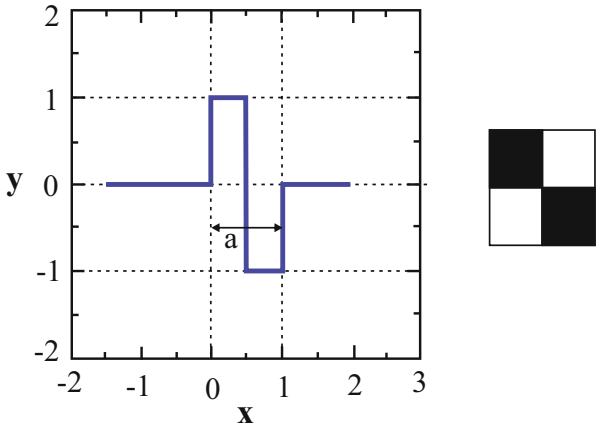

Fig. 4.14 *Left*: 1D Haar wavelet in diagram representation. *Right*: 2D Haar wavelet in *black-white* representation

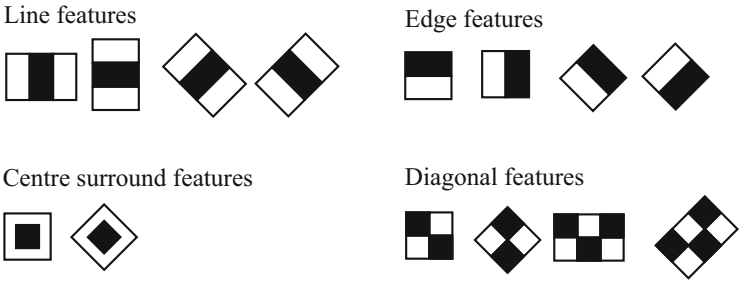

Fig. 4.15 Common types of Haar-like features

Haar-like Features Intensity distributions in gray-level or colour images are approximated by such 2D Haar wavelets, known as *Haar-like features*. (We recall that the mean $(R + G + B)/3)$ defines the intensity of a pixel in an RGB colour image.)

Viola and Jones proposed the use of rectangular Haar-like features, i.e. of patterns of adjacent black or white rectangles, originally with a focus on face detection. Samples of common Haar-like features are shown in Fig. 4.15.

Figure 4.16 illustrates matches of local intensity distributions with dark-bright (shown as black-white) patterns, illustrated by the two indicated Haar-like features. Figure 4.16 also shows two search windows which "contain" the two indicated Haar-like features. Search windows slide through a given image, and each search window may contain not only one but a small number (say two to five) Haar wavelets to be compared with the given intensity distributions for finding a good match. For a more detailed description, see [119]. A search window, characterized by the contained Haar wavelets, is scaled to different sizes before sliding through the given image. By

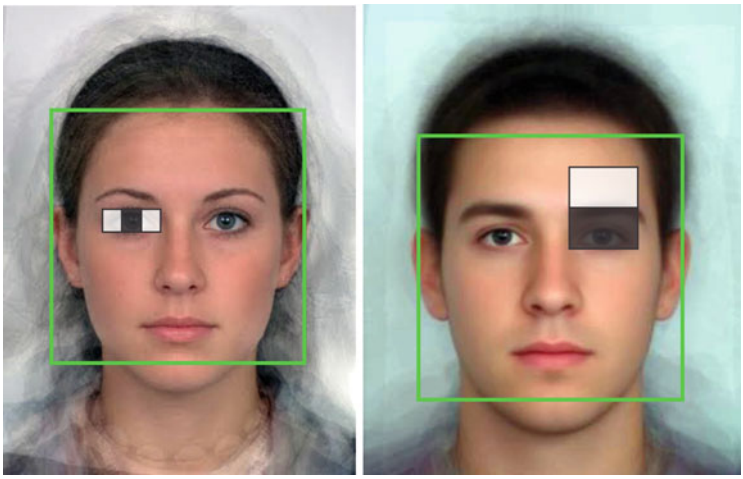

Fig. 4.16 Examples of Haar-like features in eye and forehead regions

evaluating the positioned Haar-like features, such a scaled search window generates a *weak classifier*.

Weak and Strong Classifiers A *strong classifier* is comprised of a series of *weak classifiers* (normally more than ten). A weak classifier itself includes a set of a few Haar-like features (normally two to five).

While a weak classifier acts only a little better than a random object classification (due to the lack of sufficient features to evaluate an object), a strong classifier should be able to detect objects at a high rate, such as 95% or more.

Figure 4.17 visualizes a *cascade of weak classifiers*, that all together make a strong classifier. The classifier starts with the first weak classifier by evaluating a search window at a given scale, also known as a *region of interest* (ROI).[2] It proceeds to the second stage (second weak classifier) if all the Haar-features in the first weak classifier match with the ROI, and so on, until reaching the final stage; otherwise the *search-region* under the sliding window will be rejected as being a non-object. Then the sliding window moves on to the next adjacent ROI. If all the Haar-features, within all the weak classifiers, successfully match with the ROI, then the region can be considered as being *marked* with a bounding box as being a *detected object*. Such a cascade defines an efficient way to create a strong classifier.

Training and Strong Classifier Definition The process of selecting appropriate Haar-like features for each weak classifier, and then for the whole strong classifier, constitutes the *training phase*, which can be done by using a machine learning technique such as *AdaBoost* [70]. This will be discussed in more detail in Chap. 5.

[2] A region of interest is commonly a rectangular sub-image, say of size $k \times l$, with $k \ll N_{rows}$ and $l \ll N_{cols}$, in which a weak classifier searches for an object.

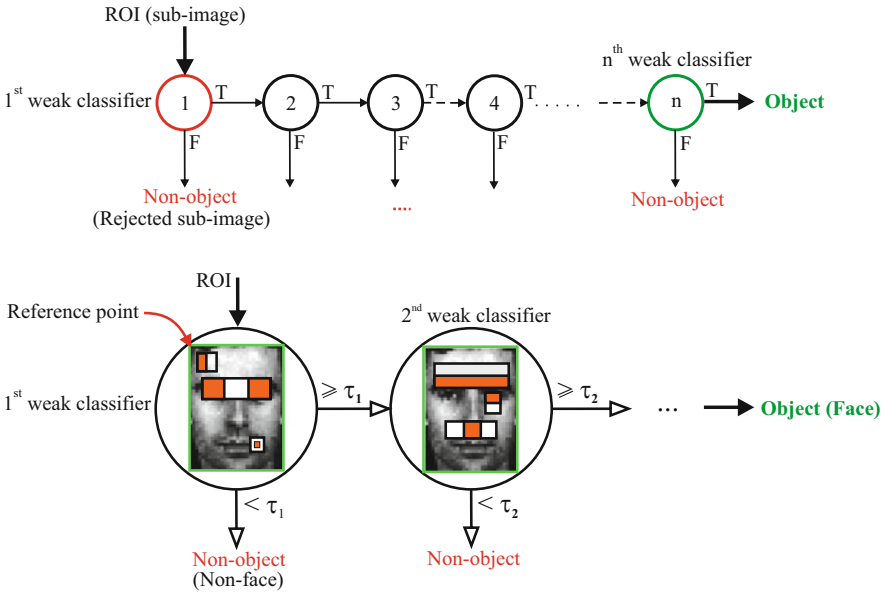

Fig. 4.17 A cascade of weak-classifiers that defines a strong classifier

As a final step in the training phase, the strong classifier is learned by selecting an order of the weak classifiers, each defined by specific Haar-features. These priority orders define a cascade as illustrated in Fig. 4.17. A resulting cascade depends upon a given object detection task and the considered weak classifiers (e.g. the selected Haar-features in the 1st and 2nd weak classifiers represented in Fig. 4.17 for face detection).

Application of the Learned Classifier Now we continue with the technical part of the *application phase*. For a given Haar-feature, defined (say) by three sub-rectangles, formalized by $\psi_i = (W_1, B_1, W_2)$, and illustrated in Fig. 4.18, we define the *Haar-feature value* as follows:

$$\mathscr{V}(\psi_i) = \omega_1 \cdot S_{W_1} + \omega_1 \cdot S_{W_2} - \omega_2 \cdot S_{B_1}, \tag{4.14}$$

where S_{W_1} and S_{W_2} are sums of image intensities in the white rectangles, and S_{B_1} is the sum of intensity values in the black rectangle (these sums are time-efficiently calculated as per the use of integral images described in Eq. (3.5)), and $\omega_i > 0$ are adjustable weights that depend on the context of the application.

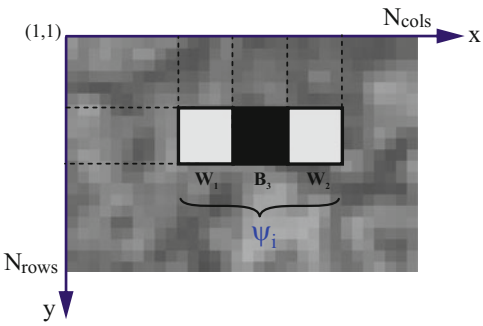

Fig. 4.18 Definition of a Haar-feature value

In the application phase, a Haar classifier tries to find matches of expected Haar-like features within the *sliding window*[3] to confirm or reject the existence of an object in the query region.

By convolution of Haar wavelets with the query image, the classifier tries to find those "adjacent" dark and bright regions in the image that closely match the appearance of given Haar features. Figure 4.16 illustrates a convolution of a line and an edge feature that matches the eyes and the forehead region. The figure indicates that, if the sliding window falls in an actual face region, we can expect some particular Haar-feature matches, as there is always a darker region of the eyes compared to the forehead region, and also a brighter region of iris compared to the eye sclera.

Match Function In order to find a match we define a *match function* using a parity $\zeta \in \{-1, +1\}$, and a threshold τ:

$$\mathscr{F}(\psi_p) = \begin{cases} +1 \text{ if } & \mathscr{V}(\psi_p) \geq \zeta \cdot \tau, \\ -1 & \text{otherwise.} \end{cases} \tag{4.15}$$

If $\mathscr{F}(\psi_p) = +1$ then we say that the given Haar-feature matches the given window (or sub-image), at the reference point p. The reference point could be any agreed point, for example, the top-left corner of a Haar-feature.

For every weak classifier we can define a mask $M = [\psi_1, \psi_2, \psi_3]$ that can include, for example, three Haar features (as shown in Fig. 4.17), each one in a particular position with respect to a reference point p (e.g. with respect to the top-left corner of the sliding window). We position the mask over the image I to check the similarity of the region with the mask. For every weak classifier, the similarity

[3]The sliding window, which defines the ROI, is a moving window starting from the top-left corner of an image, which moves over the image, from left to right, and top to bottom, in order to find feature matches in the query image. The sliding window starts with a small size (e.g. $k = l = 20$), and its size increases in each search iteration (up to the Max size of $N_{cols} \times N_{rows}$). The aim is to find feature matches for different window sizes, so ultimately detect any existing object, in any size, that falls in the region of a sliding window (or ROI).

of the mask M to the given image I is measured as a *weak-classifier value*:

$$\mathscr{D}(M_p) = \mathscr{F}(\psi_{1,p}) + \mathscr{F}(\psi_{2,p}) + \mathscr{F}(\psi_{3,p}). \tag{4.16}$$

Finally, a weak classifier produces "true" (see value T in Fig. 4.17), if the calculated value $\mathscr{D}(M_p)$ is greater than a pre-defined threshold:

$$\mathscr{H}(M_p) = \begin{cases} +1 \text{ if } & \mathscr{D}(M_p) \geq \tau_h, \\ -1 & \text{otherwise.} \end{cases} \tag{4.17}$$

In Chap. 5 we propose solutions and methodologies that enhance both the training and application phases of a Haar-based classifier, in order to achieve more effective object detections in the context of driver facial feature detection and vehicle detection.

4.3 Unsupervised Classification Techniques

In this section we introduce three selected unsupervised classification techniques, namely k-means clustering, Gaussian mixture models, and hidden Markov models, which are popular choices for object detection or object classification purposes.

4.3.1 k-Means Clustering

The standard k-means algorithm was proposed in 1957 by Stuart Lloyd [142]. Before publishing it at Bell Labs in 1982, E.W. Forgy had also published the same method in 1965 [67]. Thus, the method is also referred to as the *Lloyd–Forgy method*.

Clustering Clustering is a process that partitions a set of data points, a set of objects, or any kind of given items, into a number of subsets. For example, we can consider all the terms "stop sign", "parking sign", "no-entry sign", "speed sign", or "school sign" as identifiers for subsets (i.e. clusters) of the class of traffic signs. A clustering process may be based on either quantitative or qualitative properties of the given items.

Consider n data points $\mathbf{x}_i$, for $i = 1, \ldots, n$, that we want to partition into $k \leq n$ clusters. The goal is to assign one and only one cluster for each data point or data item. k-means tries to find optimized positions $\boldsymbol{\mu}_i$, for $i = 1, \ldots, n$, of *centroids* of the clusters which have a minimum distance to the data items in the corresponding cluster. To generate a family $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of clusters, $\boldsymbol{\mu}_i$ is the mean or centroid of the points in S_i.

Optimization Problem To formalize the minimization of distances, we can choose a particular metric d, and a particular way to compare differences in distances. The Euclidean or L_2-distance d_e is a common choice for d, with

$$d_e(\mathbf{x}, \boldsymbol{\mu}_i) = \|\mathbf{x} - \boldsymbol{\mu}_i\|_2 = \sqrt{\sum_{j=1}^{m} |x_j - c_j^i|^2} \tag{4.18}$$

when comparing data item $\mathbf{x} = [x_1, \ldots, x_m]^\top$ with centroid $\boldsymbol{\mu}_i = [c_1^i, \ldots, c_m^i]^\top$, assuming an m-dimensional space for the considered data items.

For comparing distances, a common choice is the sum of squared distances [287]. The goal is then to minimize the sum

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2 \tag{4.19}$$

by selecting k centroids to define k clusters

$$S_i = \{\mathbf{x} \in \bigcup \mathcal{S} : \|\mathbf{x} - \boldsymbol{\mu}_i\|_2 < \|\mathbf{x} - \boldsymbol{\mu}_j\|_2 \quad \text{for} \quad i \neq j \wedge 1 \leq j \leq k\}. \tag{4.20}$$

Finding optimum centroids is an NP-hard problem (i.e. informally speaking, finding k optimum centroids is a time-consuming task). Approximations may get stuck in a local minimum while aiming to move towards a global minimum.

A common k-means procedure is an iterative algorithm that gradually refines and converges to a sub-optimum clustering. After discussing the concept of the algorithm we also provide a numerical example.

STEP 1: Initialization. Select randomly an initial set of k means $\boldsymbol{\mu}_1^1$, $\boldsymbol{\mu}_2^1$, …, $\boldsymbol{\mu}_k^1$, e.g. in the set $\bigcup \mathcal{S}$ of all data items. The algorithm then proceeds iteratively by alternating between Steps 2 and 3 given below. The used superscript t shows the number of iterations, with $t = 1$ indicating the initial step.

STEP 2: Assignment Step. Assign each data item in $\bigcup \mathcal{S}$ to the cluster labelled by i where it has the minimum distance to the defining mean $\boldsymbol{\mu}_i^t$. This means that the set $\bigcup \mathcal{S}$ of data items is partitioned into Voronoi cells

$$S_i^t = \{\mathbf{x} \in \bigcup \mathcal{S} : \|\mathbf{x} - \boldsymbol{\mu}_i^t\|_2 \leq \|\mathbf{x} - \boldsymbol{\mu}_j^t\|_2 \quad \text{for} \quad i \neq j \wedge 1 \leq j \leq k\}, \tag{4.21}$$

where each data item $\mathbf{x}$ is assigned to one and only one cluster S_i^t; if there is an equal distance to two means $\boldsymbol{\mu}_i^t$ and $\boldsymbol{\mu}_j^t$ then we assign $\mathbf{x}$ to, say, the minimum of i and j.

STEP 3: Update Step. Compute new means

$$\boldsymbol{\mu}_i^{t+1} = \frac{1}{|S_i^t|} \sum_{\mathbf{x} \in S_i^t} \mathbf{x} \tag{4.22}$$

based on the data items in the current cluster, for $i = 1, \ldots, k$, where $|A|$ denotes the cardinality of a set A.

STEP 4: Iteration. Replace t by $t + 1$ and repeat Steps 2 and 3 until no further "significant" change happens to the centroids of the clusters.

Simple Example for Two Clusters Assume that the data items defined in Table 4.1 need to be clustered in $k = 2$ clusters. Figure 4.19, left, illustrates the choice of initial means as $\boldsymbol{\mu}_1^1 = [1, 1]^\top$ and $\boldsymbol{\mu}_2^1 = [2, 1]^\top$.

After this initialization, for each data item its distance is calculated to both centroids. Then we regroup them based on these distances [255]. Any data item that is closer to centroid $\boldsymbol{\mu}_1^1$ goes to Cluster S_1^1, and any data item that is closer to centroid $\boldsymbol{\mu}_2^1$ goes to Cluster S_2^1.

Let $\mathbf{D}_1$ be the distance matrix at Iteration 1. The first row of the matrix shows the Euclidean distance of the data items to $\boldsymbol{\mu}_1^1$, and the second row the Euclidean distance of the data items to $\boldsymbol{\mu}_2^1$. For example, the distance of data item 3 to mean

Table 4.1 Sample of four data items defined in a 2D feature space by property values x and y

Data item #	Property value x	Property value y
1	1.0	1.0
2	2.0	1.0
3	4.0	3.0
4	5.0	4.0

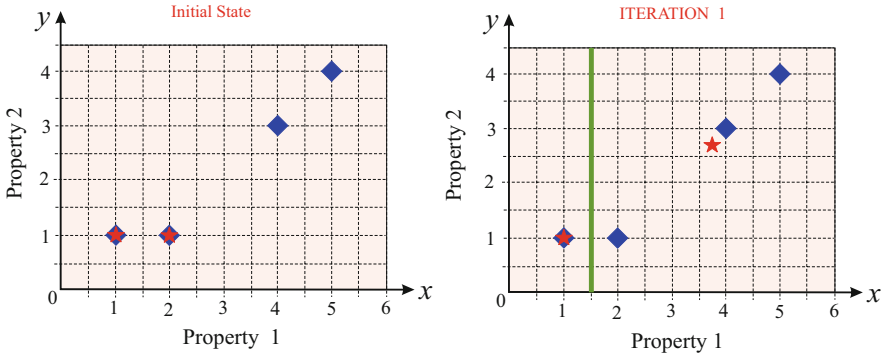

Fig. 4.19 Example for k-means clustering. *Left*: Initially selected centroids. *Right*: Updated centroids after the first iteration step

$\boldsymbol{\mu}_2^1$ equals $\sqrt{(4-2)^2+(3-1)^2} = 2.83$. We have that

$$\mathbf{D_1} = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix}. \tag{4.23}$$

The next step is grouping the data units into clusters S_1^2 and S_2^2 based on those distance values, represented by the matrix

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}. \tag{4.24}$$

The first row represents by value 1 items in cluster S_1^2, and the second row items in cluster S_2^2. Any item can only belong to one cluster. We obtain that $\boldsymbol{\mu}_1^2 = [1, 1]^\top$ (i.e. the same as before), but

$$\boldsymbol{\mu}_2^2 = \left[\frac{2+4+5}{3}, \frac{1+3+4}{3}\right]^\top = [1.38, 2.67]^\top. \tag{4.25}$$

Figure 4.19, right, shows the updated centroids. In the next iteration step we have

$$\mathbf{D}_2 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \tag{4.26}$$

and

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{4.27}$$

This leads to updated centroids

$$\boldsymbol{\mu}_1^3 = \left[\frac{1+2}{2}, \frac{1+1}{2}\right]^\top = [1.5, 1]^\top, \tag{4.28}$$

$$\boldsymbol{\mu}_2^3 = \left[\frac{4+5}{2}, \frac{3+4}{2}\right]^\top = [4.5, 3.5]^\top. \tag{4.29}$$

Figure 4.20 shows these updated centroids. Next we obtain that

$$\mathbf{D}_3 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.3 & 3.54 & 0.71 & 0.71 \end{bmatrix} \tag{4.30}$$

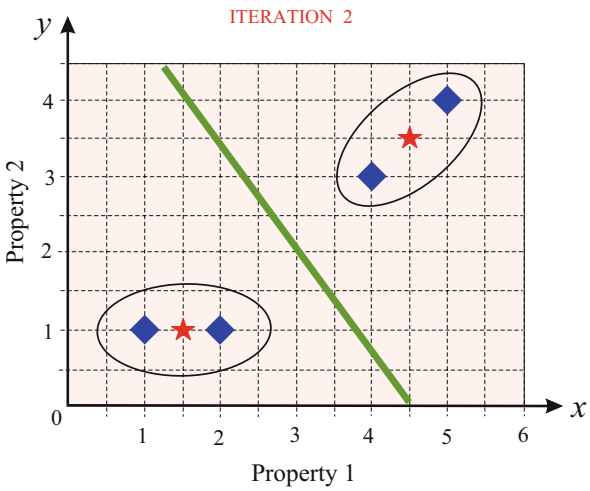

Fig. 4.20 *k*-means clustering; result of third iteration step

and

$$\mathbf{G}_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \tag{4.31}$$

There has been no change between $\mathbf{G}_2$ and $\mathbf{G}_3$, thus also no change in means. The process stops. The ellipses in Fig. 4.20 illustrate the final clustering. This is also the optimum clustering for this example.

For more complex examples of *k*-means clustering see, for example, [306].

4.3.2 Gaussian Mixture Models

In this section we follow the presentation in [19] by Bishop. Before starting a discussion on the concept of *Gaussian mixture models*, we review some of the prerequisites, including the Gaussian distribution and probability density functions.

The 1D Gaussian Distribution A common distribution of events in nature or technology is the *normal* or *Gaussian distribution*, named after the German mathematician Carl Friedrich Gauss. The Gaussian distribution is a unimodal distribution defined by a mean and a variance of events. The distribution graph is symmetric with respect to the mean. Figure 4.21 shows an example of a Gaussian distribution for a 1D event x. This distribution is formally defined by

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}, \tag{4.32}$$

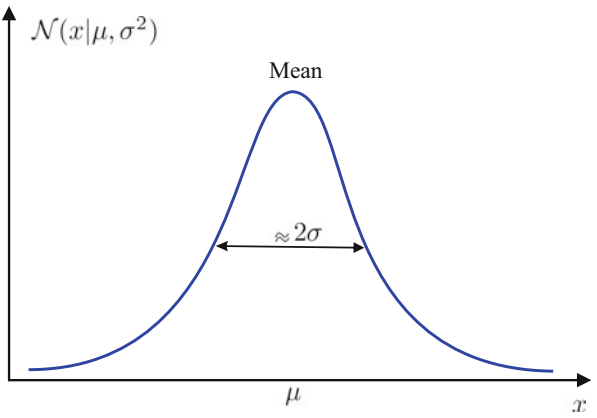

Fig. 4.21 Normal (Gaussian) distribution function

with two parameters: μ for the *mean* and σ^2 for the *variance*. The square root σ of the variance is the *standard deviation*. We have that $\mathcal{N}(x \mid \mu, \sigma^2) > 0$, for any real x, and the area below the distribution function (i.e. the integral of this function) equals 1. The function takes its maximum at $x = \mu$. The value σ controls the "slope" of the function; a smaller variance denotes a steeper and taller peak.

The Multi-dimensional Gaussian Distribution Having D-dimensional vectors $\mathbf{x}$ as studied events, their distribution can possibly be described by a multi-dimensional Gaussian distribution function as follows:

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{4.33}$$

where the mean $\boldsymbol{\mu}$ is here a D-dimensional vector; $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix that represents the covariance between components, and $|\boldsymbol{\Sigma}|$ is the determinant of the matrix $\boldsymbol{\Sigma}$. The components in the vector $\boldsymbol{\mu}$ represent the individual means of those components in the vectors $\mathbf{x}$, and $\boldsymbol{\Sigma}$ controls the slope of the Gaussians in the individual components.

When modelling random data (or more formally, random variables), besides the Gaussian distribution, there are also the Bernoulli distribution, the discrete uniform distribution, or Poisson distributions as possible alternative models.

Mixtures of Gaussian Distributions Events in nature, science or technology are often not clearly separated from each other. For example, when calculating an intensity histogram (i.e. of gray-levels) in image analysis, the histogram will rarely be similar to a Gaussian distribution, but it may look similar to a combination of a small number of Gaussian distributions, as sketched in Fig. 4.22.

Such linear combinations of Gaussian distributions can define a probabilistic model for a given process (e.g. the value distributions in gray-level images in a given field of application). These combinations are known as *mixtures of distributions*. By combining only a small number of Gaussian distributions, with properly adjusted means, variances, and linear coefficients when forming the sum, more complex distribution functions can be approximated.

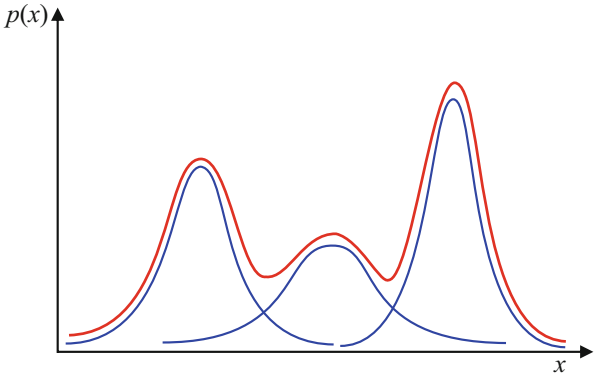

Fig. 4.22 Example of a mixture of three 1D Gaussian distributions

Therefore, we can assume a combination of K Gaussian density functions to form the following equation:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{4.34}$$

which is called a *mixture of Gaussians*.

Each of the Gaussian density functions $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a *mixture component* having its own mean value $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. We call the parameters π_k the *mixing coefficients*.

Considering the requirement that $p(\mathbf{x}) \geq 0$ and since $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a probability distribution, the parameters π_k should satisfy the two conditions:

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad 0 \leq \pi_k \leq 1. \tag{4.35}$$

Mixture models can also consist of a linear combination of other common distributions, such as a mixture of Bernoulli distributions.

Now we consider a random variable $\mathbf{z}$ (a K-dimensional binary vector) in which a specific element $z_k = 1$ but the rest of the elements are equal to 0. In short, $z_k \in \{0, 1\}$ and $\sum_z z_k = 1$. We also specify $p(\mathbf{x}, \mathbf{z})$ as the joint distribution function, in terms of $p(\mathbf{z})$ or the marginal distribution, as well as a conditional distribution $p(\mathbf{x} \mid \mathbf{z})$ corresponding to the given representation in Fig. 4.23.

Now we define $p(\mathbf{z})$ as the marginal distribution over $\mathbf{z}$, with mixing coefficients π_k, in which $p(z_k = 1) = \pi_k$ and the parameters $\{\pi_k\}$ satisfy the conditions mentioned in (4.35).

As $\mathbf{z}$ uses a 1 to K representation, a similar approach can be considered for the distribution:

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{4.36}$$

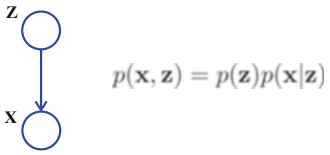

Fig. 4.23 Graphical representation of a mixture model, in the form of a joint distribution $p(\mathbf{x}, \mathbf{z})$

Similarly, the conditional distribution of $\mathbf{x}$ given a specific $\mathbf{z}$ is a Gaussian:

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4.37}$$

which we can also write in the form:

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \tag{4.38}$$

Considering the given joint distribution of $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, we can define the marginal distribution of $\mathbf{x}$ by summation of the joint distribution, for all states of $\mathbf{z}$:

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{4.39}$$

which is obtained via the product of Eqs. (4.36) and (4.38).

If we have several data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and we represent the marginal distribution as $p(\mathbf{x}) = \sum_z p(\mathbf{x}, \mathbf{z})$, then for any data item $\mathbf{x}_n$, there should be a corresponding latent variable $\mathbf{z}_n$.

Having a homologous formulation for a Gaussian mixture, based on the latent variable, enables us to work on the joint distribution $p(\mathbf{x}, \mathbf{z})$ rather than $p(\mathbf{x})$ (the marginal distribution). Later we will see that this will lead to a significant simplification by representation of the expectation-maximization, or EM algorithm.

In addition, the conditional probability of $\mathbf{z}$ given $\mathbf{x}$ performs a major role in this context. We denote by $\gamma(z_k)$ the conditional probability $p(z_k = 1|k)$ and the value of $\gamma(z_k)$ can be calculated using Bayes' theorem:

$$\begin{aligned} \gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) &= \frac{p(z_k = 1)\, p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)\, p(x|z_j = 1)} \\ &= \frac{\pi_k\, \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j\, \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \end{aligned} \tag{4.40}$$

We view π_k as the a-priori probability for $z_k = 1$. We also consider the value of $\gamma(z_k)$ as the corresponding posterior probability, after we have observed the data item $\boldsymbol{x}$.

Maximum Likelihood Assume we have a set of observed data items $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and we want to model the set of data items using a mixture of Gaussians. Representing the data items as an $N \times D$ matrix $\mathbf{X}$ in which the nth row shows the elements $\mathbf{x}_n^T$, we can similarly represent the corresponding latent variables in the form of an $N \times K$ matrix $\mathbf{Z}$ in which the nth row shows the elements $\mathbf{z}_n^T$.

Considering Eq. (4.34) the log likelihood function can be represented as follows:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} . \tag{4.41}$$

We need to find a solution to maximize this function. In other words, we seek parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, and $\boldsymbol{\pi}$ for which the Gaussian distribution best fits the data, i.e. the adjusted Gaussian is most likely to be the "true" distribution of the data.

Maximization of the likelihood function is a complex task compared to the case where we have only a single Gaussian [19]. This is due to the summation over k in Eq. (4.41).

To cope with this, we consider an alternative method, called the *expectation-maximization* algorithm, which is a very common approach in this regard.

Expectation-Maximization (EM) EM is considered to be a very powerful technique for finding the maximum likelihood using models such as the Gaussian mixture model with latent variables. The method is proposed by Dempster et al. [46] and McLachlan and Krishnan [158].

We begin with the conditions required to maximize the likelihood function. If we set the derivative of $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ in Eq. (4.41) equal to zero, with respect to $\boldsymbol{\mu}$, we have:

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \Sigma_k(\mathbf{x}_n - \boldsymbol{\mu}_k). \tag{4.42}$$

We should mention that the posterior probabilities (also called responsibilities) given by Eq. (4.40) naturally appear on the right-hand side of the equation. Multiplying by $\boldsymbol{\Sigma}_k^{-1}$ and rearranging the equation, we have:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{n_k}) \mathbf{x}_n, \tag{4.43}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{n_k}). \tag{4.44}$$

N_k can be interpreted as the number of effective data items assigned to cluster k. Similarly, by setting the derivative of $p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ equal to 0, with respect to $\boldsymbol{\Sigma}_k$. We can follow a similar argument to maximize the likelihood of the covariance matrix with a single Gaussian:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{n_k})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}. \tag{4.45}$$

Finally, using a similar approach, we can maximize the $p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ again by setting the derivative equal to 0, this time with respect to the mixing coefficient π_k, where we already knew that the sum of the mixing coefficients is equal to 1. This can be obtained by using a Lagrange multiplier to maximize the quantity below:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right), \tag{4.46}$$

which gives

$$0 = -\sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda. \tag{4.47}$$

Multiplying both sides of the equation by π_k and summing over k using the constraints in (4.35) we can find that $\lambda = -N$. By eliminating λ and rearranging the equation, we have:

$$\boldsymbol{\pi}_k = \frac{N_k}{N}. \tag{4.48}$$

We mention that the results in Eqs. (4.43), (4.45) and (4.48) cannot simply lead to a closed-form easy solution to define the mixture model's parameters, as the responsibilities $\lambda(z_{n_k})$ are directly dependent on the parameters of the model in a complicated way that go through Eq. (4.40). However, we can follow an iterative approach to solve the maximize likelihood problem. For the Gaussian mixture model, first we can consider some initial and preliminary values for means $\boldsymbol{\mu}$, covariances $\boldsymbol{\Sigma}$, and mixing coefficients $\boldsymbol{\pi}$, followed by alteration and update between two steps called expectation (E) and maximization (M).

Here, we summarize the EM approach in four steps as follows:

1. **Initialization**: Choose initial values for means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$, and mixing coefficients $\boldsymbol{\pi}_k$ and perform the evaluation of the first value of the log likelihood.

2. **Expectation**: Calculate the responsibilities based on the current parameter values:

$$\gamma(z_{n_k}) = \frac{\pi_k \, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{4.49}$$

3. **Maximization**: Perform re-estimation of the same parameters $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, $\boldsymbol{\pi}_k$ based on the current responsibilities:

$$\boldsymbol{\mu}_k^* = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{n_k}) \mathbf{x}_n, \tag{4.50}$$

$$\boldsymbol{\Sigma}_k^* = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{n_k}) (\mathbf{x}_n - \boldsymbol{\mu}_k^*)(\mathbf{x}_n - \boldsymbol{\mu}_k^*)^{\mathrm{T}}, \tag{4.51}$$

$$\boldsymbol{\pi}_k^* = \frac{N_k}{N}, \tag{4.52}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{n_k}). \tag{4.53}$$

4. **Evaluation**: Evaluate the log likelihood based on the new parameters' values

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{4.54}$$

and check for the convergence of either log likelihood or the parameters. If convergence has been met, stop the algorithm, otherwise iterate from Step 2.

4.4 Object Tracking

Typically, an object detection algorithm such as a Haar-like classifier can locate the objects anywhere within the input image. However, there are many cases of temporary object occlusion, as well as impossibility of object detection, e.g. due to a harsh sunlight, reflections, back-lights, or strong shades.[4]

[4]Different to a shadow, which is a silhouette cast by an object that blocks the source of either an indoor (e.g. candle) or outdoor light (e.g. sun), a shade is "darkness" that only applies to outdoor

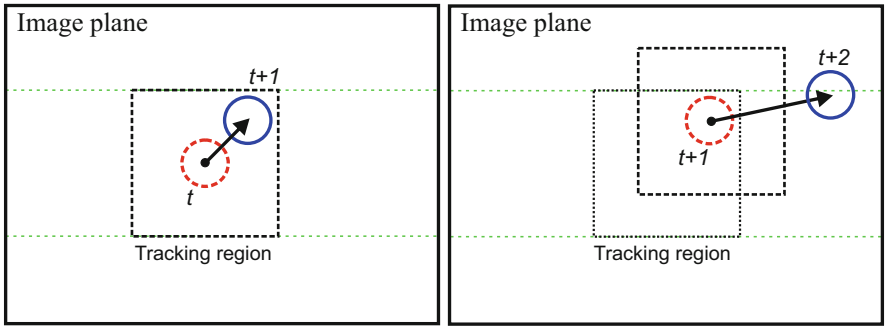

Fig. 4.24 Failure in face tracking if the driver moves fast away more than expected

Furthermore, even in the case of ideal conditions and the possibility of easy object detection, it may suffice that we only search for the object in a specific region of the image, instead of searching in the whole image plane. This can be considered with respect to the temporal information, and the location and size of a detected object in previous frames.

A simple and intuitive method for predicting an object's location is to search in a limited region of interest around the last identified position of the object. However, as illustrated in Fig. 4.24, the solution may easily fail in four cases:

- If the object's movement is faster than expected;
- If the input frame rate is low, it is possible to miss movements between frame t and $t + 1$;
- If the feature detection process takes a considerable time, we may encounter a large shift between two subsequent preprocessed frames;
- If the given tracking region is not large enough to cover the natural dynamics of the object movement.

On top of the above limitations, we have the issue of *noisy images* for our application of driver monitoring under challenging outdoor lighting conditions.

There are various tracking filters, but some of the most common object tracking techniques include:

- Mean shift
- Continuously adaptive mean shift (CAM shift)
- Kanade–Lucas–Tomasi (KLT)
- Kalman filtering

applications such as the shade underneath a tree, or the shade underneath a car. Of course, a car itself can also have a shadow. In our application, there are many challenges for driver's face or vehicle detection, due to existing outdoor shades.

Depending on our application any of the above tracking techniques could be a tracker of choice. For example, Kalman filtering can be a good choice if we are dealing with noisy input data. In the following pages we provide some further details about each filter.

4.4.1 Mean Shift

Mean shift, proposed by Cheng in 1998 [30], is an algorithm to locate the maxima of a density function based on existing discrete data samples from that function. The generalization of the method resembles a k-means clustering algorithm. This method is also appropriate for mode seeking, probability density estimation, and tracking.

In this section we follow the presentation in [113]. Consider a set S consisting of n data items $\mathbf{x}_i$ in a D-dimensional Euclidean space X. Let $K(\mathbf{x})$ denote a kernel function that indicates the contribution of $\mathbf{x}$ to estimate the mean value. We then define the model mean $\mathbf{m}$ for the data set $\mathbf{x}$ and a kernel function K:

$$\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i)}. \tag{4.55}$$

We call the difference $\mathbf{m}(\mathbf{x}) - \mathbf{x}$ the *mean shift*.

In this method we iteratively move to the mean of the data points, and after each iteration we set $\mathbf{x} \leftarrow \mathbf{m}(\mathbf{x})$.

The algorithm stops when it converges with no further change [e.g. $\mathbf{m}(\mathbf{x}) = \mathbf{x}$].

We refer to the $\mathbf{m}, \mathbf{m}(\mathbf{x}), \mathbf{m}(\mathbf{m}(\mathbf{x})), \ldots$ sequence as the $\mathbf{x}$ *trajectory*.

If the sample means have already been calculated for multiple data items, then we also need to perform an iteration on all the points, simultaneously. Generally, the kernel K is a function of $\|\mathbf{x}\|^2$ that can be represented as follows:

$$K(\mathbf{x}) = k(\|\mathbf{x}\|^2) \tag{4.56}$$

where we call k the *profile* of K with the following properties:

- k is non-negative;
- k is non-increasing: $k(x) \leq k(y)$ if $x > y$;
- k is a continuous variable and we have $\int_0^\infty k(x)dx < \infty$.

There are two types of kernels that are commonly used in mean shift (see Fig. 4.25): The flat kernel:

$$K(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \|\mathbf{x}\| \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.57}$$

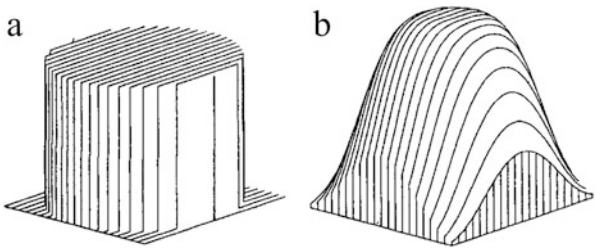

Fig. 4.25 (**a**) Flat kernel. (**b**) Gaussian kernel (Image source [30])

and the Gaussian kernel:

$$K(\mathbf{x}) = \exp(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}). \tag{4.58}$$

Now the main question is how to estimate the density function based on the given set of data samples, which can be scattered and sparse. A possible approach can be smoothing data, for example by convolving data with a kernel with width h, or a kernel with radius h.

Parzen's window method is a common technique used for probability estimations. In this method, for a dataset including n data items $\mathbf{x}_i$ in a D-dimensional space, we estimate the kernel density using a kernel $K(\mathbf{x})$ with radius h as follows:

$$\widetilde{f}_K(\mathbf{x}) = \frac{1}{nh^D} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \tag{4.59}$$

$$c = \frac{1}{nh^D} \sum_{i=1}^{n} k\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right). \tag{4.60}$$

We can assess the accuracy of the kernel density estimator by calculating the mean square error between the estimated density and the actual density.

To minimize the mean square error we can use the Epanechnikov kernel as follows:

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2C_D}(D+2)(1-\|\mathbf{x}\|^2) & \text{if} \quad \|\mathbf{x}\| \leq 1, \\ 0 \qquad \text{otherwise}, \end{cases} \tag{4.61}$$

where C_D represents the volume of the D-dimensional sphere [113], with the following profile:

$$k_E(x) = \begin{cases} \frac{1}{2C_D}(D+2)(1-x) & \text{if} \quad 0 \leq x \leq 1, \\ 0 \qquad \text{if} \quad x > 1. \end{cases} \tag{4.62}$$

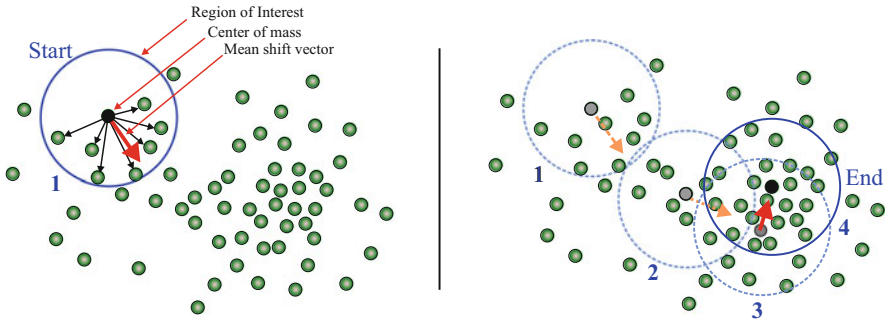

Fig. 4.26 *Left*: Initialization of the mean shift kernel, calculation of the mean shift based on summation of data point vectors. *Right*: Trajectory of the algorithm and the final optimum position, with the highest density of data points

Figure 4.26, left, shows the initial location of the kernel, and Fig. 4.26, right, shows the moving trajectory of the mean shift algorithm towards the most dense location of the data point, in four iterations, and finally the algorithm has converged in the best position, where the maximum density is achieved.

4.4.1.1 Mean Shift Tracking

The basic idea of tracking using mean shift was proposed by Comaniciu et al. [34]. The idea is to use the mean shift algorithm to model an object based on colour probability density. For example, we can track an object in a video by colour probability matching of the object model with the target in the video. This can be done using mean shift to estimate the colour probability, therefore matching part of the image frame as the target object.

In brief, we have three major steps to be taken in mean shift tracking. First, modelling the desired object based on its colour probability density. Second, evaluating the potential target in a search window, based for its colour probability; and third, matching the target candidate with the object model using mean shift.

Object Model: Let $\mathbf{x}_i, i = 1, \ldots, n$, represent the locations of the pixels in our model which are centred at 0. We consider an m-bin colour histogram to represent the colour distribution. $b(\mathbf{x}_i)$ represents the colour bin for the colour at $\mathbf{x}_i$. Assuming a normalized size of the model, the radius of the kernel is $h = 1$. Let's define q as the probability of the colour $u, u = 1, \ldots, m$, within the object model, as follows:

$$q_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i\|^2)\, \delta(b(\mathbf{x}_i) - u), \tag{4.63}$$

where C is the normalization constant:

$$C = \left[\sum_{i=1}^{n} k(\|\mathbf{x}_i\|^2) \right]^{-1}. \tag{4.64}$$

The kernel profile k denotes a contribution weight by distance to centroid, and δ:

$$\delta(a) = \begin{cases} 1 & \text{if } a = 0, \\ 0 & \text{if otherwise,} \end{cases} \tag{4.65}$$

represents the Kronecker delta function. In other words, if $b(\mathbf{x}_i) = u$ the kernel $k(\|\mathbf{x}_i\|^2)$ contributes to q_u

Target candidate: Similar to the object model, let $\mathbf{y}_i, i = 1, \ldots, n$, denote the pixel locations of targets which are centred at 0. We also define p as the probability of colour u in the target candidate:

$$p_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\| \frac{\mathbf{y} - \mathbf{y}_i}{h} \right\|^2 \right) \delta(b(\mathbf{y}_i) - u), \tag{4.66}$$

where C is the normalization constant

$$C_h = \sum_{i=1}^{n_h} \left[\sum_{i=1}^{n_h} k\left(\left\| \frac{\mathbf{y} - \mathbf{y}_i}{h} \right\|^2 \right) \right]^{-1}. \tag{4.67}$$

Color Density Matching: At this stage we measure the similarity of the target at location $\mathbf{y}$ with the colour probability of p with the object model q. For this purpose, mean-shift tracking uses the Bhattacharyya coefficient, which is indicated by ρ in the following equation:

$$\rho(p(\mathbf{y}), q) = \sum_{u=1}^{m} \sqrt{p_u(\mathbf{y}) q_u}, \tag{4.68}$$

where ρ is the cosine of vectors $[\sqrt{p_1}, \ldots, \sqrt{p_1}]^\top$ and $[\sqrt{q_1}, \ldots, \sqrt{q_m}]^\top$. Larger values of ρ represent a better colour probability match between the model and the target candidate.

Let us consider $\mathbf{y}$ as the current location of the target, with colour probability $p_u(\mathbf{y}) > 0$ for $u = 1, \ldots, m$. Also let $\mathbf{z}$ define the new estimated location of the target near $\mathbf{y}$, with no significant change in its colour probability, compared to its previous location.

Using a Taylor series expansion, we have:

$$\rho(p(\mathbf{z}), q) = \frac{1}{2} \sum_{u=1}^{m} \sqrt{p_u(\mathbf{y}) q_u} + \frac{1}{2} \sum_{u=1}^{m} p_u(\mathbf{z}) \sqrt{\frac{q_u}{p_u(\mathbf{y})}}. \tag{4.69}$$

Updating the second term of Eq. (4.69) by (4.66), we have:

$$\rho(p(\mathbf{z}), q) = \frac{1}{2}\sum_{u=1}^{m}\sqrt{p_u(\mathbf{y})q_u} + \frac{C_h}{2}\sum_{i=1}^{n_h}\omega_i k\left(\left\|\frac{\mathbf{z}-\mathbf{y}_i}{h}\right\|^2\right), \tag{4.70}$$

where the weight ω_i is

$$\omega_i = \sum_{u=1}^{m}\delta(b(\mathbf{y}_i) - u)\sqrt{\frac{q_u}{p_u(\mathbf{y})}}. \tag{4.71}$$

To maximize the $\rho(p(\mathbf{z}), q)$ we should only consider maximizing the second term of Eq. (4.70) as the first term is $\mathbf{z}$-independent.

Given q_u as the model, and also $\mathbf{y}$ as the target location in the previous frame, we can summarize the mean shift algorithm as the following steps:

1. Initialize the target location as $\mathbf{y}$ in the current frame.
2. Compute $p_u(\mathbf{y})$ for $u = 1, \ldots, m$. Also compute $\rho(p(\mathbf{y}), q)$.
3. Compute the weights $\omega_i, i = 1, \ldots, n_h$.
4. Apply the mean shift and calculate $\mathbf{z}$ (the new location):

$$\mathbf{z} = \frac{\sum_{i=1}^{n_h}\omega_i g\left(\left\|\frac{\mathbf{y}-\mathbf{y}_i}{h}\right\|^2\right)\mathbf{y}_i}{\sum_{i=1}^{n_h}\omega_i g\left(\left\|\frac{\mathbf{y}-\mathbf{y}_i}{h}\right\|^2\right)}, \tag{4.72}$$

   where $g(x) = -k'(x)$. In this step, we consider a window of pixels $\mathbf{y}_i$ and the size of the window is related to h.
5. Compute $p_u(\mathbf{z})$ for $u = 1, \ldots, m$. Also compute $\rho(p(\mathbf{z}), q)$.
6. While the relation $\rho(p(\mathbf{z}), q) < \rho(p(\mathbf{y}), q)$ is true, set $\mathbf{z} \leftarrow \frac{1}{2}(\mathbf{y} + \mathbf{z})$. This is intended for validation of the new location of the target.
7. If $\|\mathbf{z} - \mathbf{y}\|$ is close to zero (very small), then the algorithm has converged; so stop. Otherwise, set $\mathbf{y} \leftarrow \mathbf{z}$ and repeat from Step 1.

4.4.2 *Continuously Adaptive Mean Shift*

While mean shift considers a fixed colour distribution, the CAMSHIFT (*continuously adaptive mean shift*), proposed by Bradski [21], is adaptive to dynamic colour distribution changes due to lightning, illumination, and depth changes.

The algorithm adapts the size of the search window and also calculates the colour distribution for the search window. The search window is calculated as follows. First we calculate the zeroth moment or mean in the window W:

$$M_{00} = \sum_{(x,y)\in W} I(x,y). \tag{4.73}$$

Then we calculate first moments of x and y:

$$M_{10} = \sum_{(x,y)\in W} xI(x,y), \qquad M_{01} = \sum_{(x,y)\in W} yI(x,y). \tag{4.74}$$

Thus we have the search window centred at:

$$x_c = \frac{M_{10}}{M_{00}} \qquad y_c = \frac{M_{01}}{M_{00}}. \tag{4.75}$$

After defining the above concepts and equations the CAMSHIFT algorithm can be performed in the following order:

Step 1: Define any initial or preliminary location for the search window.
Step 2: Apply the mean-shift tracking method using the revised search window technique.
Step 3: Save the zeroth moment.
Step 4: Set the window size depending on the zeroth moment.
Step 5: Repeat Steps 2 and 4 until there are no further changes (i.e. convergence).

4.4.3 The Kanade–Lucas–Tomasi (KLT) Tracker

Feature tracking is essential in many computer vision applications from object tracking to 3D reconstruction and optical flow. The high performance of robust tracking is crucial in order to get better results in higher-level algorithms such as visual odometry in visual navigation and driver assistance systems. In this section, the *KLT (Kanade–Lucas–Tomasi)* feature tracker is explained. The goal of KLT is to track a set of feature points in an image sequence. This tracker is based on the early work of Lucas and Kanade in 1981 [145]; it was expanded in 1991 by Tomasi and Kanade [256], and later was clearly explained in the paper by Shi and Tomasi [235].

The idea behind proposing KLT was to answer two fundamental questions. First, how do we choose the best features for tracking? Secondly, how do we track those selected features from the current frame to the next frame?

To answer the first question, Lucas and Kanade [145] introduced a method to register left and right images for the application of stereo matching. The main objective is to minimize the sum of squared intensity errors between a previous and current window.

Having a small inter-frame motion has some advantages. The first benefit is the approximation of the current frame by using the translation of the past frame. Apart from that, the image intensities of the current frame can be written as the intensities of the past frame together with a residue term which depends on the translation vector. The question regarding the "good features" was unanswered in [145]; the concept of what "good features" are is of interest to many researchers in the area. The proposed methods were independent of the registration algorithm. In KLT, "good features" are those that can be tracked well.

KLT is proposed to deal with different problems of traditional image registration techniques, which are generally costly and unable to handle rotation, scaling and shearing. Spatial intensity information is used to search for the position of the best-matched features. Briefly, best-matched features can be extracted by utilizing the minimum eigenvalue of each 2×2 gradient matrix. This can be a denoising Sobel or Sharr operator. Then, in order to track the selected features, the Newton–Raphson method is used to minimize the difference between two windows.

Feature Point Tracking Image intensity may change frame by frame in an image sequence. To describe the algorithm let's assume $I(x, y, t)$ represents an image sequence where x and y are the space variables, and t is considered as a time variable:

$$I(x, y, t + \tau) = I(x - \varepsilon, y - \eta, t), \tag{4.76}$$

where $t + \tau$ represents the time shift for the next frame. This is identified by translation of every pixel in the current frame t by an appropriate motion vector $\mathbf{d} = [\varepsilon, \eta]^\top$. $\mathbf{d}$ represents the displacement of the point $X = (x, y)$ between time frame t and $t + \tau$, as shown in Fig. 4.27. The properties in Eq. (4.76) are sensitive, even in the case of a static environment under constant lighting. Moreover, when the brightness of different frames is changing, the photometric appearance of the region of interest on a visible surface will also be modified.

In contrast, Eq. (4.76) is unperturbed by surface markings, as well as the areas which are far from occluding contours. When there is an abrupt change in the image intensities, the changing pixels can still be well defined, even with minor variations

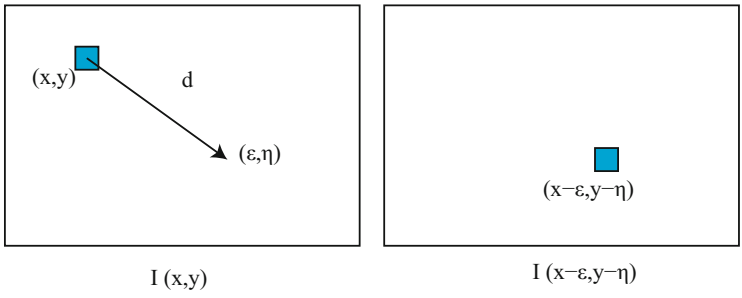

Fig. 4.27 The displacement vector of an image

in overall brightness. Surface markings contribute both to dense shape results and good motion estimations.

Finding the displacement vector **d** from one frame to another is a challenging issue in tracking a single pixel, unless it has suitable brightness compared to the neighbouring pixels. Based on the fact that a pixel can easily be confused with its adjacent pixels due to the existence of noise, tracking of that pixel in the next frame is a difficult task when only local information is exploited. Consequently, KLT does not consider tracking of single pixels, but windows of pixels which contain acceptable texture information. A problem has arisen when considering different behaviours of the pixels within a window; when the intensity pattern in the window warps from one frame to another. In addition, the pixels can even disappear when the window is along the occluding boundary.

By considering these concerns, Lucas and Kanade used residue monitoring to overcome the problem of changing the contents over time [145]. It continuously checks the appearance of the query window and if a significant change is detected, then the algorithm discards the window. In order to combine different velocities to measure the displacement of the window, Lucas and Kanade modelled the changes in the form of a complex transformation, similar to an affine transform map. So different pixels or part of the query window can have different velocities. These concerns lead to an important disadvantage, where the system may suffer from an over-parametrization that may outweigh the advantages of the model. To estimate more parameters, larger windows are needed. In contrast, only a few parameters can be estimated reliably, but this allays the issue.

Hence, using KLT we only estimate the displacement vector (with two parameters) for small windows. Here, the error is any disparity between two successive frames which cannot be described by translation and the displacement vector should be selected to minimize the residue error. For this, the local frame model is as follows:

$$J(p) = I(p - \mathbf{d}) + n(p), \tag{4.77}$$

where $I(p) = I(x, y)$, $I(p - \mathbf{d}) = I(x - \varepsilon, y - \eta)$, and n is noise. The time variable is omitted for clarity.

The residue error is shown by the given error function (as follows) over the given window W:

$$\epsilon(\mathbf{d}) = \sum_{p \in W(q)} [J(p + \mathbf{d}) - I(p)]^2. \tag{4.78}$$

Briefly, in Eq. (4.78), ϵ should be zero in the ideal case, as it is used to compute intensity changes between neighbourhoods of feature points located in I and J. The displacement vector **d** for a given feature point q can be calculated using the following steps. $W(q)$ represents a window centered at q. Usually, the size of the window is adjusted to 5×5 pixels, and the M_{it} and ε values (used below) are set to 10 and 0.03 pixels, respectively:

1. Initialize the displacement vector $\mathbf{d} = [0, 0]^\top$.
2. Compute the image gradient $\nabla I = \frac{\partial I}{\partial p}$.
3. Compute the structure matrix $\mathbf{G} = \sum_{p \in W(q)} \nabla I(p) \cdot \nabla I(p)^\top$.
4. For $k = 1$ to M_{it}:
   (a) Find the image change value (image difference) $h(p) = I(p) - J(p + \mathbf{d}^k)$.
   (b) Calculate the mismatch vector using the equation

$$\mathbf{b} = \sum_{p \in W(q)} h(p) \cdot \nabla I(p). \tag{4.79}$$

   (c) Update the displacement vector: $\mathbf{d}_{k+1} = \mathbf{d}_k + \mathbf{G}^{-1}\mathbf{b}$.
   (d) If $\| \mathbf{d}_{k+1} - \mathbf{d}_k \| < \varepsilon$ then stop.
5. The final displacement vector $\mathbf{d}$ is reported.

Feature Point Detection Not all image pixels contain valuable information for tracking. Moreover, the motion features orthogonal to the edge can be determined only if we have straight edges. In order to overcome these issues, detection of new feature points in the query image I is required. After that, we need to add those new feature points to the existing ones. Furthermore, for a reliable feature points tracking, their neighbourhood pixels should also be well-structured. Hence we define the structure matrix $\mathbf{G}$ as an assessment of the "structuredness" of the neighbourhood of pixel location q:

$$\mathbf{G} = \sum_{p \in W(q)} \nabla I(p) \cdot \nabla I(p)^\top. \tag{4.80}$$

As the matrix is positive-semidefinite and the two eigenvalues of $\mathbf{G}$ (which are λ_1, λ_2) are guaranteed to be greater than 0, we can obtain useful information about the neighbourhood region W.

In the situation where W is totally homogeneous, we have $\lambda_1 = \lambda_2 = 0$. In contrast, an edge occurs when $\lambda_1 > 0$, $\lambda_2 = 0$ and a corner occurs when $\lambda_1 > 0$ and $\lambda_2 > 0$. Therefore, "cornerness" of W can be measured using the smallest eigenvalue $\lambda = \min(\lambda_1, \lambda_2)$, where larger values indicate stronger corners. We can summarize the discussed KLT-based feature detection algorithm in seven steps as follows:

1. Initialize the structure matrix $\mathbf{G}$ and "cornerness" value λ.
2. Compute $\mathbf{G}$ and λ for all pixels in image I.
3. Estimate the λ_{max} value for the given image I.
4. Store the pixels having λ value greater than a set value (e.g. 5–10% of λ_{max}).
5. Perform a non-maxima suppression with a size of 3×3 over the neighbourhood of the residual, to retain the local maxima only. See Fig. 4.28.
6. From the residual, add as many new feature points to the existing feature points, then begin with the feature points that hold the greater λ values.

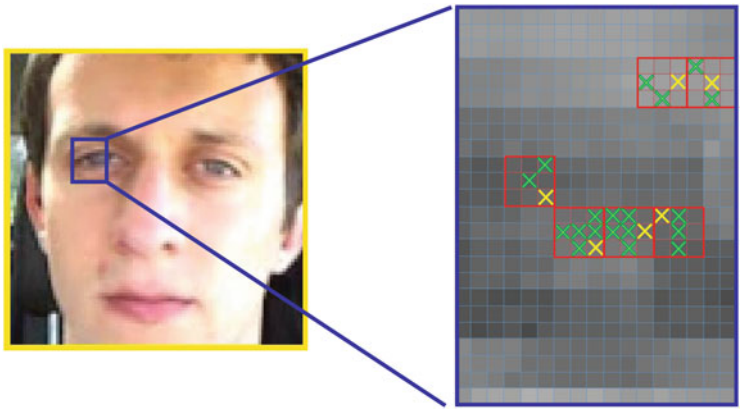

Fig. 4.28 Non-maxima suppression implementation. *Left:* A selected small window for illustration. *Right:* A 3×3 pixel neighbourhood. *Green pixels* are the result of pixels that have a λ higher than a selected percentage. The *yellow pixels* are the result of non-maxima suppression

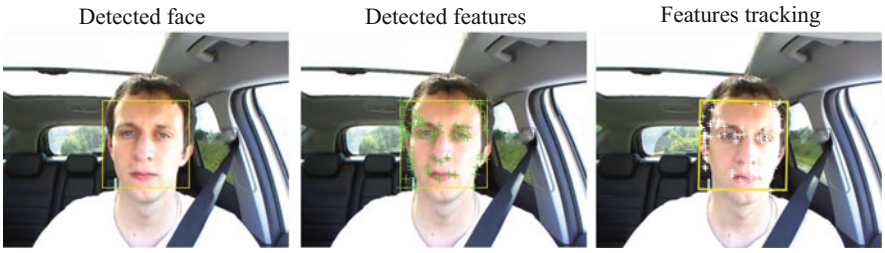

Fig. 4.29 The KLT feature tracking method used to detect a face. *Left:* Detected face, *Center:* Features are detected. *Left:* Features are tracked

7. Minimum Distance-Enforcement: In particular situations when there are points concentrated in some areas in I, set a minimum enforced distance (for example 5–10 pixels) to the existing points, besides the newly inserted points.

Figure 4.29 illustrates how KLT is used in order to detect a face in a driving-assistant application.

4.4.4 Kalman Filter

In order to use the Kalman filter for our face tracking purpose, we require a statistical model for our system and for the measurement instrument, which are usually not available or are hard to obtain. There are three major assumptions that need to be considered in the theoretical construction of a Kalman filter tracker [22, 110]:

1. The modelled system is *linear*, or has a small ignorable non-linearity;

2. The system noise is subject to be *white*;
3. The system noise is *Gaussian* in nature (normal bell-shaped curve).

By the first assumption, *linear*, we mean that the state of our system at time k can be modelled in the form of the state of the system at the previous time $(k-1)$ pre-multiplied by a matrix. The second assumption, *white noise*, means that the noise is not time-correlated, and by *Gaussian noise*, we mean that the noise amplitude can be modelled using mean and covariance.

In our system we "observe" the *measurements* $\mathbf{z}_k$, for $k = 0, \ldots, n$, which are detected faces in incoming sequences from our vision sensor, which records the driver seat area. However, we should also "estimate" the *state* $\mathbf{x}_k$ because, in practice, we cannot be 100% sure about the measurements $\mathbf{z}_k$. In our application, we estimate the state $\mathbf{x} \in \mathbb{R}^m$ as a discrete controlled process of face location by a linear stochastic difference equation, with the state transition from time $k-1$ to k as follows:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1}. \tag{4.81}$$

We also need to define a measurement $\mathbf{z} \in \mathbb{R}^n$ to model the relationship between the current state and our measurements:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{4.82}$$

where $\mathbf{A}$ is an $n \times n$ matrix known as the *state transition matrix*. The matrix transforms the previous state with time-stamp $k-1$ into the current state with time-stamp k.

$\mathbf{B}$ is an $n \times l$ matrix known as the *control input transition matrix*. This matrix is related to the optional control parameter $\mathbf{u} \in \mathbb{R}^l$, which could have positive or negative component values that need to be added to the previous state to reflect the current state. In our case, we do not have any control on the driver face movement, so no control parameter is required.

In addition to the linear state estimation discussed in Eq. (4.81), we also need to have a measurement model estimable via Eq. (4.82).

$\mathbf{z}_k$ is a *measurement vector* that takes into account the estimated state and measurement noise.

$\mathbf{H}$ is an $n \times m$ matrix referred to as the *measurement transition matrix*, which is related to the state of the measurement.

$\mathbf{x}_{k-1}$ is an $n \times 1$ vector which is referred to as the *previous state vector*, and finally $\mathbf{w}_k$ and $\mathbf{v}_k$ are process noise and measurement noise in Eqs. (4.81) and (4.82), respectively.

Process (or control) noise results from inaccurate driver face movement modelling and from machine computational tasks; and similarly, measurement noise results from an imprecise sensory device (camera) or the amount of error inherent in the face localization by the face detector algorithm. As already discussed, noise

is subject to be time-independent, white (with zero mean), and have Gaussian distributions:

$$p(\mathbf{w}) \approx N(0, \mathbf{Q}) \quad \text{and} \quad p(\mathbf{v}) \approx N(0, \mathbf{R}). \tag{4.83}$$

$\mathbf{Q}$ is the *process noise covariance* and $\mathbf{R}$ is the *measurement noise covariance*, which may randomly change in each time-stamp of the process (between time $k-1$ and k), but are limited to a maximum range of n_{max}.

4.4.4.1 Filter Implementation

Before implementing the filter in practice, here we define terms and parameters needed in our system. In order to estimate $\mathbf{x}_k$, we have to take into account all the information that is available from previous steps. If all the previous measurements before time k are available, then in order to estimate $\mathbf{x}_k$ we can form an *a priori state estimation*, which we denote as $\hat{\mathbf{x}}_k^- \in \mathbb{R}^n$. One of the ways to define the *a priori state estimate* is to calculate the expected value of $\mathbf{x}_k$ based on the measurements before (and not including) time k:

$$\hat{\mathbf{x}}_k^- = E\left[\mathbf{x}_k | \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{k-1}\right]. \tag{4.84}$$

Similarly, if all the previous measurement up to (and including) time k are available, then in order to estimate $\mathbf{x}_k$, we can define an *a posteriori state estimation* which is denoted as $\hat{\mathbf{x}}_k^+ \in \mathbb{R}^n$. So we can form a *posterior state estimate* by computing the expected value of $\mathbf{x}_k$ conditioned on previous measurements up to (and including) time k:

$$\hat{\mathbf{x}}_k^+ = E\left[\mathbf{x}_k | \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{k-1}, \mathbf{z}_k\right]. \tag{4.85}$$

$\hat{\mathbf{x}}_k^-$ and $\hat{\mathbf{x}}_k^+$ are both estimating the same $\mathbf{x}_k$, before and after actual measurement $\mathbf{z}_k$, respectively. Generally, we would expect $\hat{\mathbf{x}}_k^+$ to be more accurate than $\hat{\mathbf{x}}_k^-$ as we take more information into account to calculate $\hat{\mathbf{x}}_k^+$. Figure 4.30 describes the time slots between a priori, posteriori, and actual measurement.

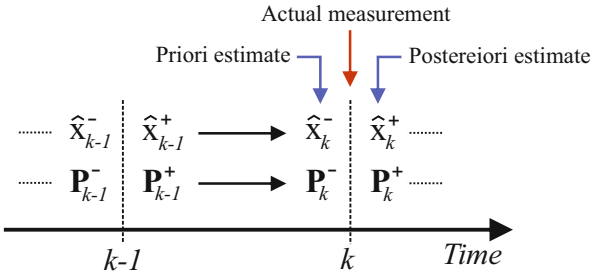

Fig. 4.30 Priori and posteriori estimates and errors

As the next step we define the *priori error* and *posteriori error* based on the actual measurement z_k:

$$\mathbf{e}_k^- = \mathbf{z}_k - \hat{\mathbf{x}}_k^-, \tag{4.86}$$

$$\mathbf{e}_k^+ = \mathbf{z}_k - \hat{\mathbf{x}}_k^+. \tag{4.87}$$

The *priori estimation error covariance* is then

$$\mathbf{P}_k^- = E\left[\mathbf{e}_k^- \mathbf{e}_k^{-T}\right] \tag{4.88}$$

and similarly the *posteriori estimation error covariance* is

$$\mathbf{P}_k^+ = E\left[\mathbf{e}_k^+ \mathbf{e}_k^{+\top}\right]. \tag{4.89}$$

Below we find $\hat{\mathbf{x}}_k^+$ by a linear combination of $\hat{\mathbf{x}}_k^-$ and the weighted difference between the measurement prediction $\mathbf{H}_k\hat{\mathbf{x}}_k^-$ and the actual measurement $\mathbf{z}_k$:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-). \tag{4.90}$$

$(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-)$ is called the *residual* or *measurement innovation*, which refers to disagreement between the predicted measurement and the actual measurement. Hence, we can determine our confidence in the state estimate. The ideal case is when this error is zero.

The *Kalman gain*, $\mathbf{K}_k$, is an $n \times m$ matrix which is a gain factor used to minimize the posteriori error covariance $\mathbf{P}_k^+$.

Since $\mathbf{P}_k$ is a function of $\mathbf{K}_k$ and $\mathbf{K}_k$ is the only unknown, we can minimize the $\mathbf{P}_k$ with respect to $\mathbf{K}_k$. Finding partial derivatives and solving the equation:

$$\frac{\partial \mathbf{P}_k}{\partial \mathbf{K}_k} = 0 \tag{4.91}$$

$\mathbf{K}$ could be calculated in the following form [27]:

$$\mathbf{K}_k = \frac{\mathbf{P}_k^- \mathbf{H}_k^\top}{\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{R}_k}. \tag{4.92}$$

When the measurement covariance matrix $\mathbf{R}$ approaches zero, the gain increases, thus causing higher importance (larger weight) for the residual value. On the other hand, when $\mathbf{P}_k^-$ approaches zero, the gain $\mathbf{K}$ leads to a smaller weight for the residual. The Kalman filter uses Bayes' rule to estimate the priori and posteriori estimates of $\hat{\mathbf{x}}_k^-$ and $\hat{\mathbf{x}}_k^+$. More details on the probabilistics used in this filter may be found in [97].

As discussed earlier, by applying this filtering technique we aim to estimate the face position as well as the face width, and then based on the feedback from a *noisy* measurement (in practice, we cannot say that we have a perfect measurement), we update our next prediction.

4.4.4.2 Tracking by Prediction and Refinement

Prediction is the step before a face has been actually located by our detection algorithm. We consider two prediction methods.

First, a simple prediction method in which we assume that the face-size and face-position shifts equally at each time step (i.e. the target has a constant velocity). So the predicted position $\mathbf{x}_k$ can be calculated based on the two previous positions $\mathbf{x}_{k-1}$ and $\mathbf{x}_{k-2}$:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + (\mathbf{x}_{k-1} - \mathbf{x}_{k-2}) = 2\mathbf{x}_{k-1} - \mathbf{x}_{k-2}. \tag{4.93}$$

However, this cannot be a rational assumption for our application and it fails after 4–5 frames of tracking, as we cannot expect a constant velocity for face movement.

As the second option, we continue with the Kalman filter method, in which the prediction is an expected measurement based on all previous measurements (not the last few measurements); in brief we expect it by $[\mathbf{z}_k|\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{k-1}]$. Before finding the expected value of $\mathbf{z}_k$, we have to know the expected state at time k, which is $E[\mathbf{x}_k|\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}]$. To solve this recursion algorithm, we take the two recursive steps below in order to refine the predicted state:

1. The *time update* step specifies the expected value of $\hat{\mathbf{x}}_k^-$ and the covariance matrix $\mathbf{P}_k^-$. Then a more accurate $\hat{\mathbf{x}}_k^+$ can be computed from the expected value of $\hat{\mathbf{x}}_k^-$ and the actual measurement $\mathbf{z}_k$.
2. The *measurement update* step exerts the measured $\hat{\mathbf{x}}_k^+$ to define the next $\hat{\mathbf{x}}_k^-$ and $\mathbf{P}_k^-$ in order to initiate the next recursion.

The time update Eqs. (4.95) and (4.96) project forward the priori state and priori error covariance from time $k-1$ to k. The expected state at k could be defined by applying the state transition matrix $\mathbf{A}$ to $\hat{\mathbf{x}}_{k-1}$:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1}. \tag{4.94}$$

Recalling the assumption that $\mathbf{w}_k$ has zero mean, and is independent of $\mathbf{x}_{k-1}$, we can determine the covariance matrix $\mathbf{P}_k^-$ as follows:

$$\begin{aligned}
\mathbf{P}_k^- &= E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^\top | \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{k-1}\right] \\
&= E\left[(\mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} - \mathbf{A}\hat{\mathbf{x}}_{k-1}^-)(\mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} - \mathbf{A}\hat{\mathbf{x}}_{k-1}^-)^\top | \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_{k-1}\right] \\
&= \mathbf{A}\mathbf{P}_{k-1}^+\mathbf{A}^\top + \mathbf{Q}.
\end{aligned} \tag{4.95}$$

The *measurement update equations* below apply corrections for the next round of the recursive process:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{R})^{-1}, \tag{4.96}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-), \tag{4.97}$$

$$\hat{\mathbf{P}}_k^+ = (1 - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-. \tag{4.98}$$

In the measurement update phase, first the *Kalman gain* $\mathbf{K}_k$ is calculated. This is based on the estimated error-covariance as calculated in the previous update step, as well as the measurement noise covariance $\mathbf{R}$. Then we estimate the posteriori state $\hat{\mathbf{x}}_k^+$ based on the current measurement $\mathbf{z}_k$, priori state $\hat{\mathbf{x}}_k^+$, and Kalman gain $\mathbf{K}_k$.

The Kalman gain expresses the certainty of old and new information using a weighted combination of all available information from the beginning of the process up to the time t. For example, if $\mathbf{K}_k$ is 0.5, then both priori and posteriori measurements have equal variance (certainty), so the expected value of $\mathbf{x}_k$ is exactly at the middle of them and the filter acts as a simple averaging. In general, if the certainty level of the new measurement is not high enough (i.e. we have a very high uncertainly), then the new measurement statistically contributes almost nothing in Equation (4.97), and the final result of $\hat{\mathbf{x}}_k^+$ would be very close to $\hat{\mathbf{x}}_{k-1}^+$. One the other hand, if we have a large variance in the $\mathbf{z}_{k-1}$ previous observations, but more accuracy in the new measurement, then we assume $\hat{\mathbf{x}}_k^+$ to be more similar to the recent measurement $\mathbf{z}_k$. The Kalman filter is responsible for finding the optimum averaging factor for each consequent time step. Finally, given the priori error covariance and calculated Kalman gain, a corrected value for $\mathbf{P}_k$ could be determined. After each pair of time-update and measurement-update, the process repeats again in a recursive manner in order to refine the parameters and predictions.

Chapter 5
Driver Drowsiness Detection

5.1 Introduction

Face and eye detection is the key step in many face analysis systems [301, 310]. Since the early 2000s, researchers such as Viola and Jones [269], Jesorsky et al. [101], and Hsu et al. [205] have made important progress in model- and learning-based object detection methods. Current research aims at increasing the robustness of the detectors. Among the proposed face detection algorithms, *boosting*-based detection with efficient use of integral image, *Haar-like features* and a cascade of weak classifiers, have defined high-performance systems [154, 225, 311].

Following the well-known Viola–Jones face detector [271], many researchers have achieved further improvements in the performance of this detector. Currently, research in the field can be categorized into four subject areas:

1. Speeding up the learning process;
2. Speeding up the detection process;
3. Defining a better trade-off between detection rate and false-positive rate;
4. Combining the three previous approaches.

Examples of the above approaches are: heuristic methods trying to improve the detection speed [210], or different versions of boosting algorithms like Float boost [133], ChainBoost [297], cost-sensitive boosting [154, 298], KLBoost [139], FCBoost [225], or RCECBoost [226] that aim at speeding up the AdaBoost convergence, or at improving the final performance of the detector.

Some variations of the Haar-like features have been proposed to improve the performance of boosting-based detectors [135, 194, 195]. These types of Haar-like based algorithms were introduced to improve the detection rate in rotated faces. Many of these face detectors use similar variations of the image preprocessing step suggested by Viola and Jones [271]. Some others like Struc et al. [245] use a variance-normalization preprocessing for face and non-face windows to deal with illumination artefacts.

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_5

Despite general progress in detection methods, face and eye detection under non-ideal lighting conditions still requires further improvements. Even in recent efforts such as [104, 147, 262], limited verification tests have been applied only for normal situations. Driver behaviour monitoring is an example of a challenging environment for eye analysis, where the light source is not uniform, or the light intensity may change rapidly and repeatedly (e.g. due to entering a tunnel, shade, turning into very bright light, or even sun strike). Although recent techniques for frontal face detection under normal lighting conditions are quite robust and precise [104, 132, 302, 312], sophisticated and sensitive tasks such as driver eye-status monitoring (open, closed) and gaze analysis, are still far away from being solved accurately.

Among related works, there are publications on single and multi-classifier approaches for the addressed area of applications. Brandt et al. [23] designed a coarse-to-fine approach for face and eye detection using Haar wavelets to measure driver blinking with satisfactory results under ideal conditions. Majumder [147] introduced a hybrid approach to detecting facial features using Haar-like classifiers in HSV colour space, but only tested it on a very limited number of frontal faces. Zhua and Ji [316] introduced robust eye detection and tracking under variable lighting conditions; nonetheless, their method is not effective without support of IR illumination. Research results in the field often suffer from a lack of verification and performance analysis on a wide-range of video or image data.

In this chapter we pursue four goals: (1) to improve noisy measurements of a Haar-classifier by a more stable solution for detecting and localizing features in the image plane; (2) to overcome issues of detection failures due to sophisticated lighting conditions by introducing a novel technique, namely an *adaptive classification*; (3) to further reduce the overall computational cost by minimizing the search region using a *Kalman filter* tracker, as well as minimizing false detections by having a limited operational area within the image plane; and (4) to perform a faster classification by introducing a novel version of Haar-features we call *Dynamic Global Haar-like features* (DGHaar).

The rest of the chapter is organized as follows. Sections 5.2 and 5.3 discuss the training phase of our classifier. Section 5.4 outlines the main ideas for the application phase. Section 5.5 discusses our adaptive classifier. Section 5.6 provides information about the implementation of a tracking module along with technical considerations for face and eye tracking. Section 5.7 provides a discussion of an image denoising method based on a phase-preserving algorithm. Global and dynamic global Haar-like features are detailed in Sect. 5.8. Section 5.9 proposes the new idea of boosted cascades based on global and dynamic global features. Section 5.10 details the experimental and validation results followed by performance analysis of the introduced techniques, as well as comparison with the standard Viola–Jones method and other state-of-the-art techniques. Section 5.11 concludes the chapter.

5.2 Training Phase: The Dataset

For driver facial analysis, we use the standard Haar-like face detector provided by Leinhart [135]. However, we trained our own eye-state classifiers, in order to perform further research on the parameters that may affect the performance of a Haar classifier. We trained and created two individual cascade classifiers, one for open-eye detection and one for closed-eye detection.

There is a general belief that any increase in the number of positive and negative images in the training dataset can improve the performance of the trained classifier. However, after several experiments we noticed that by increasing the number of training images there is also an increasing risk of feature mismatching by the boosting algorithms. The process of selecting positive and negative images is an important step that affects the overall performance considerably. Thus, a careful consideration of the number of positive and negative images and their proportional ratio was essential. The multi-dimensionality of the training parameters and the complexity of the feature space also defines further challenges. Here we review the specifications of our robust dataset as well as the optimized training parameters we obtained.

In the initial negative image database, we removed all images that contained any objects similar to a human eye (e.g. animal eyes such as tiger eyes, dog eyes). We prepared the training database by manually cropping closed or open eyes from positive images. Important questions needed to be answered were: how to crop the eye regions and in what shapes (e.g. circular, isothetic rectangles, squares). There is a general belief that circles or horizontal rectangles are best for fitting eye regions (as it is also used in the OpenCV eye detector); however, we obtained the best experimental results by cropping eyes in a square shape. We fit the square enclosing the full eye-width; and for the vertical positioning we select equal portions of skin area below and above the eye region. We cropped 12,000 open and closed eyes from online images plus seven other databases including the *Yale* dataset [130], the *FERET* database sponsored by the Department of Defense (DoD) Counterdrug Technology Development Program Office [64, 196], the *Radbound* face database [127], the *Yale B* face database [300], the *BioID* database [101], the *PICS* database [197], and the Face of Tomorrow (*FOS*) dataset [59]. The positive database includes more than 40 different poses and emotions for different faces, eye types, ages, and races as follows:

- Gender and age: females and males between 6 and 94 years old;
- Emotion: neutral, happy, sad, anger, contempt, disgusted, surprised, feared;
- Looking angle: frontal (0°), ±22.5°, and profile (±45.0°);
- Race: East-Asians, Caucasians, dark-skinned people, and Latino-Americans.

Samples of open and closed eyes from our created dataset *IntelliEye* are shown in Figs. 5.1 and 5.2.

The generated multifaceted database is unique, statistically robust and competitive compared to other training databases.

Fig. 5.1 Samples of open-eyes from our *IntelliEye* dataset

Fig. 5.2 Samples of closed-eyes from our *IntelliEye* dataset

We also selected 7,000 negative images (non-eye and non-face images) including a combination of common objects in indoor or outdoor scenes. Considering a search window of 24×24 pixels, we had about 7,680,000 windows in our negative database. An increasing number of positive images in the training process caused a higher rate for true positive cases (TP) which is good, but also increased false positive cases (FP) due to increasing the complexity of feature selection and feature matching by the AdaBoost machine learning. Similarly, when the number of negative training

images increased, it led to a decrease in both FP and TP. Therefore we needed to consider a good trade-off for the ratio of the number of negative windows to the number of positive images. For eye classifiers, we got the lowest FN and the highest TP rate when we arranged the ratio of $N_p/N_n = 1.2$. This may slightly vary for face detection.

5.3 Boosting Parameters

As the next step of the training phase, we achieved a maximum performance of the classifier by applying the following settings as boosting parameters.

- **Size of feature-window**: 21×21 pixels.
- **Total number of classifiers (nodes)**: 15 stages; any smaller number of stages brought more false positive detection, and a larger number of stages reduced the rate of true positive detection. The minimum acceptable hit rate for each stage was 99.80%. Setting the hit-rate higher than this (too close to 100%) may cause the training process to take forever, or lead to early failure before completing the training stage.
- **The maximum acceptable false alarm rate** for each stage was set at 40.0% (despite the commonly chosen rate of 50%); this error exponentially goes to zero when the number of iterations increases in each stage (i.e. 0.40^n).
- **Weight trimming threshold**: 0.95; this is the similarity weight to pass or fail an object at each stage.
- **Boosting algorithm**: among four types of boosting (Discrete AdaBoost, Real AdaBoost, Gentle AdaBoost (GAB), and LogitBoost), we got about a 5% higher TP detection rate with GAB boosting. Lienhart et al. [134] have also claimed that GAB results in lower FP ratios for face detection.

Our initial performance evaluation test on 2,000 images from the second part of the FERET database [64] plus on 2,000 other image sequences recorded in HAKA1 (our research vehicle) showed up to an 8% higher detection rate compared to the open eye classifier provided by Intel in the OpenCV library.

5.4 Application Phase: Brief Ideas

By applying heuristic techniques on training and classification phases, and by developing a rich eye dataset, we gained detection improvements as discussed in the previous section. However, we also need further improvements in the application phase, as we still may encounter issues of either missing detections (FNs) or false detections (FPs), under low light conditions. In this section and the next section we propose two methods to tackle the above issues: by being adaptive to lighting

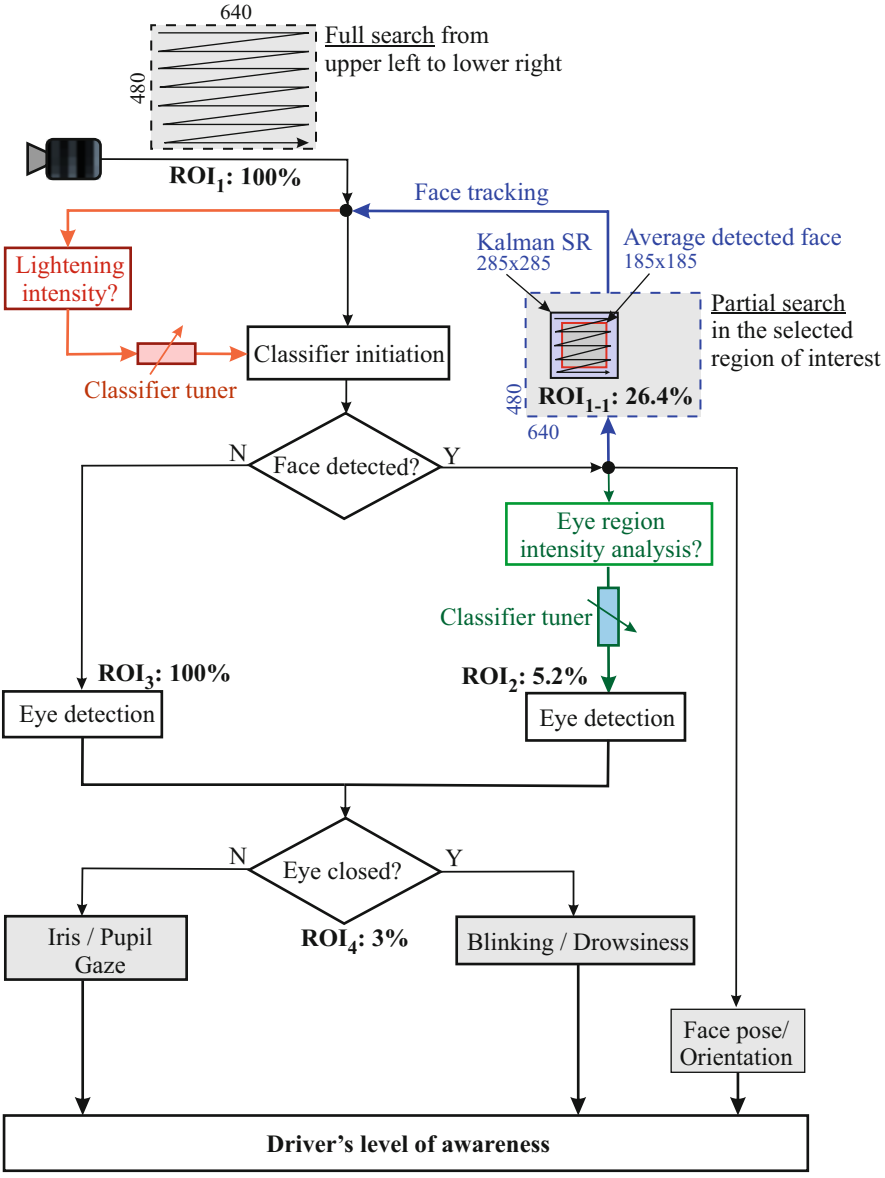

Fig. 5.3 A flowchart for driver awareness monitoring

condition, and by application of a tracking solution to limit the ROI, hence, lower FPs.

Figure 5.3 illustrates the overall structure of our approach. Using Haar-feature based classifiers, two possible options are considered. In Option 1 we quantify the

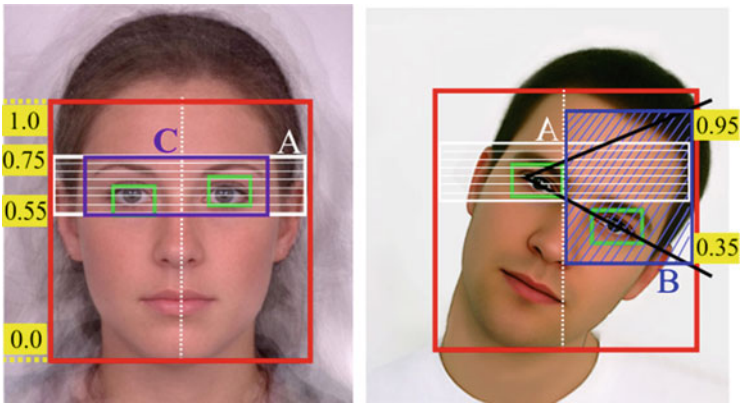

Fig. 5.4 Optimized ROIs for eye tracking, after successful face detection

region of interest (ROI_1) as 100%, which means that the whole area of an input VGA image needs to be searched by our classifiers, from upper-left to the lower-right of the input image. Generally, such a full search needs to be repeated three times for three individual Haar classifiers in order to detect "face", "open eyes", or "closed-eyes" status.

However, analyzing the ground truth information for eye localization on the two standard databases of *FERET* [64] and *Yale* [130], we calculated the eye location in a range of 0.55–0.75 as shown in Fig. 5.4. In case of head tilt, the eye location may vary in a range of 0.35–0.95 on one side of the face. Therefore, assuming an already detected face, an eye classifier can perform on regions *A* and *B* only (ROI_2), which are only 5.2% of the input image. If face detection fails, then a second full search in the image plane is required for eye classification (ROI_3), as we do not have any prior estimation for face location. This causes a total search cost of 200%. If the open-eye classifier detects at least one eye in segment *A* within the last five frames, then the closed-eye classifier is called to look at region *C* (ROI_4). This region covers about 3% of a VGA image. In brief, assuming a successfully detected face in the first stage, we have a maximum search area of 108.2% ($ROI_1 + ROI_2 + ROI_4$) for a complete eye status analysis, while a face detection failure causes an increased search cost of up to 203% ($ROI_1 + ROI_3 + ROI_4$).

Assessing 1,000 recorded frames and ground truth faces, we measured the mean size of the detected faces as 185×185 pixels, which covers only 11% of a VGA image plane. Based on this idea we plan for a partial search that potentially defines a limited search region as Option 2 instead of Option 1.

Fig. 5.5 Examples of "non-ideal" lighting conditions while driving, causing difficulties for driver's eye monitoring (images from public domains)

As shown in Fig. 5.3, implementing a face tracker reduces the search region for face detection, thus a faster eye analysis through the tracked face can be achieved. Later we discuss that using a tracking solution as Option 2, the total search cost can be reduced to around 34.6% ($ROI_{1\_1} + ROI_2 + ROI_4$) which is 6 times faster than a blind search (Option 1).

In addition to an optimized search policy, we require proper face localization from the first step, followed by robust eye status detectors, applicable on all lighting conditions. Figure 5.5 shows examples which would require robust classifiers that need to be adaptive under extremely challenging lighting conditions. This requirement is considered as classifier tuner modules in Fig. 5.3, before initiating a face or eye detection. We introduce adaptive Haar-like classifiers to overcome the weakness of a standard Viola–Jones classifier [269] under non-ideal lighting conditions.

5.5 Adaptive Classifier

The adaptation module is detailed in three sub-sections, addressing the weakness of a Viola–Jones detector, statistical analysis of intensity changes around the eye region, and dynamic parameter adaptation to overcome inefficiency of Haar-feature based detectors under non-ideal conditions.

5.5.1 Failures Under Challenging Lighting Conditions

We examined five well-recognized and publicly available Haar classifiers developed by Castrillon, Lienhart, Yu and Hameed [292], for our nominated application, driver monitoring. Although they work well for non-challenging and normal lighting scenes, we realized that due to frequent on-off shades and artificial lighting in day and night, those Haar-like classifiers are likely to fail. The situation becomes even more complicated when one part of the driver's face is brighter than the other

part (due to light falling in through a side-window), making eye status detection extremely difficult. Similar to the training phase, there are some parameters that need to be noted in the application phase of a Haar-based classifier. The main parameters are:

- Initial search window size (SWS) or initial size of sliding windows, which should be equal to or greater than the scale size of positive images in the training step (i.e. 21×21 pixels).
- Scale factor (SF) to increase the SWS in each subsequent search iteration (e.g. 1.2, which means a 20% increase in window size for each search iteration).
- Minimum expected number of detected neighbours (MNN), which is needed to confirm an object, when there are multiple object candidates in a small region (e.g. 3 neighbour objects).

In general, the smaller the SF, the more detailed the search in each iteration. However, this also causes higher computational costs.

Decreasing MNN causes an increase of detection rate; however, it increases the false detection rate as well. Larger values for MNN lead to more strictness to confirm a candidate for face or eye, thus a reduced detection rate.

Figure 5.6 shows potential issues related to the MNN parameter for an eye classification example. The left image shows 10 initial eye candidates before trimming by the MNN parameter. Detections are distributed in 5 regions, each region shows 1–4 overlapping candidates. In order to minimize this problem, it is common to assign a trade-off value for the MNN parameter to achieve the best possible results. Figure 5.6, top right, shows one missed detection with MNN equal to 3 or 4, and Fig. 5.6, bottom right, shows one false detection with MNN equal to 2. MNN equal to 1 causes 3 false detections, and any MNN greater than 4 leads to no detection at all; so there is no optimum MNN value for this example.

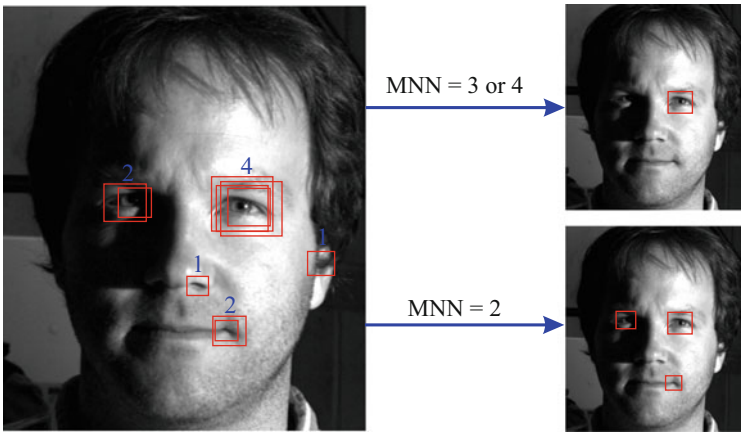

Fig. 5.6 *Left*: Initial eye detection results before trimming by the MNN factor. *Right*: Final detections using various MNN values

We conclude that although we can define trade-off values for SWS, SF and MNN to obtain the optimum detection rate for "ideal" video sequences, a diversity of changes in light intensity over the target object can still significantly affect the performance of the given classifier in terms of TP and FP rates [43, 290].

5.5.2 *Hybrid Intensity Averaging*

To cope with the above-mentioned issues, we propose that Haar-classifier parameters have to be adaptive, varying with time depending on lighting changes. Figures 5.7 and 5.8 illustrate that we cannot measure illumination changes by simple intensity averaging over the whole input frame: In the driving application, there can be strong back-light from the back windshield, white pixel values around the driver's cheek or forehead (due to light reflections), or dark shades on the driver's face. All these conditions may negatively affect the overall mean intensity measurement. Analysing various recorded sequences, we realized that pixel intensities in an eye region can change independently from its surrounding regions. Focusing on a detected face region, Fig. 5.7, right, shows very dark and very bright spots in two sample faces (in a grey-level range of 0–35 and 210–255, respectively). It also shows a considerable illumination difference for the left and right side of a driver's

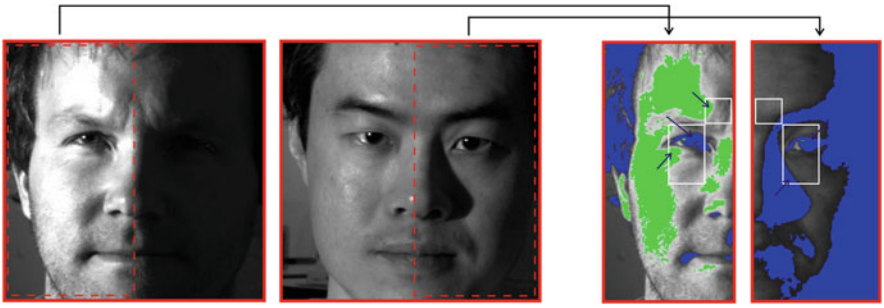

Fig. 5.7 Mean intensity measurement for eye detection by excluding *very bright* and *very dark* regions. *Green*: thresholding range 210–255. *Blue*: thresholding range 0–35

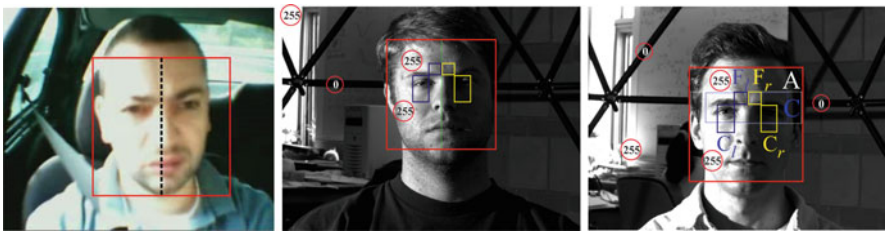

Fig. 5.8 Selected regions to sum up the mean intensity around eyes (Images Source: Yale database)

face. Apart from the iris intensity (for dark or light eyes), the surrounding eye intensities play a very crucial role in eye detection. Thus, proper classifier parameter adjustment based on proper intensity measurement in the region can guarantee a robust eye detection. Following this consideration, we defined two rectangular region around eyes (as shown in Fig. 5.7, right) which not only can provide a good approximation of both vertical and horizontal light intensities around the eyes, but they are also very marginally influenced by the outer green or blue (very bright or very dark) regions.

Considering an already detected face, and expected eye regions A and C (based on Fig. 5.4), we can geometrically define C_r, F_r, C_l and F_l as being the optimum regions in order to gain an accurate estimation of light intensity around the eyes (Fig. 5.8). We also consider independent classifier parameters for the left and right half of the face, as each half of the face may receive different and non-uniform light exposures.

Performing a further analytical step, Fig. 5.7, right, shows that a few small green or blue segments (extreme dark or light segments) have entered the regions of white rectangles. This can affect the actual mean intensity calculations in the C or F regions. Thus, in order to reduce the effect of this kind of noise on our measurements, we apply a hybrid averaging by combining *mean* and *mode* ($\mathtt{Mo}$) of pixel intensities as follows:

$$I_r(\alpha) = \frac{1}{2}\left[\left(\alpha \cdot \mathtt{Mo}(C_r) + \frac{(1-\alpha)}{m}\sum_{i=1}^{m} C_r^i\right) + \left(\alpha \cdot \mathtt{Mo}(F_r) + \frac{(1-\alpha)}{n}\sum_{j=1}^{n} F_r^j\right)\right], \tag{5.1}$$

where $I_r(\alpha)$ is the hybrid intensity value of the *right eye* region of the face, and m and n are the total numbers of pixels in the regions C_r and F_r, in the range $[0, 255]$, pointing to cheek and forehead light-intensity, respectively.

An experimentally defined α-value of 0.66 assumes a double importance of *mode* intensity measurement compared to *mean* intensity; the integration of the *mode* reduces the impact of eye iris colour (i.e. blue segments) and of the effect of very bright pixels (i.e. green segments) for our adaptive intensity measurement. Similarly, we can calculate $I_l(\alpha)$ as the hybrid intensity value of the *left eye* region.

5.5.3 Parameter Adaptation

The final step of the detection phase is *classifier parameter optimization* based on the measured I_r and I_l values, to make our classifier adaptive for every individual

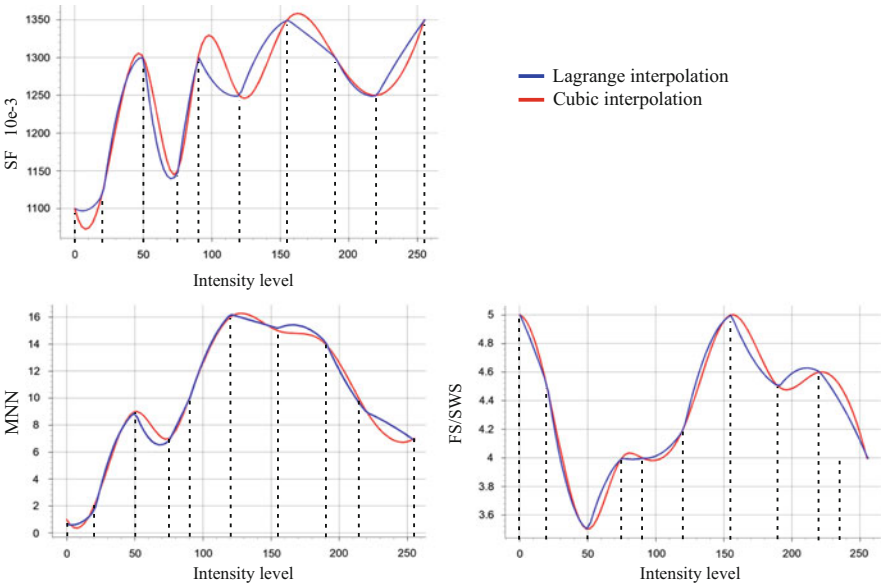

Fig. 5.9 Curve fitting for the parameters MNN, SF and SWS using Lagrange and cubic interpolation

Table 5.1 Optimum parameters for 10 selected intensity levels in terms of *Search Window Size*, *Scale Factor*, and *Minimum Number of Neighbours*. FS: detected *Face Size*

Light intensity	*SWS*	*SF*	*MNN*
0	FS/5.0	1.10	1
20	FS/4.5	1.12	2
50	FS/3.5	1.30	9
75	FS/4.0	1.15	7
90	FS/4.0	1.30	10
120	FS/4.2	1.25	16
155	FS/5.0	1.35	15
190	FS/4.5	1.30	14
220	FS/4.6	1.25	9
255	FS/4.0	1.35	7

input frame. Now we need to find optimum parameters (SWS, SF, MNN) for all intensity values in the range 0–255, which is a highly time-consuming experimental task. Instead, we defined optimum parameters for 10 selected intensities, followed by a data interpolation method to extend those parameters to all 256 intensity values (Fig. 5.9).

Table 5.1 shows optimum parameter values for 10 data points obtained from 20 recorded videos in different weather and lighting conditions. These factors are adjusted to lead to the highest rate of true positives for each of the 10 given intensities. The parameter values in Table 5.1 show a non-linear behaviour over intensity changes; therefore we applied two non-linear *cubic* and *Lagrange*

interpolations [48] on the data samples to extend the adaptive values in the whole range from 0 to 255. While the cubic interpolation was performing better for MNN and SWS, the Lagrange interpolation provided a better curve fitting for the highly fluctuating SF data samples.

5.6 Tracking and Search Minimization

As part of the discussed framework shown in Fig. 5.3, this section pursues five objectives including minimizing the search region for eye status detection, time efficiency, lower computational cost, more precise detections, and lower rate of false detection.

5.6.1 Tracking Considerations

A simple tracking around a previously detected object can easily fail due to occlusion or fast change in both size and moving trajectory of the object (Fig. 5.10). Therefore, in order to minimize the search region, we need to perform a dynamic and intelligent tracking strategy. We considered three trackers as the candidates of choice: *Particle filter*, *Kalman filter*, and *unscented Kalman filter*. While the Particle filter can generally perform better in the case of multi-object tracking and complicated scenarios, the Kalman filter performs more accurately for single-object tracking [151]. Furthermore the Particle filter has significantly more computational cost than the Kalman filter [217], which is in contradiction with our policy to develop a real-time ADAS. In our application the driver's face cannot move quickly or irregularly, so we assume it as a linear movement or with a little non-linearity. Altogether our filter of choice is a standard Kalman filter.

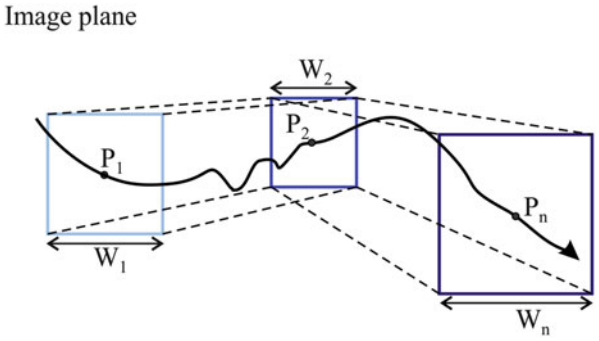

Fig. 5.10 Sample movement trajectory: simultaneous changes in face position and size

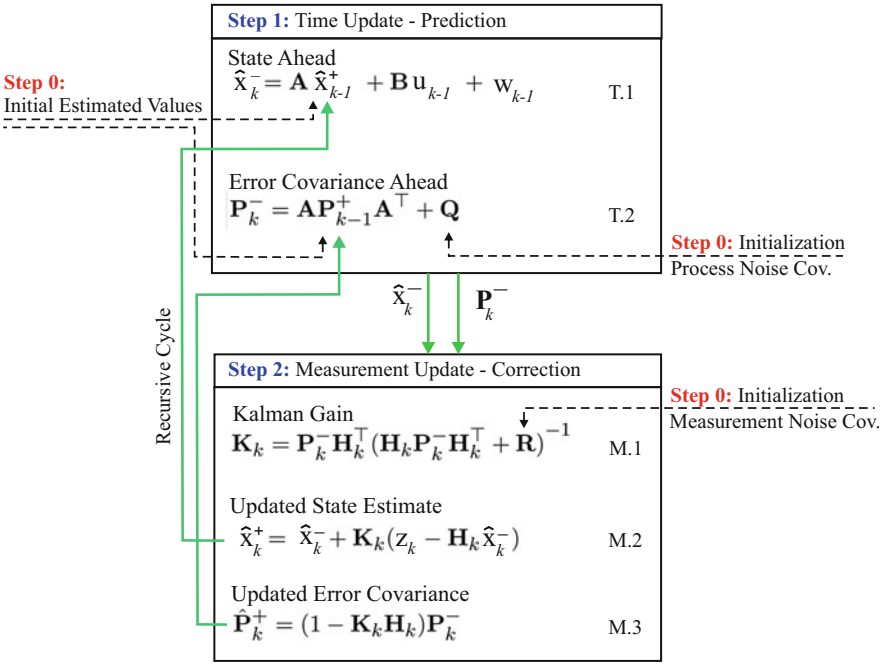

Fig. 5.11 Kalman filter in a nutshell

Following the detailed discussion in Chap. 4, Fig. 5.11 shows the brief structure of the Kalman filter [110] including time update and measurement steps, where $\hat{\mathbf{x}}_k^-$ and $\hat{\mathbf{x}}_k^+$ are priori and posteriori states estimated for the centre of the detected face, $\mathbf{z}_k$ is a *Haar-classifier measurement vector*, $\mathbf{A}$ is an $n \times n$ matrix referred to as the *state transition matrix* which transforms the previous state at time-step $k-1$ into the current state at time-step k, $\mathbf{B}$ is an $n \times l$ matrix referred to as the *control input transition matrix* which relates to the optional control parameter $u \in \mathbb{R}^l$, $\mathbf{w}_k$ is process noise which is assumed to be Gaussian and white, and $\mathbf{H}$ is an $n \times m$ matrix called the *measurement transition matrix*. $\mathbf{P}_k^-$ and $\mathbf{P}_k^+$ are the *priori and posteriori estimation error covariance* based on predicted and measured values (by Haar-classifier), and $\mathbf{K}_k$ is the *Kalman gain*.

5.6.2 *Filter Modelling and Implementation*

There are many different motion equations such as linear acceleration, circular acceleration, or Newtonian mechanics; however, for a driver's face movements we simply consider a linear dynamic system and assume constant acceleration for a

short Δt time frame. We define the state vector as:

$$\mathbf{x}_t = \left[x, y, w, v_x, v_y, a_x, a_y\right]^\top, \tag{5.2}$$

where x and y are the centre of the detected face in the image coordinate, w is the face width, v_x and v_y are the velocity of the moving face, and a_x and a_y are acceleration components, respectively. Based on the theory of motion we have

$$p(t+1) = p(t) + v(t)\Delta t + a(t)\frac{\Delta t^2}{2}, \tag{5.3}$$

$$v(t+1) = v(t) + a(t)\Delta t, \tag{5.4}$$

where p, v, a, Δt are position, velocity, acceleration, and time difference between input images, respectively. Δt is the average processing time which is the time between starting the process of face detection on a given frame at time k until the end of the eye detection process and accepting the next frame at time $k+1$. Relying on the definition of state transition in Eq. T.1 (in Fig. 5.11), we modelled the transition matrix $\mathbf{A}$ as a 7×7 matrix. Thus, the next state is estimated as below:

$$\begin{bmatrix} x_k \\ y_k \\ w_k \\ v_{x_k} \\ v_{y_k} \\ a_{x_k} \\ a_{y_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ w_{k-1} \\ v_{x_{k-1}} \\ v_{y_{k-1}} \\ a_{x_{k-1}} \\ a_{y_{k-1}} \end{bmatrix} + \begin{bmatrix} r(2e-3) \\ r(2e-3) \\ . \\ . \\ . \\ . \\ r(2e-3) \end{bmatrix} . \tag{5.5}$$

Before running the filter in the real world, we need to have a few initializations, including $\hat{\mathbf{x}}_0^+$, $\hat{\mathbf{P}}_0^+$, $\mathbf{R}$, $\mathbf{Q}$, and $\mathbf{H}$. The measurement noise covariance matrix, $\mathbf{R}$, directly depends on the measurement accuracy of our camera along with the accuracy of our face/eye detection algorithm. Comparing the ground truth information and the result of our face detection method, we determined the variance of the measurement noise in our system as $\mathbf{R} = rand(1e-4)$. The determination of $\mathbf{Q}$ is generally more difficult than $\mathbf{R}$, and needs to be tuned manually. A good tuning of $\mathbf{R}$ and $\mathbf{Q}$ stabilizes $\mathbf{K}_k$ and $\mathbf{P}_k$ very quickly after a few iterations of filter recursion. We achieved the best system stability when we set the process noise to $\mathbf{Q} = rand(2e-3)$.

We take $\mathbf{H} = 1$ because the measurement is composed of only the state value and some noise. Matrix $\mathbf{B}$ can also be omitted as there is no external control for driver face movement. For $\hat{\mathbf{x}}_0^+$ and $\hat{\mathbf{P}}_0^+$ we assumed the initial position of a face at position $x = 0$ and $y = 0$, with an initial speed of zero, as well as a posteriori error covariance of 0. We also considered Δt as being between 33 and 170 ms, based on computation cost on our PC platform, and a processing rate between 6 and 30 Hz. We discuss this further in Sect. 5.10.

5.7 Phase-Preserving Denoising

Different than a basic variance normalization, this section proposes a further preprocessing step to overcome the problem of noisy images and that of images contaminated with illumination artefacts. We use the *phase-preserving denoising* method suggested by Kovesi in [125], which is able to preserve the important *phase information* of the images, based on the non-orthogonal and complex-valued log-Gabor wavelets.

The method assumes that phase information of images is the most important feature and tries to preserve this information, while also maintaining the magnitude information.

Let M_ρ^e and M_ρ^o denote the even-symmetric and odd-symmetric wavelets at a scale ρ, which are known as quadrature pairs. Considering the responses from each quadrature pair of the filters, a resultant response vector is defined as follows:

$$[e_\rho(x), o_\rho(x)] = [f(x) * M_\rho^e, f(x) * M_\rho^o], \tag{5.6}$$

where $*$ denotes convolution and $e_\rho(x)$ and $o_\rho(x)$ are the real and imaginary parts in the complex-valued frequency domain, respectively. The amplitude of the transform at a given wavelet scale is given by:

$$A_\rho(x) = \sqrt{e_\rho(x)^2 + o_\rho(x)^2} \tag{5.7}$$

and the local phase is given by:

$$\varphi_\rho(x) = \tan^{-1}\Big[\frac{o_\rho(x)}{e_\rho(x)}\Big]. \tag{5.8}$$

Having one response vector for each filter scale, there will be an array of such vectors for each pixel x in a signal. The denoising process includes defining an appropriate noise threshold for each scale as well as reducing the magnitudes of the response vectors, while maintaining the phase without any changes. The most important step of the denoising process is determining the thresholds. To this end, Kovesi [125] used the expected response of the filters to a pure noise signal.

If the signal is purely Gaussian white noise, then the position of the resulting response vectors from a wavelet quadratic pair of filters at some scale forms a 2D Gaussian distribution in the complex plane [125]. Kovesi shows that the distribution of the magnitude responses can be modelled by the Rayleigh distribution

$$R(x) = \frac{x}{\sigma_g^2} \exp^{\frac{-x^2}{2\sigma_g^2}}. \tag{5.9}$$

Also, the amplitude response from the smallest scale of the filter pair across the whole image will be the noise with Rayleigh distribution.

$$\mathcal{V}(\psi_p) = \omega_1 \cdot S_W - \omega_2 \cdot S_B = 15573 - 14733 = 840$$

$$\mathcal{V}(\psi_p) = \omega_1 \cdot S_W - \omega_2 \cdot S_B = 23429 - 19326 = 4103$$

A sample Haar-feature applied in the eye region

Fig. 5.12 Improved result for $\mathcal{V}(\psi_p)$ (feature value) after phase-preserving denoising. *Top*: Original image. *Middle*: Denoised image. *Bottom*: Sample of an applied Haar-like feature

Finally, by estimating the mean value μ_r and standard deviation σ_r of the Rayleigh distribution, the shrinkage threshold can be estimated. The thresholds are automatically determined and applied for each filter scale.

A number of parameters impacts the quality of the denoised output image. The threshold of noise standard deviations to be rejected (k), the number of filter scales to be used (N_ρ), and the number of directions (N_r) are the key parameters.

We set the parameters $k = 3$, $N_\rho = 4$ and $N_r = 4$ in our experiments. These parameters result in an acceptable representation of small and middle-size faces (in a range of 30×30 pixels to 200×200 pixels). However, for large faces, it can lead to erroneous results. One approach is to use a set of different parameters to obtain different images. Another approach is to scale the original images and then use the same parameters for conversions. We used the second approach to speed up the process. After a conversion to the denoised form, adaptive histogram equalization is used for both training and test images.

Figure 5.12 shows the sample Haar-like feature applied on the eye region and the discriminate advantage of using denoised images. The increased feature values $\mathcal{V}(\psi_p)$ lead to a faster convergence of the classifier.

Figure 5.13 shows the results obtained with the same standard Haar-like detector for a "noisy" input image (top), and the improvements on the denoised version of the same image (bottom), still with two missing detections.

5.8 Global Haar-Like Features

We apply the technique of denoising in conjunction with a novel version of the Haar-like features method which together lead to an outperforming result. As one of the main contribution of this chapter, we propose *global Haar-like* (GHaar) features

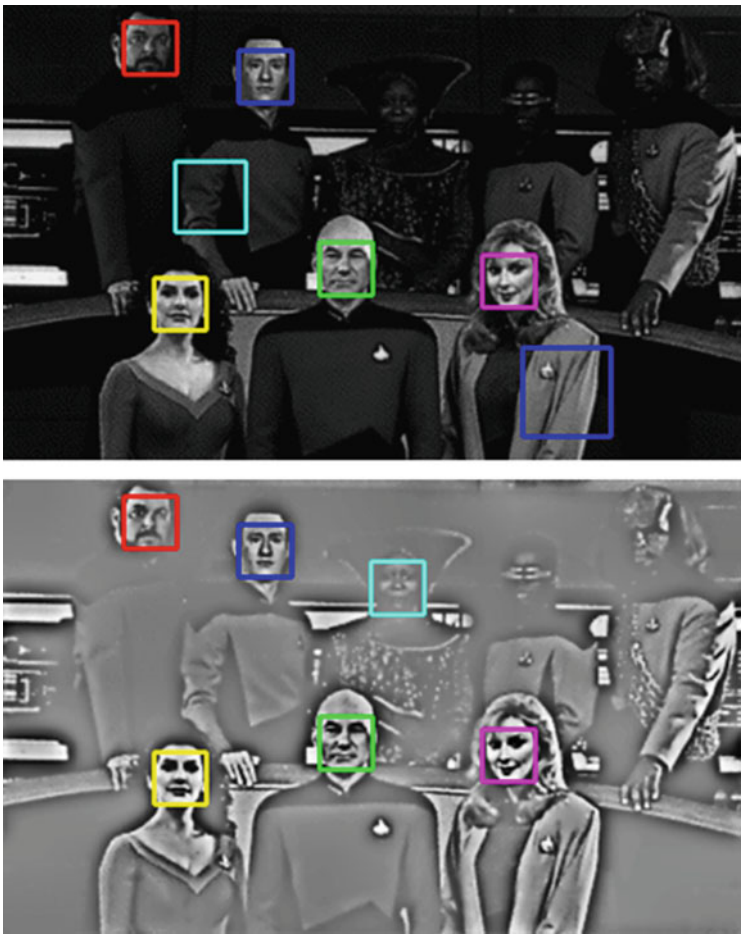

Fig. 5.13 Detection results using the standard Haar-like detector. *Top*: Detection results on a "noisy" input image. *Bottom*: Detection results on the denoised image

which complement the commonly used standard Haar-like features. With global Haar-like features we introduce a new point of view to gain benefit from the intensity information of the whole sliding query window. This is beyond the standard Haar-like features that only look through adjacent dark-bright rectangular regions.

5.8.1 Global Features vs. Local Features

Viola and Jones used five types of Haar-like features, which, from now on, we call *local Haar features*.

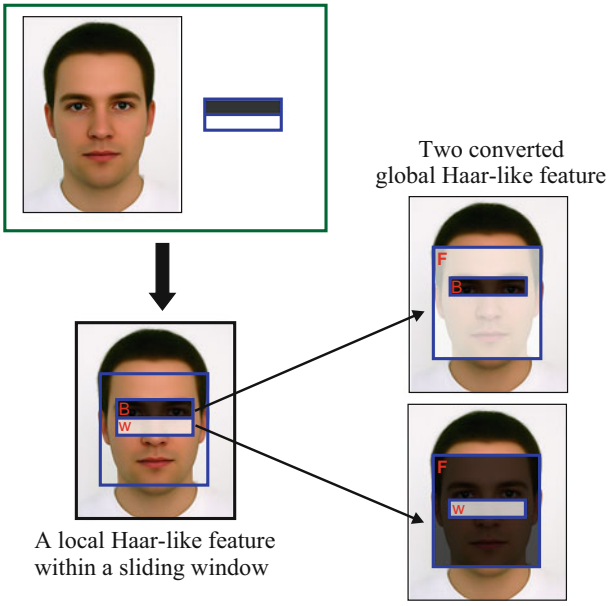

Fig. 5.14 Extraction of Global Haar-features from a standard Haar-feature

Recalling the discussions in Sect. 4.2.3 and Eq. (4.14), the value of a local feature $\mathscr{V}(\psi_i)$ is computed by a weighted subtraction of integral images on white and black regions, $\mathscr{V}(\psi_i) = \omega_1 \cdot S_{W_i} - \omega_2 \cdot S_{B_i}$, where S_{B_i} and S_{W_i} are integral images of black and white rectangles in the given local Haar-feature.

For every local feature, we introduce two *global Haar-like features* as

$$\mathscr{V}_{G_B}(\psi_i) = S_F - S_{B_i} \quad \text{and} \quad \mathscr{V}_{G_W} = S_F - S_{W_i}, \tag{5.10}$$

where S_F is the integral value of the whole sliding window (Fig. 5.14). Global features will be used in conjunction with local features. We call them global features, as these feature values provide global information in addition to standard (local) information.

In short, the term *global* in this chapter refers to a comparison between the whole window and a portion of that window, while the common local features only refer to *adjacent* rectangles of *equal* size.

If a local feature is being selected by a boosting algorithm for the formation of a cascade, then it would be a candidate to be a global feature as well. Global features are faster than local features, since the required values for calculating them are already computed at earlier stages. Figure 5.14 illustrates conversion steps from one local feature to two global features.

A major problem with the cascaded classifiers is that in the last stages, more and more weak classifiers need to reject 50% of non-face samples, taking much more time than earlier stages. The inclusion of these late-stage weak classifiers can highly increase the computational cost. Global features are an efficient alternative for two reasons. Firstly, they are faster to calculate and secondly, due to adding a new level of information (by extracting different patterns than the local features), they can offer a better classification in early stages without the need to go to the higher stages.

5.8.2 *Dynamic Global Haar Features*

Based on the method proposed in the previous section, a current local feature $\mathcal{V}(\psi_i)$ is now accompanied by two global features $\mathcal{V}_{G_W}(\psi_i)$ and $\mathcal{V}_{G_B}(\psi_i)$, to be used in a weak classifier. In the dynamic version of global Haar features (DGHaar), we update S_F by

$$S_{F_d} = S_F + \sum_{i=1}^{j \leq n} (S_{W_i} - S_{B_i}), \tag{5.11}$$

where n is the total number of local features in the current cascade and j is the current index over the global feature being assessed.

By using this equation, as the input sliding windows progress through the cascade, the value of S_F will be updated to S_{F_d}. We call these types of features *dynamic global* Haar-like features. We expect that the dynamic global features can obtain a higher detection rate and a lower false positive rate in comparison with the non-dynamic version of the global features.

5.9 Boosting Cascades with Local and Global Features

In this section, a cascade of weak classifiers is designed by considering the global features. It is a common approach that each stage of a cascade rejects 50% of non-object samples while the true detection rate remains close to optimal (e.g. 99.8%).

When global features are considered, it is important to decide which of the local features should be considered as being global. One approach is to temporarily keep a current global feature and continue to search for the next local feature, without considering the effect of the current global feature. If a candidate global feature shows a better rejection rate, then it is efficient to choose the candidate as an appropriate feature, and then start searching for the next local features again. Also,

Algorithm 1 Learning weak classifiers by using local and dynamic global features

Input: N_p positive samples; N_n negative samples.
Initialisation: Let $\mathscr{V}_{G_W} = \mathscr{V}_{G_B} = S_F$, where S_F is the sum of intensities in the whole window. Let $k = 1$.
Output: $(\psi_l^k, (\Phi_B^k, \Phi_W^k)), \ldots, (\psi_l^n, (\Phi_B^n, \Phi_W^n))$.

1: Find the k^{th} local weak classifier ψ_l^k with threshold $T_l^k = \sum_{i=1}^{m_k} (S_{W_i} - S_{B_i})$; where m_k is the total number of local features in the k^{th}classifier.
2: Find the next $(k + 1^{th})$ weak classifier ψ_l^{k+1};
3: Find the k^{th} pair of global weak classifiers Φ_B^k and Φ_W^k, corresponding to the black and white parts of the local feature, respectively; set $T_B^k = \sum_{i=1}^{m_k} (\mathscr{V}_{G_B} - S_{B_i})$, and $T_W^k = \sum_{i=1}^{m_k} (\mathscr{V}_{G_W} - S_{W_i})$;
4: Decide to choose the best classifier(s) among (Φ_B^k), (Φ_W^k) and ψ_l^{k+1};
5: **if** a global classifier is selected **then**
6: Update the values of $\mathscr{V}_{G_W}$ and $\mathscr{V}_{G_B}$ as: $\mathscr{V}_{G_W} = \mathscr{V}_{G_W} + S_{W_i}$, $\mathscr{V}_{G_B} = \mathscr{V}_{G_B} - S_{B_i}$;
7: Set $k = k + 1$, find the next local weak classifier ψ_l^k;
8: Go to Step 3;
9: **else**
10: $k = k + 1$;
11: Add ψ_l^k to the cascade and search for next local weak classifier ψ_l^{k+1};
12: Go to Step 3;
13: **end if**

even if their rejection rate becomes equal or near to equal, the global features are preferred.

Pseudo-code for the learning cascades is provided as Algorithm 1. Applying the learning process, the following weak classifiers are obtained:

$$(\psi_l^k, (\Phi_B^k, \Phi_W^k)), \ldots, (\psi_l^n, (\Phi_B^n, \Phi_W^n)), \tag{5.12}$$

where the optional pairs (Φ_B^k, Φ_W^k) denote global features and $(\psi_l^n, (\Phi_B^n, \Phi_W^n))$ are the triplets referring to the sets of "one" local feature and "two" subsequent global features.

5.10 Experimental Results

This section reports on the experiments done for the proposed adaptive classifier, the tracking platform, and the global Haar-classifier.

Two video sequences and two datasets were used in the experiments. Figures 5.15, 5.16, 5.17 and 5.18 show the results of eye-status detection using a standard classifier and the implemented adaptive classifier. The images have been selected from various recorded video sequences with extremely varying lighting. Tables 5.2 and 5.3 provide the details of TP and FP detection rates performed on the two most difficult videos (5 min each), and two face datasets (2,000 images each).

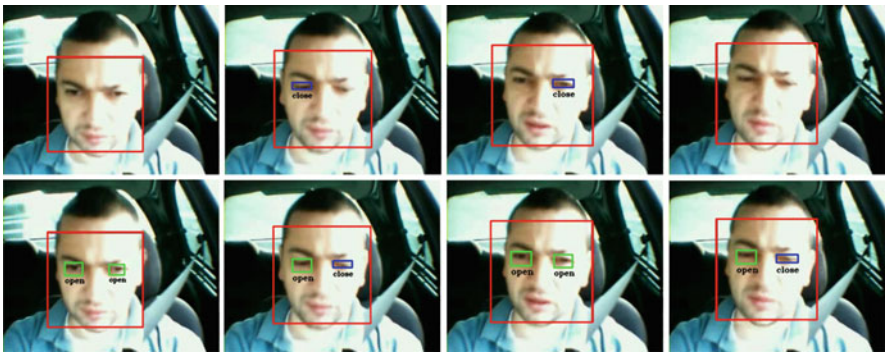

Fig. 5.15 Video 1: Face and eye-state detection under sun strike and very bright lighting condition; standard classifier (*top*) vs. adaptive classifier (*bottom*)

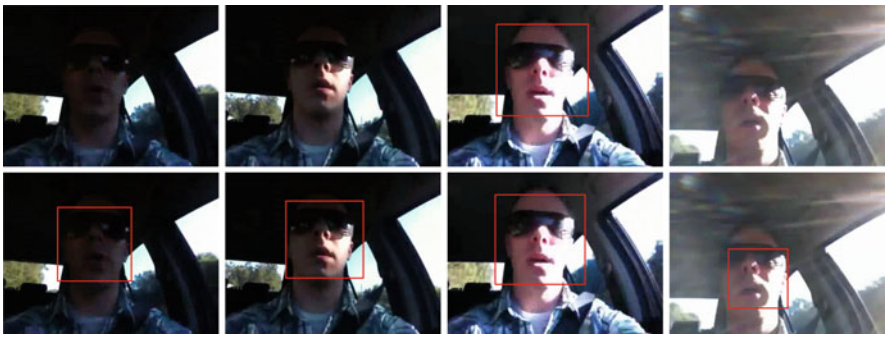

Fig. 5.16 Video 2: Face detection with sunglasses under light-varying conditions; standard classifier (*top*) vs. adaptive classifier (*bottom*)

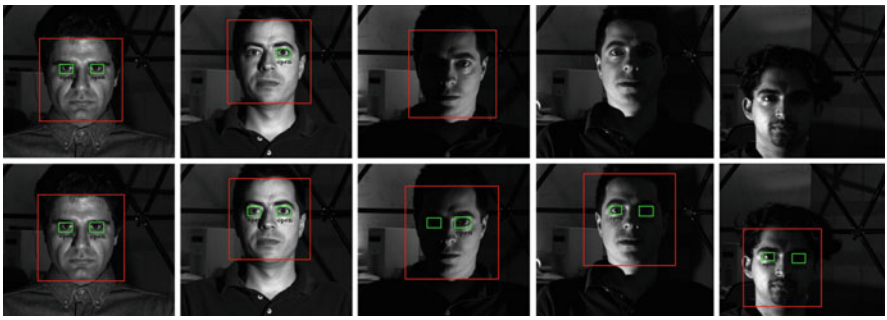

Fig. 5.17 Yale database: face and eye-state detection under difficult lighting conditions; standard classifier (*top*) vs. adaptive classifier (*bottom*)

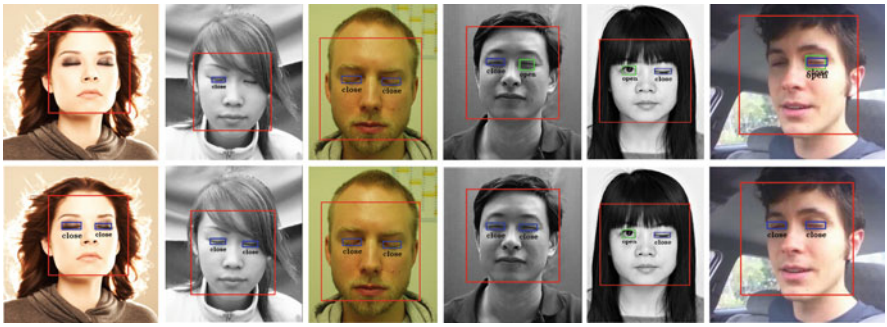

Fig. 5.18 Closed-eye dataset: face and eye state detection with standard classifier (*top*) and adaptive classifier (*bottom*)

Table 5.2 Performance analysis for the *standard* classifier

	Standard V-J classifier					
	Face		Open		Closed	
	TP	FP	TP	FP	TP	FP
Video 1	97.5	0.01	82	3.2	86	4.4
Video 2	81.1	1.02	–	0.5	–	0.32
Yale DB	86.3	0.05	79.4	0.1	–	0.07
Closed eye DB	92.2	0.06	87.5	3.7	84.2	3.9

Table 5.3 Performance analysis for the *adaptive classifier*

	Proposed adaptive classifier					
	Face		Open		Closed	
	TP	FP	TP	FP	TP	FP
Video 1	99.3	0	96.1	0.27	95.7	0.32
Video 2	94.6	0.01	–	0.01	–	0
Yale DB	98.8	0.02	97.3	0	–	0.01
Closed eye DB	99.5	0.02	99.2	0.4	96.2	0.18

Figure 5.19 illustrates partial face tracking results and error indices in x-coordinates, for 450 recorded sequences while driving. Using adaptive Haar-like detectors, we rarely faced detection failure for more than five consecutive frames. However, we deliberately deactivated the detection module for up to 15 consecutive frames to check the tracking robustness (shown as a grey bar in Fig. 5.19). The results show promising tracking without any divergence. Similar results were obtained for y-coordinates and face size tracking. Comparing ground truth and tracking results, the average error index was ± 0.04. Adding a safety margin of 4% around the tracking results, we were able to recursively define an optimized rectangular search region for the adaptive Haar-classifier instead of a blind search

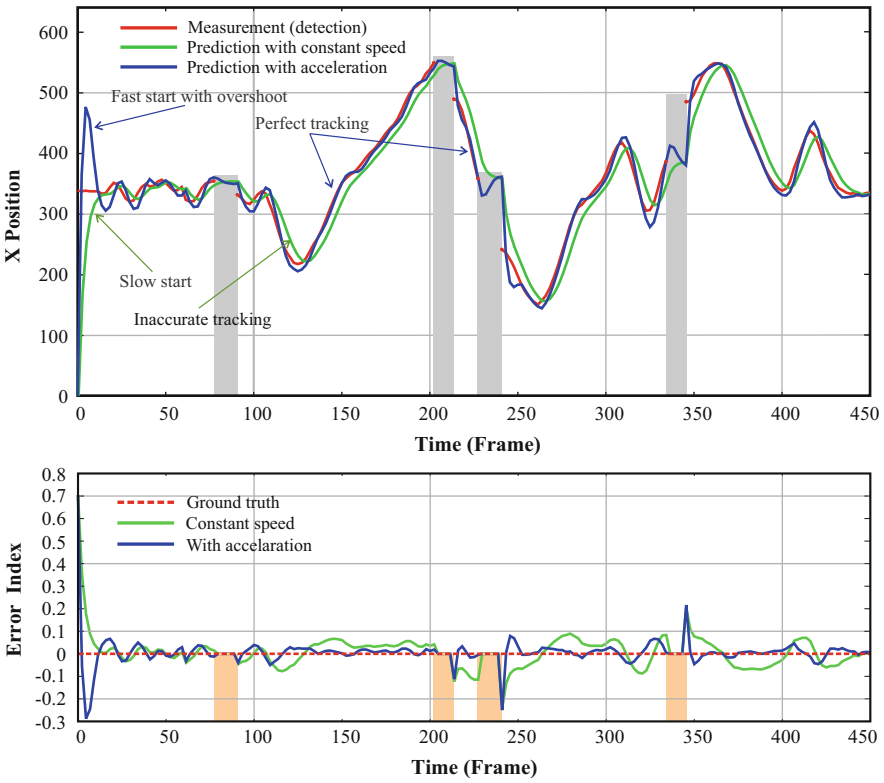

Fig. 5.19 Tracking results with acceleration (the *blue graph*) and without acceleration (the *green graph*). *Grey blocks*: the moments with deliberately deactivated measurement-module (no *red graph*), but still successful tracking results

in the whole image plane. We define

$$P_k^1 = (x_{k-1}^1 - 0.04 \times 640,\ y_{k-1}^1 - 0.04 \times 480) \quad \text{and} \tag{5.13}$$

$$P_k^2 = (x_{k-1}^2 + 0.04 \times 640 + 1.4 \cdot w,\ y_{k-1}^2 + 0.04 \times 480 + 1.4 \cdot w) \tag{5.14}$$

as the optimized search region, where P_k^1 and P_k^2 point to the upper-left and lower-right corners of search regions at time k, pairs of $(x_{k-1}^1,\ y_{k-1}^1)$ and $(x_{k-1}^2,\ y_{k-1}^2)$ are upper-left and lower-right coordinates of predicted faces at time $k-1$, and w is the predicted face width at time $k-1$.

Figure 5.20a shows a good tracking after 5 frames; Fig. 5.20b shows a perfect tracking after 10 frames. Figure 5.20c illustrates a failed face detection due to face occlusion while steering; however, a successful face tracking (yellow frame) has still led to a proper eye detection. Figure 5.20d shows another good "match" of detection and tracking with accurate closed-eye detection.

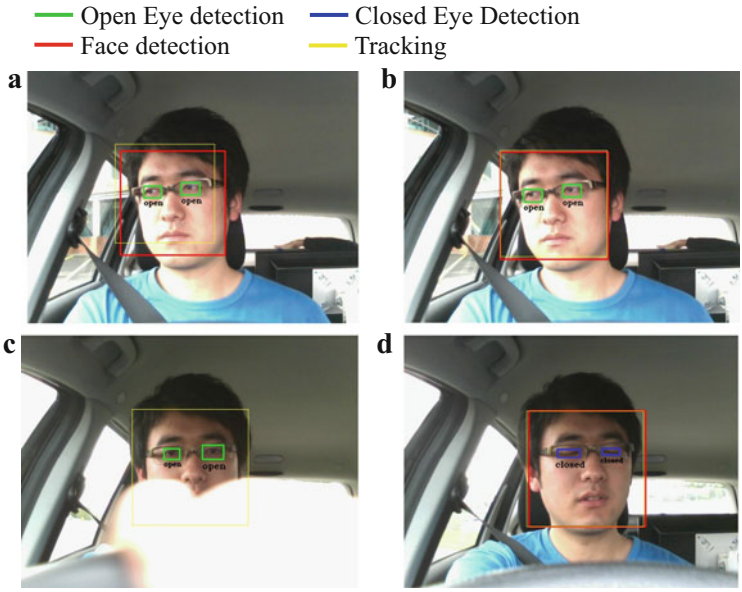

Fig. 5.20 *Face*, *open eye*, and *closed eye* detection and tracking while driving in daylight

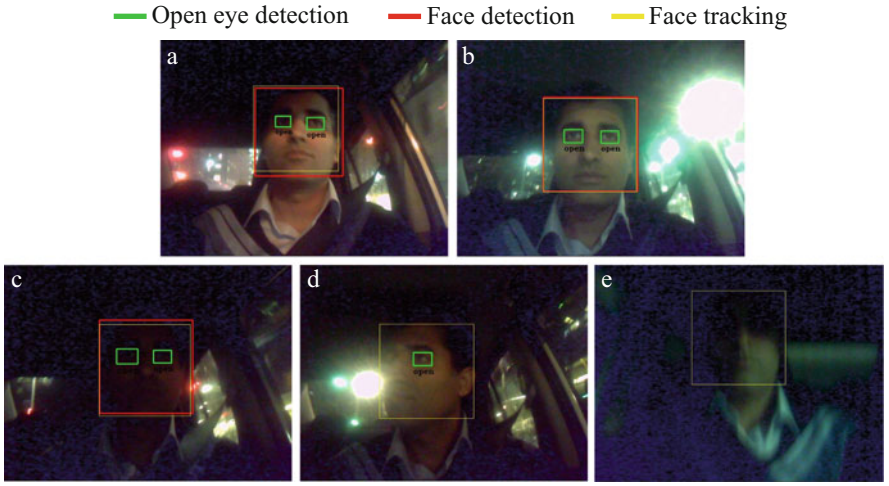

Fig. 5.21 *Face*, *open eye*, and *closed eye* detection and tracking at night under difficult lighting. (**e**) detection failure due to motion blur; yet still robust tracking

Figure 5.21 shows very good results for eye detection and tracking at night in the presence of sharp back-lights, strong shades, and very dark conditions.

While a standard Haar-like face detector performs acceptably under ideal lighting conditions [311], it underperforms under challenging illumination conditions, even

for frontal face detections. We observed that a phase-preserving denoising, followed by the proposed dynamic global features, can lead to considerable improvements in the true detection rate, in comparison to that obtained using a standard Haar-like classifier.

In order to validate the methods proposed in this chapter, we designed four classifiers using two preprocessing methods and different combinations of the local, global and dynamic global features. The first detector (*VN + Standard Local Haar*) was trained based on variance-normalized samples using only the standard local Haar-like features. The second classifier (*VN + DGHaar*) was also trained from variance-normalized preprocessed samples, but this time by using both local and dynamic global Haar features. The third detector (*PPD + GHaar*) was trained using local and global features, and the training dataset was enhanced by phase-preserving denoising techniques. The last detector (*PPD + DGHaar*) used local and dynamic global features, also based on the denoised training dataset.

For training all four classifiers, we used a large dataset of 10,000 face samples from different ages, genders, races, and nationalities, mainly from the AR [6] and Yale [300] datasets.

In addition, 50,000 non-face samples were considered as negative samples to train each stage of the cascade. Non-face samples were selected randomly from a dataset of scene categories [128]. This dataset consists of 15 natural scene categories, including buildings, offices, roads, and landscapes.

We evaluated the performance of the four classifiers in terms of the number of features involved at each stage, speed, and precision rate.

Figures 5.22, 5.23, 5.24 and 5.25 depict the distribution graphs of local and global features for each of the four mentioned classifiers.

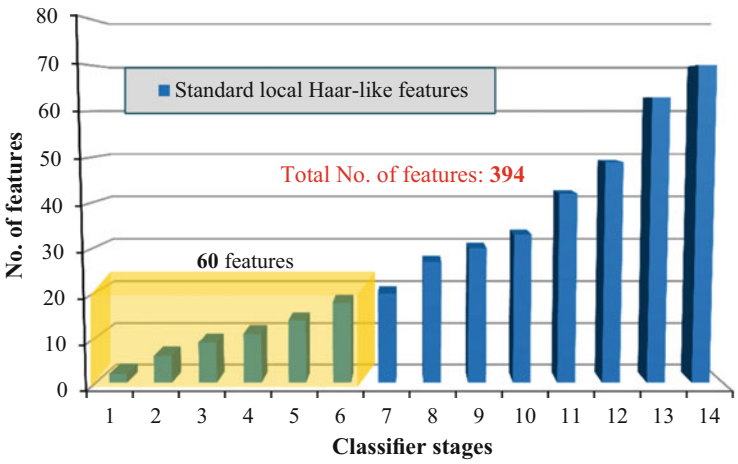

Fig. 5.22 Distribution of features for the trained classifier, based on the variance normalized dataset and standard (local) Haar features (*VN + Standard local Haar*)

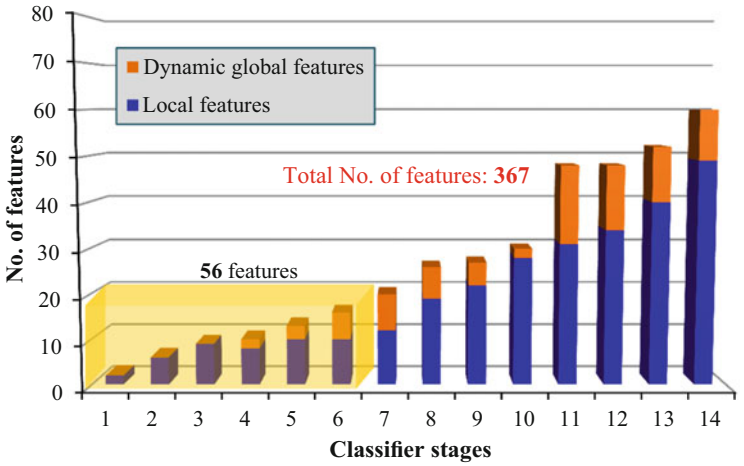

Fig. 5.23 Distribution of features for the trained classifier, based on the variance normalized dataset, local Haar features, and dynamic global Haar features (*VN + DGHaar*)

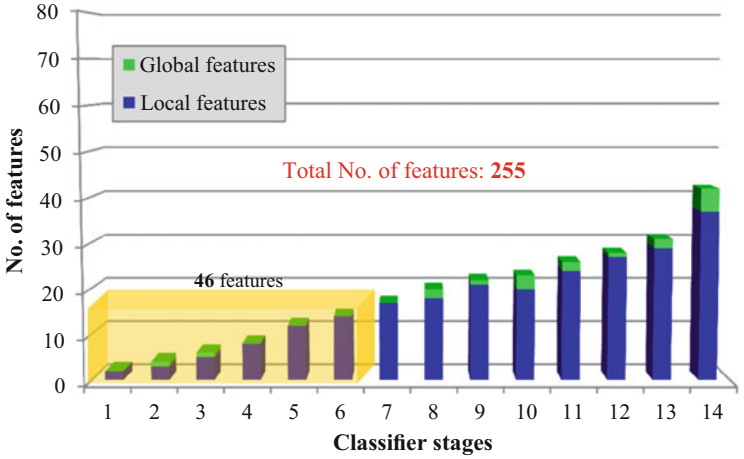

Fig. 5.24 Distribution of features for the trained classifier, based on the denoised dataset, local Haar features, and global Haar features (*PPD + GHaar*)

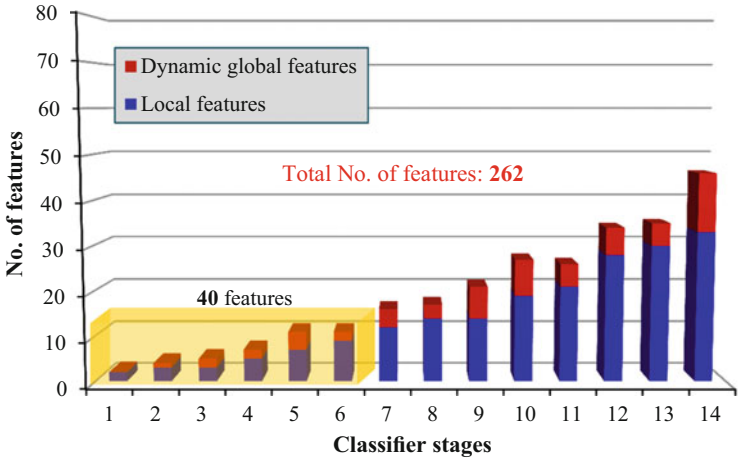

Fig. 5.25 Distribution of features for the trained classifier, based on the denoised dataset, local Haar features, and dynamic global Haar features (*PPD + DGHaar*)

Figure 5.22 is related to the first classifier (*VN + Standard Local Haar*) and shows a total number of 394 features in a cascade of 14 weak classifiers. The graph shows that the first classifier involves the highest number of features, compared to three other classifiers. This also means a higher computational cost for the detection process.

Figure 5.23 represents a faster classifier with a considerably smaller number of local features and a total number of 367 features, including both local and dynamic global features. The classifier was trained based on the variance normalized dataset (*VN + DGHaar*). Figures 5.24 and 5.25 show feature distributions when we trained the classifiers with a phase-based denoised dataset. While the total number of features in the graphs shown in Figs. 5.24 and 5.25 are very close to each other (255 and 262, respectively), the Classifier *PPD + DGHaar* outperforms the other three classifiers as explained below:

Considering a 50% rejection rate for each stage (each weak classifier), 98.4% of non-face images will be rejected within the first six stages ($\sum_{n=1}^{6} 0.5^n = 0.984$). Thus, having the minimum number of features in the first six stages plays a very crucial role (i.e. the smaller the number of features, the faster the classifier). As the graphs show, the classifier *PPD + DGHaar* uses only 40 features in its first six stages, while the classifiers *VN + Standard local Haar*, *VN + DGHaar* and *PPD + GHaar* involve 60, 56 and 46 features, respectively, in the six early stages. This means the *PPD + DGHaar* classifier is performing 50, 40 and 15% faster than the other three classifiers, respectively.

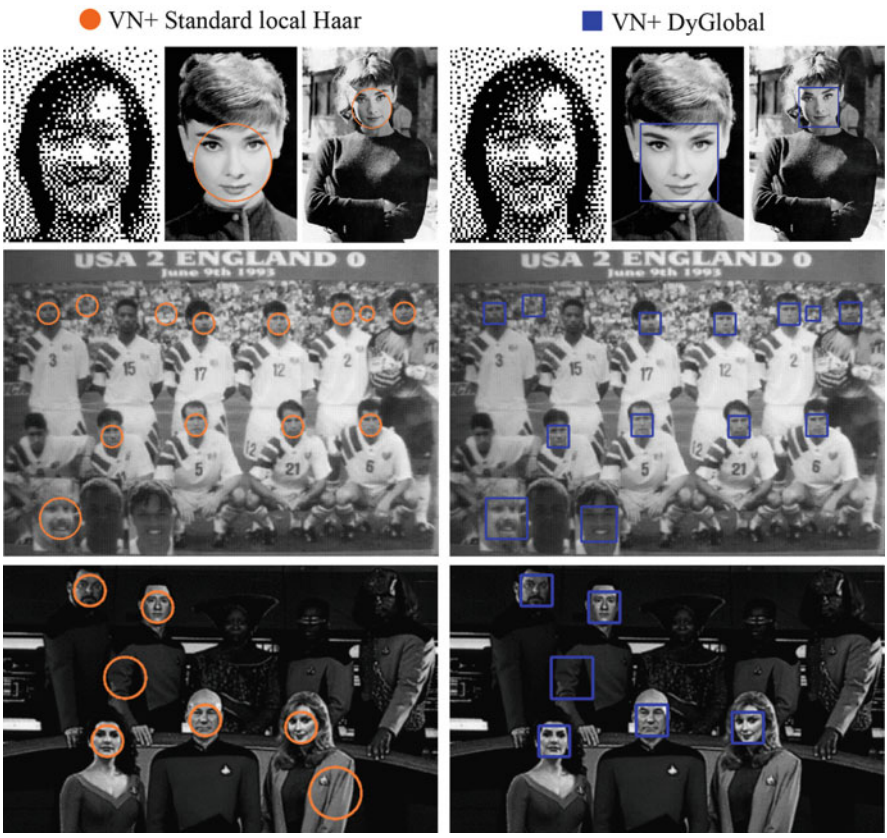

Fig. 5.26 Sample detection results (on low-quality images) for the first two classifiers trained based on standard Haar features (*orange circles*) or dynamic global features (*blue squares*), using a variance-normalized dataset

In addition to computational cost, we also analyzed the effectiveness and accuracy of the four classifiers in terms of recall rate and false alarm.

Figures 5.26 and 5.27 show detection results for the MIT-CMU test images that were not used for training of the classifiers. The figures illustrate that the trained classifier based on the dynamic global classifier and phase-based denoising provides more accurate results.

Figure 5.28 illustrates the *receiver operating characteristic* (ROC) curves for the four mentioned detectors evaluated on the MIT-CMU [33] dataset. The graphs show that the best results were obtained using the *PPD + DGHaar* detector. It can also be observed that *PPD + GHaar* performs better compared to the classifiers that only use local features.

Fig. 5.27 Sample detection results for the last two classifiers trained based on global features (*red circles*) or dynamic global features (*green squares*), using the proposed denoised dataset

Pursuing further evaluations, Table 5.4 provides a comparison between the proposed *PPD + DGHaar* classifier, standard Viola–Jones, as well as four other state-of-the-art face detection techniques.

The overall results confirm that the proposed method not only incorporates a smaller number of features (thus, faster operation), but also outperforms the others in terms of a higher detection rate and a lower number of false-positives. This being achieved by (1) consideration of both local and global intensity information within the sliding windows; (2) the denoised training dataset; (3) the training strategies proposed in Sects. 5.2, 5.3 and 5.5.

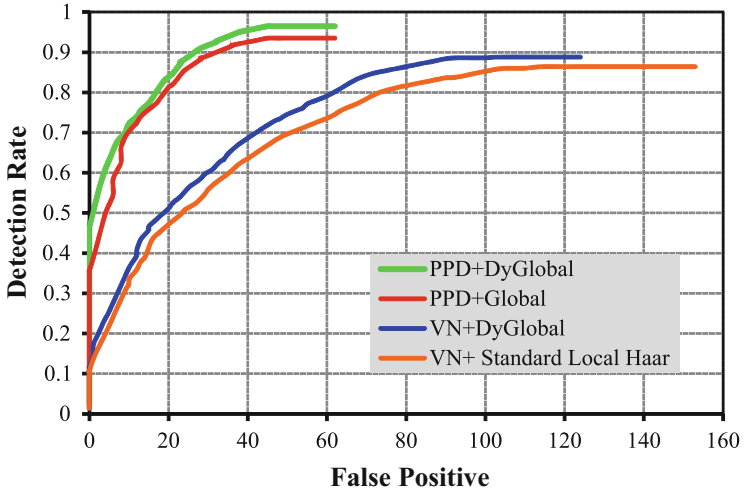

Fig. 5.28 ROC curve of the proposed detectors on the MIT-CMU dataset. *PPD+DGHaar* denotes the classifier trained based on denoised samples using the local and dynamic global features. *PPD + GHaar* denotes the classifier trained based on denoised samples using local and global features. Similarly, the other two detectors are trained by variance normalized datasets

Table 5.4 Comparison between the proposed *PPD+DGHaar classifier* and five other state-of-the-art face detection techniques

Method	Detection rate (%)	False positives (#)
Proposed method	96.4	62
Viola and Jones [271]	86.3	153
RCBoost [225]	94.0	96
Pham [194]	91.0	170
Opt [226]	95.0	900
Chain [297]	93.0	130

5.11 Concluding Remarks

The chapter considered four parameters "speed", "noise", "lighting condition" and "tracking" as the important factors in any classification problem. First, we introduced an adaptive framework in order to obtain a fast and efficient facial feature tracking under difficult lighting, clearly performing better than standard Viola–Jones classifiers. While the *tracking module* minimized the search region, the *adaptive module* focused on the given minimized region to adapt the SWS, SF and MNN parameters depending on regional intensity changes in eye-pair surrounding areas. Both modules recursively improved each other; the face tracking module provides an optimum search region for eye detection, and in turn, the adaptive eye detection module provides geometrical face location estimation to support a Kalman tracker,

in case of a temporary tracking failure. This led to a six times faster processing and more accurate results using only a low-resolution VGA camera, without application of IR light, or any pre-processing or illumination normalization techniques. We recommend the proposed method to be considered as an amendment for other Haar-based object detectors (not limited to face or eye-status detection) to gain detection improvements under challenging environments.

We investigated further the effect of a preprocessing technique on the training image sets used for boosting-based detectors. Phase-preserving denoising of images is used to pre-process the input images. We also proposed two new types of Haar-like features called "global" and "dynamic global" Haar features, for the first time. These types of features supported faster calculations than local Haar features and provided a new level of information from the query patches.

Four distinct cascades were trained with and without using the denoised images, and with and without global Haar features. Finally, the resulting detection rates and the false-alarm rates proved a significant advantage for the proposed technique against state-of-the-art face detection systems, especially for challenging lighting conditions and noisy input images.

The techniques proposed in this chapter are expected to be effective and applicable for various object detection purposes.

Chapter 6
Driver Inattention Detection

6.1 Introduction

Face detection is arguably still difficult for extreme head poses or challenging lighting conditions [294]. A real-world ADAS needs to understand and update the driver's behaviour over time (e.g. by analyzing facial features, or by steering-wheel motion analysis). The system also needs to detect potential hazards on the road. The requirement of simultaneous in-out monitoring (i.e. driver and road) requires concurrent object detection, tracking, and data fusion, all in real-time and with a high level of precision.

In this chapter, we introduce solutions for estimating the driver's head pose, to assess the driver's direction of attention, yawning detection, and head nodding based on two novel ideas of *asymmetric appearance modelling* (ASAM), and what we call the *Fermat-point transform*.

Using monocular vision only, we keep the system as low computation as possible, while the high accuracy of the proposed methods enables us to compete with state-of-the-art techniques. To the best of our knowledge, no previous research has jointly addressed all of the above-mentioned subjects in one integrated real-time solution.

Jian-Feng et al. [102] propose driver-fatigue detection using the same standard *active appearance model* (AAM) introduced by Cootes [35] by fitting it to an eye region, followed by head-gesture detection based on the face-centroid. The method appears to be too basic to be applicable in highly-dynamic real-world scenarios.

Mosquera and Castro [254] use a recursive algorithm to improve the convergence accuracy of driver's face modelling. Results show improvements compared to Cootes' AAM method [35]; however, it does not take the driver's facial features into account.

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_6

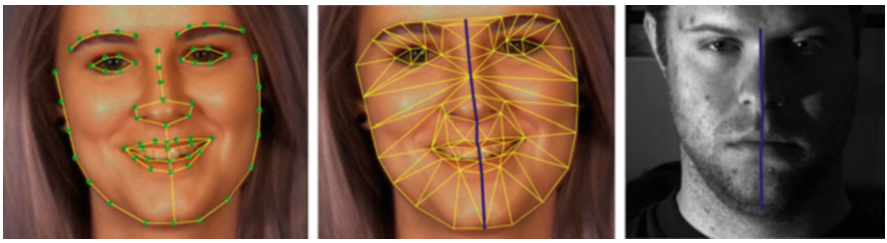

Fig. 6.1 Sixty-four keypoint landmarks (*left*). Symmetric Delaunay triangulation (*middle*). Asymmetric intensity variations (*right*)

Chutorian and Trivedi [175] propose a method to monitor driver's activity by using an array of Haar-wavelet AdaBoost cascades for initial face detection, and by applying localized gradient orientation (LGO) as input for support vector regressors. The method uses a rigid facial-mesh model to track the driver's head. There is a general weakness here as the tracking module may easily diverge for face shapes that are very different from the reference "mesh model".

Visage Technologies® provides a state-of-the-art head tracker [273] based on a feature-point detection and tracking of nose boundary and eye regions. Despite obtaining accurate results under ideal conditions, their tracking fails in the presence of noise and non-ideal conditions.

Krüger and Sommer use Gabor wavelet networks [126] for head-pose estimation. Claimed advantages cover, for example, independence to affine deformations and high-precision of the algorithm for *any* desired input; also the input may range from a coarse representation of a face to an almost photo-realistic subject. Nevertheless, the experimental results are not backed-up by a validation or comparison with other techniques.

In this chapter, we provide solutions for two important challenges that have been addressed insufficiently so far: **(A)** How to deal with intensity asymmetry and unbiased illumination for the same object, such as a driver's face (Fig. 6.1, right)? **(B)** How to map one generic 3D face model into various deformable faces (Figs. 6.7, 6.8, and 6.9)?

The next sections of this chapter are organized as follows: Sect. 6.2 proposes an asymmetric shape- and appearance-modelling technique to retain the shape and appearance of an input face of an unknown driver. Section 6.3 introduces an improved technique for 2D to 3D pose estimation based on the standard EP*n*P technique and the *Fermat-point transform* (for the first time), that together lead to a highly accurate head pose estimation. Section 6.4 provides further experimental results. Apart from analyzing the selected feature points obtained from the previous shape modelling stage, this section also discusses yawning and head-nodding detection. Section 6.5 summarizes the chapter with concluding remarks.

6.2 Asymmetric Appearance Models

Appearance models (AM), as originally introduced by Cootes et al. [35], are widely used for object modelling, especially in the context of face processing. In order to define a face-AM, we need to train a variation of face shapes (the *shape model*) and variation of face intensities (the *texture model*). In other words, we need to combine a model of shape variation with a model of intensity variation. Rather than tracking a particular object (or a particular face) with a particular pose (or expression), an *active appearance Model* (AAM) iteratively tries to refine the trained model with the target image, in order to reduce the matching errors and to lead to a best model match. Current research addresses the underlying optimization problem (to find an improved fitting algorithm) and the reduction of matching errors.

6.2.1 Model Implementation

Considering 64 point-landmarks, as illustrated in Fig. 6.1, left, and using the *MUCT* face dataset [172], we create an annotated face dataset in order to train a generic face-shape model. Following the standard AM approach [35], and applying a uniform coordinate system, a vector $\mathbf{f} = [x_0, y_0, \ldots, x_i, y_i]^\top$ represents an annotated face. A face-shape model is defined as follows:

$$\mathbf{f} = \bar{\mathbf{f}} + \mathbf{P}_s \mathbf{b}_{s_i}, \tag{6.1}$$

where $\bar{\mathbf{f}}$ is the mean face shape after applying a *principal component analysis* (PCA) on the available annotated face dataset, $\mathbf{P}_s$ is an orthogonal matrix of face-shape variations, and $\mathbf{b}_s$ is a vector of face-shape parameters (given in distance units).

By applying a translation (t_x, t_y) and a rotation with scaling ($s_x = s \cdot \cos\theta - 1$, $s_y = s \cdot \sin\theta$), each sample face is warped into the mean shape model, thus creating a new face F. Let $F = S_t(f)$ be this warped image, where S_t is the warping function, and $\mathbf{t} = [s_x, s_y, t_x, t_y]^\top$ is the pose parameter-vector.

Figure 6.2 illustrates the steps for creating the *appearance face model* based on just two sample faces. The second row of Fig. 6.2 shows examples of shape variations with different deformation parameters applied on each sample face, based on the chosen 64 point-landmark strategy. The blue shape shows the obtained mean *face-shape model*.

To create a face texture model (intensity model), first a symmetric *Delaunay triangulation* is applied on shape feature points for each sample face (Fig. 6.1, middle). Considering $\mathbf{g}$ as a texture vector of a sample face image, similar to the shape-warping stage, we have a mapping $\mathbf{g} \rightarrow \mathbf{g}^*$, where $\mathbf{g}^*$ is generated after scaling and adding an offset to the current intensity vector $\mathbf{g}$. In this way we create a shape-free "intensity patch" for each sample face given in the training dataset. This is done by raster scanning into the texture vector $\mathbf{g}$, and a linear normalization of $\mathbf{g}$

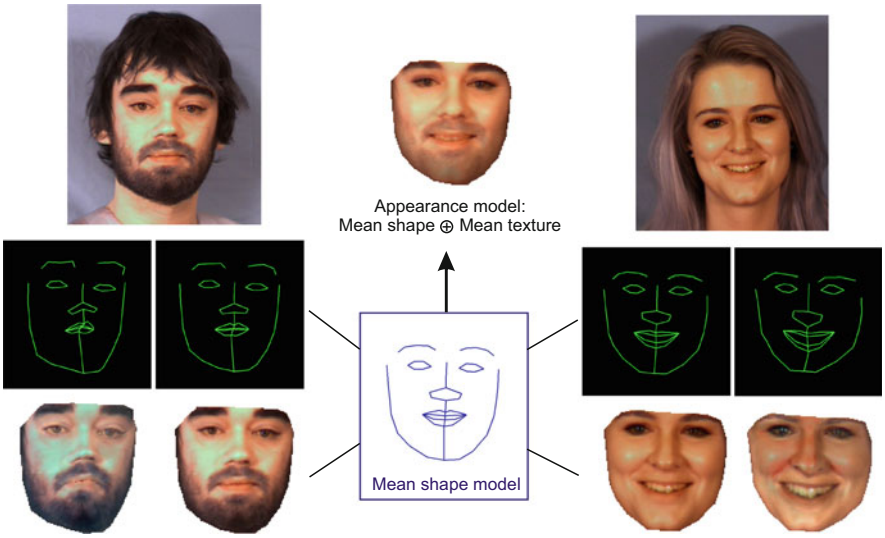

Fig. 6.2 Conversion of face *shape* and face *texture models* of two sample faces into a mean *appearance model*

for every half of the face as follows:

$$\mathbf{g}_L^* = \frac{[\mathbf{g}_L - \boldsymbol{\mu}_L, \mathbf{1}]^\top}{\sigma_L} \quad , \quad \mathbf{g}_R^* = \frac{[\mathbf{g}_R - \boldsymbol{\mu}_R, \mathbf{1}]^\top}{\sigma_R}, \tag{6.2}$$

where $\boldsymbol{\mu}_L$, $\boldsymbol{\mu}_R$ and σ_L^2, σ_R^2 are means and variances for the left and right part of the face-intensity patch, $\mathbf{g}_L$, $\mathbf{g}_R$ are the left and right half of the $\mathbf{g}$ vector, $\mathbf{g}_L^*$, $\mathbf{g}_R^*$ are normalized data, and $\mathbf{1}$ is a vector of ones (for filling in the remaining positions). After normalization we have that $\mathbf{g}^{*^\top} \cdot \mathbf{1} = 0$ and $|\mathbf{g}^*| = 1$.

As part of the *asymmetric appearance models* (ASAM), Eq. (6.2) considers individual asymmetric intensity normalization for the left and right half of a face. This is a simple but important step that can help to prevent divergence of face-shape matching due to cumulative intensity errors. Figure 6.1, right, shows how face intensity can vary depending on the light source location and due to the nose bridge. This is a very common case in applications such as driving scenarios, where one side of the face appears brighter than the other side.

Similar to the shape-model case, by applying PCA on the normalized intensity data, a linear face intensity model is estimated as follows:

$$\mathbf{g}_L = \bar{\mathbf{g}}_L^* + \mathbf{P}_{\mathbf{g}_L}\mathbf{b}_{\mathbf{g}_L} \quad \mathbf{g}_R = \bar{\mathbf{g}}_R^* + \mathbf{P}_{\mathbf{g}_R}\mathbf{b}_{\mathbf{g}_R}, \tag{6.3}$$

where $\bar{\mathbf{g}}^*$ is the mean vector of normalized grey-level or intensity data, $\mathbf{P}_\mathbf{g}$ is an orthogonal matrix of texture-modes of variations, and $\mathbf{b}_\mathbf{g}$ is a vector of intensity

parameters in grey-level units (Fig. 6.2, third row). We apply this as an individual process for each half of the face.

The shape and texture of a face can therefore be summarized by $\mathbf{b}_s$ and $\mathbf{b}_\mathbf{g}$. Since there is some correlation between shape and intensity information, a combined AM is considered. For each sample face, a concatenated vector $\mathbf{b}$ is defined as follows:

$$\begin{bmatrix} \mathbf{W}_s\mathbf{b}_s \\ \mathbf{b}_\mathbf{g} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_s\mathbf{P}_s^\top(\mathbf{f}-\bar{\mathbf{f}}) \\ \mathbf{P}_\mathbf{g}^\top(\mathbf{g}-\bar{\mathbf{g}}^*) \end{bmatrix}, \tag{6.4}$$

where $\mathbf{W}_s$ is a diagonal matrix which defines appropriate weights for the concatenation of $\mathbf{b}_s$ and $\mathbf{b}_\mathbf{g}$ at places where they have different units (one comes with the units of distance, and the other one comes with the units of intensity). The RMS change in $\mathbf{g}$, per unit change in $\mathbf{b}_s$, is considered to define appropriate weights $\mathbf{W}_s$ for Eq. (6.4). This makes $\mathbf{b}_s$ and $\mathbf{b}_\mathbf{g}$ proportional. Applying another PCA to these vectors, the AM is given as

$$\mathbf{b} = \mathbf{Q}\mathbf{c}, \tag{6.5}$$

where $\mathbf{c}$ is the vector of parameters for the *combined* AM which unifies shape and intensity models:

$$\mathbf{f} = \bar{\mathbf{f}} + \mathbf{P}_s\mathbf{W}_s\mathbf{Q}_s\mathbf{c} \quad , \quad \mathbf{g} = \bar{\mathbf{g}}^* + \mathbf{P}_\mathbf{g}\mathbf{Q}_\mathbf{g}\mathbf{c}. \tag{6.6}$$

Matrix $\mathbf{Q}$ is subdivided as follows:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_s \\ \mathbf{Q}_\mathbf{g} \end{bmatrix}. \tag{6.7}$$

6.2.2 Asymmetric AAM

Reviewing the Cootes et al. method [35], the *active appearance model* (AAM) refers to an active search and refinement process to adapt a previously trained face-AM into an unknown face; and with *asymmetric AAM* (ASAAM) we process a face as an asymmetric object.

Let us assume we would like to interpret the facial features of an unknown driver. After finding the initial position of a 2D face in the input frame we aim to adjust the model parameters c to match the *trained AM* to an unknown input face as closely as possible.

Model matching is a crucial step in our algorithm, as all the next stages including *head-pose estimation*, *head-nodding* and *yawning* detection, can directly be affected by the accuracy of our model matching methodology.

Algorithm 2 Iterative search and model refinement

1: Use $\mathbf{g}_s = \mathbf{T}_u^{-1}(\mathbf{g}_{im})$ to project the texture sample frame into the texture model frame.
2: Calculate the current (initial) error, $E_0 = |\mathbf{r}|^2 = |\mathbf{g}_s - \mathbf{g}_m|^2$ for face halves.
3: Evaluate the predicted displacements based on the RMS method, $\delta\mathbf{p} = -\mathbf{R} \cdot \mathbf{r}(\mathbf{p})$, where $\mathbf{R}$ is the matrix of texture sample points and the model parameter is $\mathbf{R} = \left(\frac{\delta\mathbf{r}^\top}{\delta\mathbf{p}} \cdot \frac{\delta\mathbf{r}}{\delta\mathbf{p}}\right)^{-1} \frac{\delta\mathbf{r}^\top}{\delta\mathbf{p}}$.
4: Set $k = 1$.
5: Set $\mathbf{p} = \mathbf{p} + k\delta\mathbf{p}$ to update the parameters of the model.
6: Calculate F' and $\mathbf{g}'_m$ as the new points, and new texture model frame, respectively.
7: Sample the face at F', so as obtain a new estimate of $\mathbf{g}'_{im}$.
8: Evaluate a new error vector, $\mathbf{r}' = \mathbf{T}_{u'}^{-1}(\mathbf{g}'_{im}) - \mathbf{g}'_m$; therefore the new error $E_1 = |\mathbf{r}'|^2$.
9: **if** $E_1 < E_0$ **then**
10: Accept the last estimate,
11: **else**
12: Set $k = k/2$,
13: Go to Step 5; repeat until no further decrease for $|\mathbf{r}'|^2$
14: **end if**

Before proceeding with model initialization, we need the driver's face location. We use our proposed GHaar features and classifier from Sect. 5.8 and in [216]. The classifiers can return a robust face detection and localization even under challenging lighting conditions. Having model parameters $\mathbf{c}$, and shape-transformation parameters $\mathbf{t}$, the rough position of the model points F can be calculated on the image frame, which also represents the initial shape of the face patch.

As part of the matching process, pixel samples $\mathbf{g}_{im}$ from the region of the image are taken and projected to the texture model frame, $\mathbf{g}_s = \mathbf{T}_u^{-1}(\mathbf{g}_{im})$. Given the current texture model $\mathbf{g}_m = \bar{\mathbf{g}^*} + \mathbf{Q}_\mathbf{g}\mathbf{c}$, the difference between the current image frame and the current model is:

$$\mathbf{r}(\mathbf{p}) = \mathbf{g}_s - \mathbf{g}_m, \tag{6.8}$$

where $\mathbf{p}$ is the parameter vector of the model:

$$\mathbf{p}^\top = (\mathbf{c}^\top | \mathbf{t}^\top | \mathbf{u}^\top). \tag{6.9}$$

Applying RMS and measuring residual errors, the model parameters can be gradually refined. This can be seen as an optimization approach in which a few iterations lead to smaller residuals, thus to the best match of the model with the input face. Starting from a current estimation for appearance model parameters $\mathbf{c}$, at position $\mathbf{t}$, texture transformation $\mathbf{u}$, and a face example with the current estimate as $\mathbf{g}_{im}$, the iterative algorithm is summarized as Algorithm 2.

Experimental results show that after 3 to 5 iterations, the ASAAM method rapidly converges to the actual face image. Figure 6.3 shows an example of an inaccurate model fitting by the standard AAM, and an improvement by the *asymmetric active appearance model* (ASAAM).

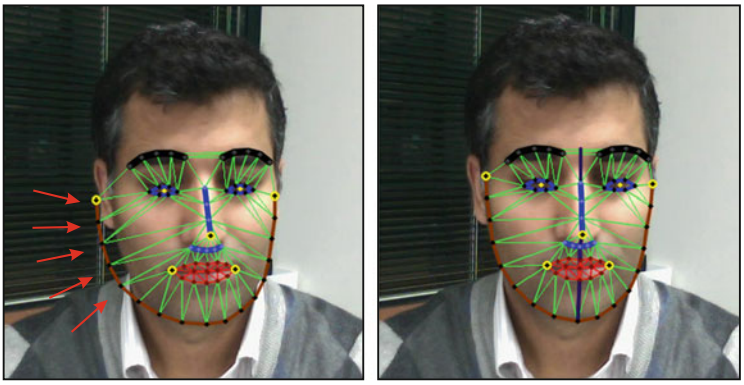

Fig. 6.3 (*Left*) An inaccurate model fitting on the left face-half using a standard AAM method. (*Right*) a perfect model matching for both face halves using ASAAM

6.3 Driver's Head-Pose and Gaze Estimation

In the previous section we described tracking of the driver's face-shape model by localization of the key feature-points.

This section introduces driver's pose and gaze estimation in six degrees of freedom, based on mapping 2D feature-points into a 3D face model. The section also introduces the *Fermat-point transform*, for the first time.

We describe a *conical gaze-estimation method* as the preliminary step to analyze the driver's direction of attention and its correlation with the location of detected hazards (e.g. vehicles) on the road.

Different approaches have been proposed such as pose detection from orthography and scaling (POS), or POS with iteration (POSIT) [45, 77, 153], 3D morphable models [295], or more recent research such as application of randomized decision forests [238], or multi-view based training approaches [294].

All of the above-mentioned work, even the most recent one, only considers a generic 3D object model, or a set of convex polygons to create a model-reference for pose detection. However, regardless of the pose-detection methodology and 3D model specification, the *matching error* of a 3D model with a query object has not been addressed so far.

This section contributes on both issues: Selection of an optimum generic 3D model, and proposing a solution to minimize the 2D to 3D matching error; therefore a more accurate and faster pose estimation can be achieved.

In the next two sub-sections, we provide a trigonometry-based solution to reduce pose-estimation errors that happen due to differences between the 3D face model and the 2D face-shape variation among different people (drivers).

6.3.1 Optimized 2D to 3D Pose Modelling

Figure 6.4 shows the pinhole-camera model [45] with calibrated focal length f, center of projection O, axes Ox, Oy and Oz, and unit vectors i, j and k in camera coordinates. In the 3D object plane, we have a face model with feature points F_1, F_2, ..., F_n. The coordinate frame for the face reference is centred at F_0, and it is specified by (F_0u, F_0v, F_0w).

Based on the steps described in Sect. 6.2, the shape of the driver's face is already computed. Hence, for every feature-point F_n we already know the coordinates of (F_nu, F_nv, F_nw). Consequently, corresponding projected points $p_1, p_2, \ldots, p_n$ on the image plane are also known as (x_n, y_n). However, the coordinates of (X_n, Y_n, Z_n) on the camera frame are unknown and need to be computed. By calculating the rotation matrix and the translation vector $O \rightarrow F_0$, the driver's face-pose can be

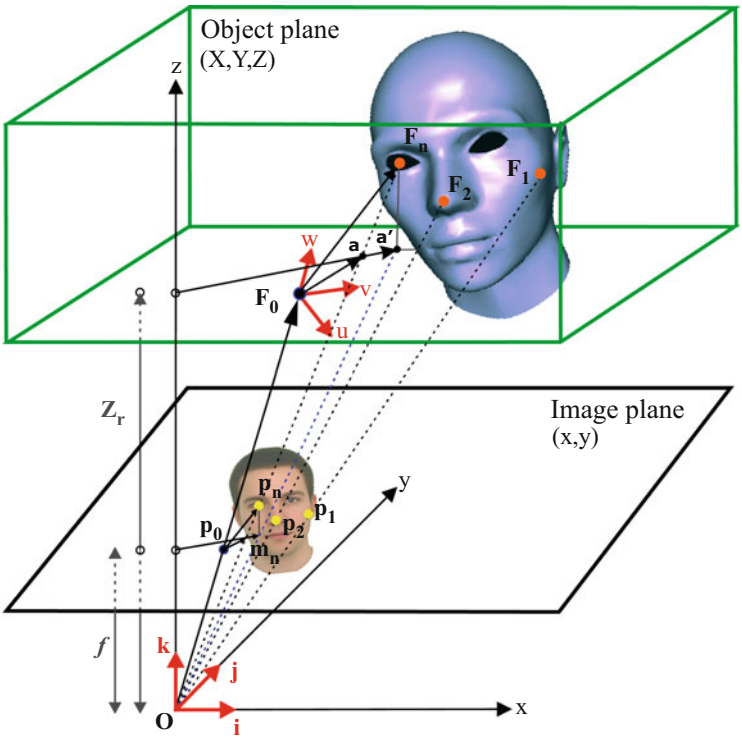

Fig. 6.4 Projection of a 3D face model M into the image plane c

defined. Combining the information derived so far, we have:

$$\mathbf{R} = \begin{bmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{6.10}$$

$$\mathbf{C} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{P} = \left[\begin{array}{c|c} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ \hline \mathbf{0}_{1\times3} & 1 \end{array}\right], \tag{6.11}$$

where $\mathbf{R}$ is the rotation matrix, $\mathbf{t}$ is the translation vector, $\mathbf{C}$ is the camera matrix, f is the focal length, (c_x, c_y) is the camera's principal point, and $\mathbf{P}$ is the pose matrix. Thus, the projection of a given object point (X_n, Y_n, Z_n) into the camera-coordinate system can be represented in the following order:

$$\begin{bmatrix} i_n \\ j_n \\ k_n \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} i_u & i_v & i_w & t_x \\ j_u & j_v & j_w & t_y \\ k_u & k_v & k_w & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \\ Z_n \\ 1 \end{bmatrix}. \tag{6.12}$$

To compute $\mathbf{R}$, we only need i and j (k is simply the cross product $i \times j$). On the other hand, according to Fig. 6.4, the translation vector $\mathbf{t}$ is equal to the vector OF_0. Having Op_0 and OF_0 aligned, it follows that

$$\mathbf{t} = OF_0 = \frac{Z_0}{f} Op_0. \tag{6.13}$$

Therefore, by calculating i, j and Z_0 (here the depth of point F_0), the pose of the face can be defined. Depth variation within a driver's facial feature-points is small, compared to the distance between camera and face (i.e. $Oa \approx Oa'$). Thus, we can approximate the depth of every point F_n as equal to Z_r. Consequently, this simplifies the previous expression, and the projection of any point (X_n, Y_n, Z_n) from the object plane to the image plane can be expressed as

$$(x_n, y_n) = \left(f\frac{X_n}{Z_n}, f\frac{Y_n}{Z_n} \right) \tag{6.14}$$

or as

$$(x_n, y_n) = \left(\frac{f}{1 + \Delta z} \frac{X_n}{Z_r}, \frac{f}{1 + \Delta z} \frac{Y_n}{Z_r} \right), \tag{6.15}$$

where $1 + \Delta z = \frac{Z_n}{Z_r}$. Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be vectors of length 4 defined by the elements in the first and second row of the matrix $\mathbf{P}$ in Eq. (6.11), respectively. Thus, the 4-dimensional vectors $\mathbf{I}$ and $\mathbf{J}$ can be defined as follows:

$$\mathbf{I} = \frac{f}{T_r}\mathbf{r}_1 \quad \text{and} \quad \mathbf{J} = \frac{f}{T_r}\mathbf{r}_2. \tag{6.16}$$

Knowing the coordinates of vectors F_0F_n in the object plane, and knowing the coordinates x_n, y_n for points p_0 and p_n in the image plane, the fundamental equations are as follows:

$$(F_0F_n)^\top \cdot \mathbf{I} = x'_n \quad , \quad (F_0F_n)^\top \cdot \mathbf{J} = x'_n \tag{6.17}$$

$$x'_n = x_n(1 + \Delta z_n) \quad , \quad y'_n = y_n(1 + \Delta z_n) \tag{6.18}$$

$$\Delta z_n = \mathbf{r}_3^\top \cdot \tfrac{(F_0F_n)}{Z_r - 1}. \tag{6.19}$$

Any given (approximated) initial value for Δz_n solves the above equations, thus the driver's face pose can be approximated.

Regarding the selection of an optimum 3D model, we perform a statistical analysis on 84 three-dimensional face models from the TurboSquid dataset [264], firstly to select the best generic 3D model, and secondly to assess depth-mean and variance of a human face in the regions of eyes, nose tip, mouth and ear-tops. Knowing f and Z_r (the distance of our camera mounted in front of the dashboard), and applying our statistical analysis mentioned above, we derive an initial value $\Delta z_n = 0.082$. Once i and j are computed, the value of Δz_n can be refined and optimized after two or three iterations. This is much faster than a blind POSIT algorithm that needs four to ten iterations to refine Δz [153].

6.3.2 *Face Registration by Fermat-Transform*

This section introduces the selection of appropriate facial-feature points and a registration technique for matching the driver's 2D face-image into a generic 3D face-model. As part of our algorithm to obtain roll, yaw, and pitch angle of the driver's face, we use the state-of-the-art method of EP*n*P by Lepetit et al. [131], in conjunction with a novel pre-processing step to minimize the mismatch errors of 2D-to-3D corresponding points.

Some of the works that use the Perspective-*n*-Points (P*n*P) method normally consider four points around the nose boundary [77, 175]. This may, in general, simplify the case to a planar problem, as the sampled feature points around the nose boundary have almost the same Y-depth in the camera-coordinate system. However,

a weakness is that those feature points cover only a limited area of the face region, which might also be affected by noise. This causes larger error values for matching the driver's face and the 3D model.

Using a five-point correspondence, we consider pose estimation of a driver's face as a P-5-P problem. Generally speaking, the method estimates the pose of the camera from n 3D points to the corresponding 2D points. The main idea is to define n 3D points as a weighted sum of four control points as below:

$$\begin{aligned} F_n &= \textstyle\sum_{j=1}^{4} \alpha_{nj} C_j^F \quad , \quad p_n = \sum_{j=1}^{4} \alpha_{nj} C_j^p \\ &\text{with} \quad \textstyle\sum_{j=1}^{4} \alpha_{nj} = 1 \quad \& \quad n = 1, \ldots, 5, \end{aligned} \tag{6.20}$$

where C_j^F and C_j^p are control points of a 3D model and the image coordinate system, respectively, and α_{nj} are homogeneous barycentric coordinates. The control points may be selected arbitrarily or aligned with the principal direction of the data. Let $\{\mathbf{i}_n\}_{n=1,\ldots,5}$, with $\mathbf{i}_n = [i_n, j_n]^\top$, be the 2D projection of reference points $\{F_n\}_{n=1,\ldots,5}$ in the 3D model. Then we have that

$$\omega_n \begin{bmatrix} \mathbf{i}_n \\ 1 \end{bmatrix} = \mathbf{A}\mathbf{p}_n = \mathbf{A} \sum_{j=1}^{4} \alpha_{nj} C_j^p \tag{6.21}$$

or

$$\omega_n \begin{bmatrix} i_n \\ j_n \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{nj} \begin{bmatrix} x_j^p \\ y_j^p \\ z_j^p \end{bmatrix}, \tag{6.22}$$

where ω_n are scalar projective parameters.

Although four sets of points in both world-coordinate and image-coordinate systems are sufficient to solve a PnP problem, we use five points (ears' top, nose tip, and mouth corners) in order to maintain both redundancy and robustness towards image noise, and to reduce the impact of potential errors from the ASAAM step. Furthermore, these five points potentially cover a wider region of the face, also with different depth values.

Before proceeding further with the EP5P solution, we propose a new point-set transform and normalization to minimize the 3D model's *matching error* with the actual face shape. The objective is to gain a more accurate pose estimation, and to avoid *model-matching divergence* and failure of the EP5P, due to residual errors.

After solving Eq. (6.12), we rescale the shape of the driver's face (obtained from the ASAAM stage) to match the points p_4 and p_5 to known corresponding points F_4 and F_5 in the 3D model (Fig. 6.5). However, due to face-shape variation, we can expect that the three remaining points (i.e. p_1, p_2 and p_3) will not exactly match to the corresponding points in the 3D model (F_1, F_2 and F_3).

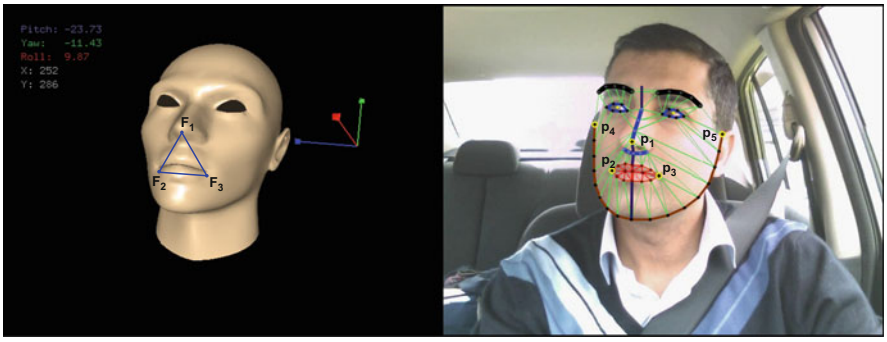

Fig. 6.5 Determining roll, yaw, and pitch of a driver's face based on ASAAM, driver's face-shape normalization, and EPnP

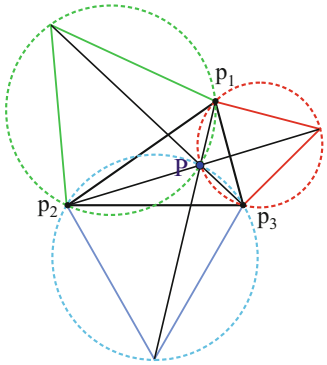

Fig. 6.6 Calculation of the Fermat point with minimum distance to the triangle's vertices

This is especially a matter of concern for the nose tip (p_1), as the nose length may highly vary from person to person. As a novel contribution in this section we adapt the Fermat–Torricelli problem [65] from the geometric world into our CV application to minimize the model matching error. Finding the *Fermat points* and the isoganic centres P1 and P2 for the triangles $\Delta F_1F_2F_3$ and $\Delta p_1p_2p_3$ (as per Fig. 6.6) and translating $p_1p_2p_3$ (with no rotation) so that P_2 matches P_1, we have that

$$P_{x,y} = \arg\min_{x,y\in\mathbb{R}} \left\{ \sum_{n=1}^{3} \sqrt{(F_x^n - p_x^n)^2 + (F_y^n - p_y^n)^2} \right\} . \tag{6.23}$$

This means that the nose-region triplets (p_1, p_2, p_3) are translated in an order such that they have the minimum total distance to corresponding triplets (F_1, F_2, F_3) in the 3D model. We call the above process the *Fermat-point transform*.

Since a vehicle's driver does not change during a driving course, we apply the same scaling and relative translation to all input faces and all the pre-calculated Delaunay triangles, with respect to the Fermat point, P. This guarantees that our *refined* face shape-model fits to our 3D model, as closely as possible, while we keep the p_4 and p_5 unchanged, at their original locations. Figure 6.5 shows the final results of driver's attention estimation based on the proposed techniques in Sects. 6.2 and 6.3.

6.4 Experimental Results

For the training stage of the proposed ASAAM model, we used a set of 7512 images from the MUCT face dataset [172], each one annotated as per the proposed 64 point-landmark approach (6.1, left), followed by 2500 pixel intensity sampling from each half of a face.

6.4.1 Pose Estimation

In addition to the sample result shown in Fig. 6.5, we performed a further analysis to verify the robustness and accuracy of the proposed method in comparison to a standard AAM.

Figures 6.7, 6.8, 6.9, 6.10, and 6.11 show the experimental results applied for different drivers, different poses, and different lighting conditions. Without the need for ground truth data analysis a visual inspection confirms a significant improvement compared to the AAM method as well as a very close match for the pose of the 3D model and the actual pose of the driver face. Regardless of the face shape, appearance, and lighting condition, we used the same generic 3D model for all the faces, and the proposed method was able to follow the driver's pose very accurately without any model divergence or failure over a video dataset of 26 min driving in day and night. Using a Core i7 PC with 8 GB RAM, the entire system was able to perform in real time, at the speed of 21 frames per second.

6.4.2 Yawning Detection and Head Nodding

Having already detected feature points for the driver's face, we can easily detect yawning as per Fig. 6.11, left, by measuring mouth openness over a continuous period defined by a time threshold (e.g. more than 1.5 s of continued wide-mouth

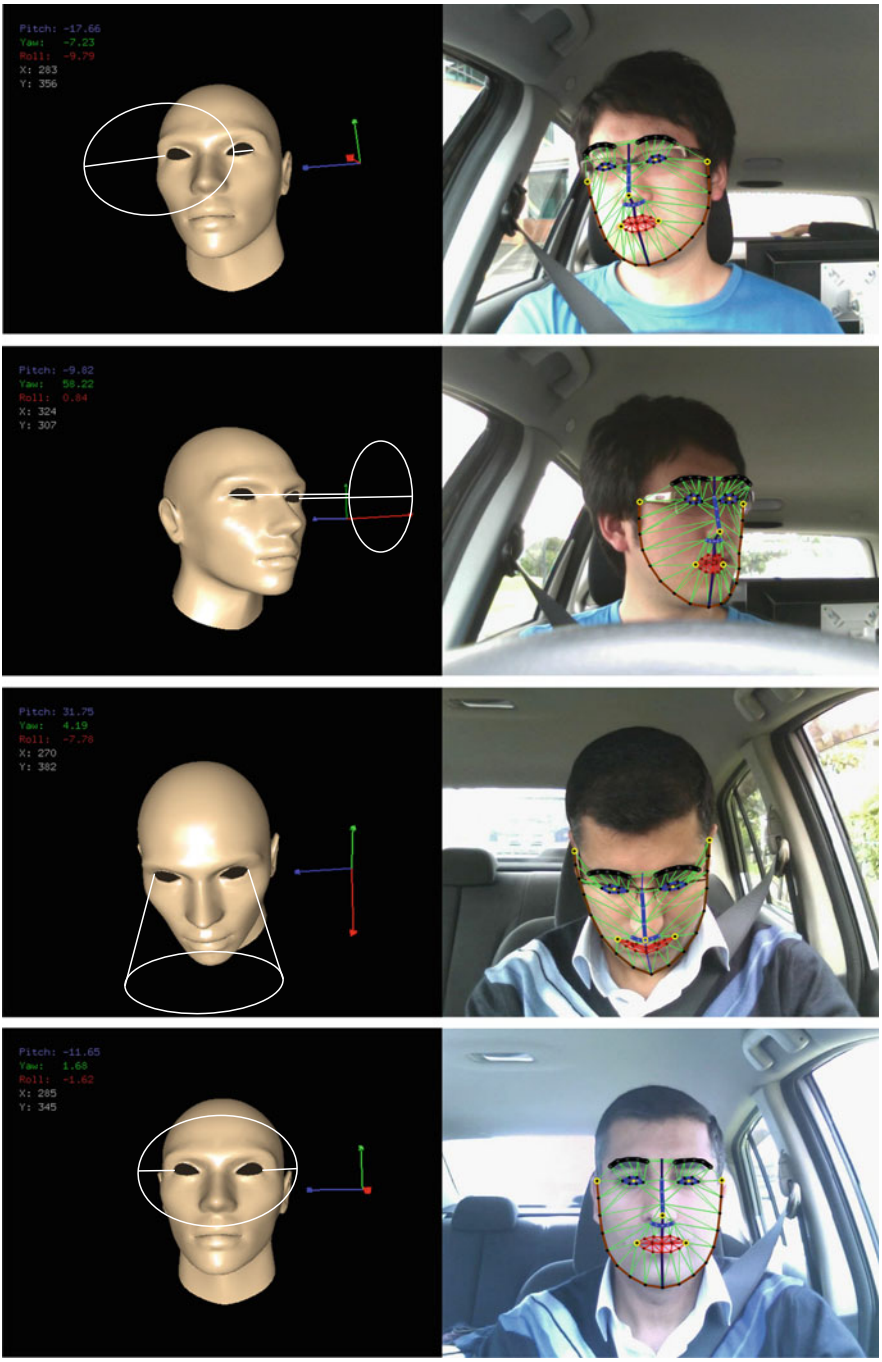

Fig. 6.7 Driver's head-pose estimation based on ASAAM, Fermat-point transform, and EP*n*P; (Daylight 1)

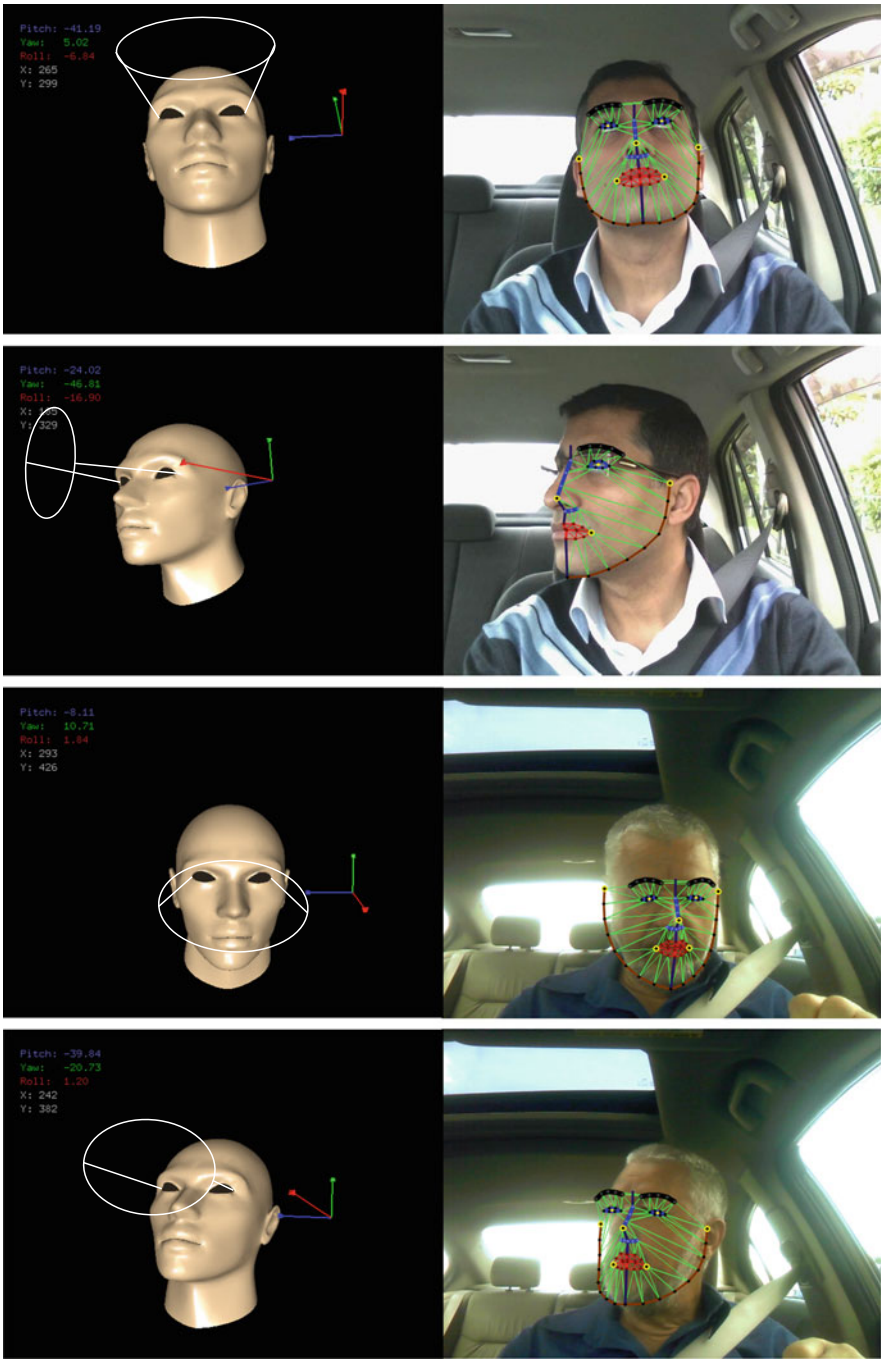

Fig. 6.8 Driver's head-pose estimation based on ASAAM, Fermat-point transform, and EP*n*P; (Daylight 2)

Fig. 6.9 Driver's head-pose estimation based on ASAAM, Fermat-point transform, and EP*n*P; (Night Condition)

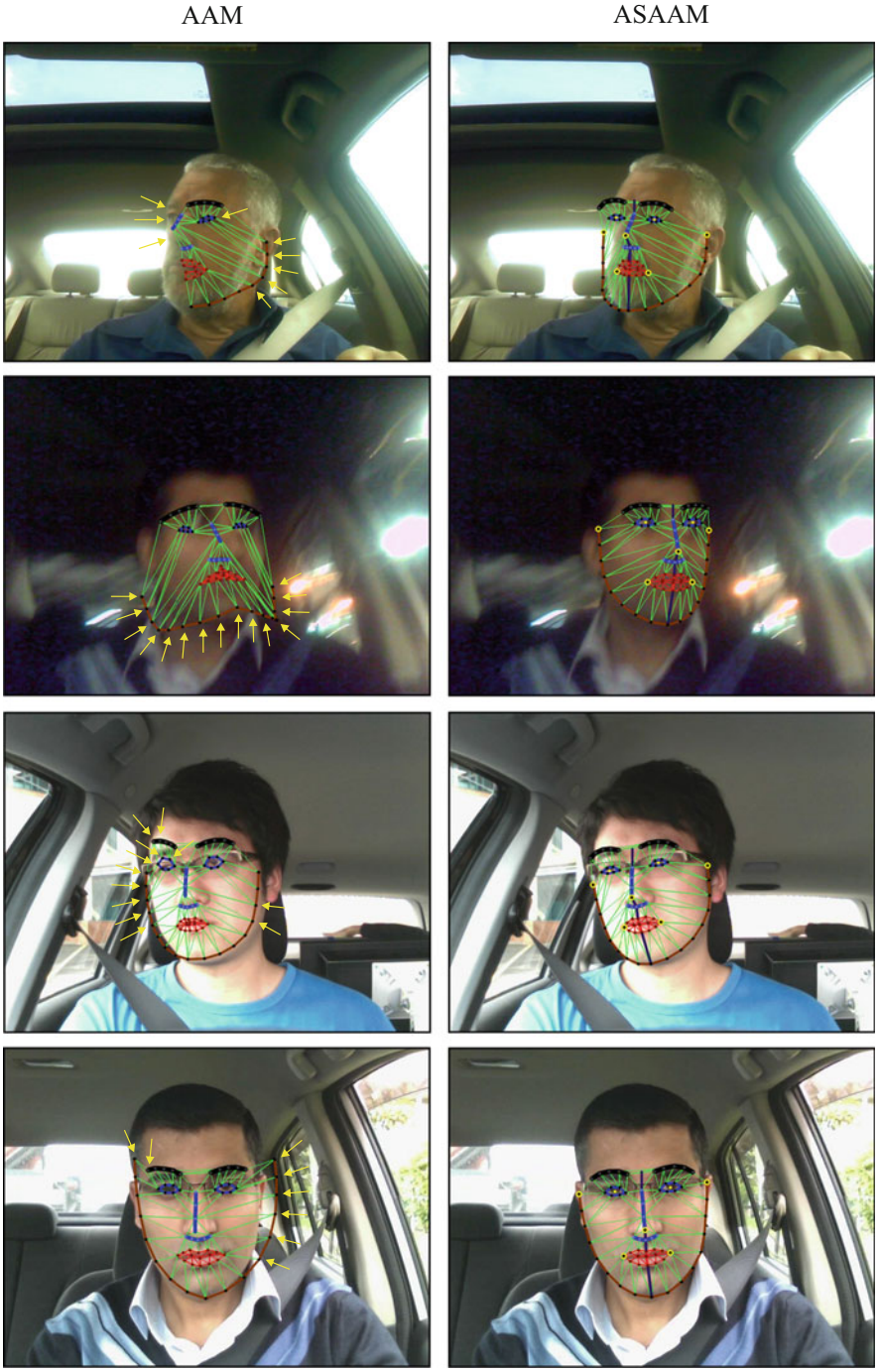

Fig. 6.10 AAM vs. the proposed ASAAM. The shape deformations due to residual errors are highlighted by *yellow arrows*

Fig. 6.11 Yawning and head nodding detection

openness):

$$f_y(t) = \begin{cases} t & \text{if } \frac{d(a,a')}{d(p_2,p_3)} \geq 0.6 \quad \text{and} \quad \Delta t \geq \tau_1, \\ 0 & \text{otherwise,} \end{cases} \tag{6.24}$$

where t is time, and Δt is a continuous time elapsed after the first mouth openness detection. Similarly for the head-nodding detection we define:

$$f_n(t) = \begin{cases} t & \text{if } \frac{d(b,b')}{d(p_2,p_3)} \geq 0.16 \quad \text{and} \quad 2.78e^{-4} \cdot V_{km/h} \geq \frac{2.5s}{\Delta t}, \\ 0 & \text{otherwise,} \end{cases} \tag{6.25}$$

where $f_n(t)$ measures the proportion of the *mouth width* to the distance of nose tip (i.e. b) to the upper lip (i.e. b') (Fig. 6.11, left). The threshold is defined based on 2.5 m of distracted driving, depending on the vehicle's speed, and the elapsed time, after the first occurrence of a head-nodding detection. Considering 0.5–0.8 s as the driver's reaction time after raising a warning alarm (Fig. 6.11, right), the system will not allow more than 4.5 m of unconscious driving.

6.5 Concluding Remarks

This chapter reviewed the latest research on head pose estimation. We identified three important weaknesses among the proposed methods, including failure of face shape modelling due to challenging lighting conditions and intensity variation in face halves; unstable pose estimation and pose divergence due to residual errors from mismatch of the face model to the generic 3D model; and slow operation of previous work due to the need to model refinement within several iterations (up to 10 iterations). We proposed methods and solutions for all three identified issues.

We introduced the ASAAM to solve the problem of intensity variation in face halves which led to more accurate face shape modelling. We also selected a generic face model and a 5-point facial feature point analysis followed by a Fermat-point transform that led to both a faster and more accurate 2D to 3D matching with lower residual errors, 3 times faster operation, and no model divergence. The method was performed in real world driving conditions and was tested on different subjects; the result proved a considerable step forward in the context of driver pose estimation.

Empirical test and analysis confirmed the accuracy and robustness of the proposed method, over a wide range of recorded videos and different subjects compared with a standard AAM. However, further analyses were not conclusive as our asymmetric 2×32 point annotation approach was not comparable with other methods (e.g. a 72 point land-marking).

Chapter 7
Vehicle Detection and Distance Estimation

7.1 Introduction

Rear-end crashes mainly occur due to driver distraction, drowsiness, or fatigue when a driver fails to keep a safe distance from the lead vehicle. According to the statistics published in 2012 for traffic safety in the USA, 28% of all traffic accidents are rear-end collisions [178]. The cited study considers 19 crash categories such as rear-end, head-on, guard-rail, crash with animal, crash with pedestrians, or rollover, plus their rate of contribution in terms of total number of accidents, fatalities, injuries, and property loss. Compared to 18 other types of collisions, rear-end crashes present the highest rate of injuries (30.9%), with a fatality rate of 5.6%, and also the highest percentage of property loss (32.9%) among all types of vehicle accidents in the USA, at the reported time.

By maintaining early vehicle detection and warning, it is possible to provide more time for a distracted driver to take an appropriate safe action to resolve driving conflicts, and consequently, decrease the possibility of rear-end crashes.

Many researchers have already developed computer vision-based algorithms to detect and localize vehicles on the road. However, most of the methods suffer from either a lack of robustness in complicated road scenes, or from a very expensive computational cost. This makes many of the developed methods unrealistic and inapplicable as a driver assistance system.

Santos and Correia [227] use a symmetry-detection technique after performing background subtraction based on an already known estimated background, using a static surveillance camera. The approach is effective for detecting vehicles in cases such as a parking lot with a pre-analyzed parking background; it is not suitable for unknown environments or roads.

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_7

Choi [32] proposes an optical flow-based vehicle-detection method; still, there are many missing detections if the relative speed between the ego-vehicle and the observed vehicle becomes close to zero, and the road has a smooth or plain texture.

Very recent work by Garcia et al. [74] proposes a fusion technique using RADAR and optical flow information. While the RADAR sensor can have multiple detections for the same vehicle, the optical flow technique can only detect overtaking vehicles with considerable velocity differences compared to the ego-vehicle, thus there is the same weakness as in the method proposed in [32].

Haselhoff et al. [85] introduces a technique using Haar and triangular features. Reported results indicate improvements compared to a standard detector using Haar features only. Nonetheless, no validation tests and experiments have been considered for night conditions or for challenging lighting situations.[1]

Huang and Barth [136], and Premebida et al. [202] fuse LIDAR data with vision sensor data. The LIDAR sensor provides high-resolution but sparse range information with a very limited object recognition ability. Then a Haar-based vehicle detector is used to process the image recorded by the camera sensor, but only within a predefined ROI calculated using LIDAR data. Such a fusion approach may increase the certainty of the detection. However, a standard Haar-based detector can easily fail in dark, noisy, or other non-ideal lighting situations. Therefore, in such cases, LIDAR data would also not help in the reported work.

Ali and Afghani [3] and Han et al. [84] provide shade-based vehicle detection. However, shades under the vehicles are not credible indicators for the existence of a vehicle. A vehicle's shadow varies in size and position; low sun causes a long shadow, in many occasions much longer than the vehicle's actual width, which also falls to the side of the vehicle. Figure 7.1 illustrates an example of inaccurate vehicle detection biased by a shadow which is falling to the left. On uneven roads (e.g. uphill) the shadow underneath a vehicle is often not visible at all.

Nguyen et al. [268] use stereo vision and a genetic algorithm; Toulminet et al. [257] use stereo vision and 3-dimensional (3D) features. Both methods take the advantage of depth information, represented in a disparity map, and apply inverse perspective mapping. However, the reported feature detection does not support accurate distinguishing of vehicles from other objects (i.e. false-positives) at night or in complicated road scenes.

Vargas et al. [267] provide a vehicle detection system using sigma-delta-based background subtraction to separate moving vehicles (foreground) from the road (background). The recording camera is fixed (i.e. it is not mounted on a mobile platform). The method is simple and computationally cost effective. It appears to be well-suited for monitoring traffic density. However, the method is not able to identify individual vehicles.

[1] We refer to challenging lighting conditions when the subject (e.g. a vehicle) is located under any non-ideal condition such as very low light, night, very bright reflections, rainy, foggy, or snowy weather, where object detection becomes very difficult, from a technical point of view.

Fig. 7.1 An example of an inaccurate vehicle detection based on underneath shadow

In this chapter, we introduce an accurate, real-time, and effective vehicle-detection algorithm to prevent imminent accidents under various conditions (described in [120] as *situations*; e.g. day, night, rain, and so forth), also dealing successfully with image noise.

This chapter is organized as follows: Sect. 7.2 presents an overview of the methodology that will be discussed in this chapter. Section 7.4 discusses line and corner feature analysis for improvement of the detection accuracy. In Sect. 7.5, a virtual symmetry detection method is introduced for taillight pairing. In Sect. 8.2, a single-sensor multi-data fusion solution is provided for vehicle detection based on the Dempster–Shafer theory. The chapter continues by utilizing detection results for distance estimation in Sect. 7.7. Section 7.8 provides experimental results and Sect. 7.9 concludes the chapter.

7.2 Overview of Methodology

One of the most important points that has been neglected in the reviewed research is that the appearance of a vehicle can highly vary depending on the distance between the observer and the vehicle. This challenge cannot be solved even by scale-invariant methods, as the shape and details of a vehicle at close distance (a few meters) is completely different to a vehicle's appearance at a distance of, for example, 100 m (Fig. 7.2).

Fig. 7.2 *Left*: Distant vehicles appearing as plain rectangles, with a shade underneath the vehicle. *Right*: A close vehicle with multiple edges, shades, and complicated features and details

Thus, relying on a one-dimensional solution for robust vehicle detection for both short and long distances appears to be hard and unrealistic to achieve. As discussed, there are many publications on general object detection or tracking approaches that are based on LBP or Haar wavelet classification; however, not many of them can be suitable for highly dynamic and real-time applications such as vehicle surveillance or monitoring. We actually need to incorporate domain specific information from road conditions or vehicles' characteristics to prevent false alarms or missing true detections.

In order to detect vehicles ahead, we use a monocular camera, mounted in the ego-vehicle, and forward-facing to the road. The objective is to detect multiple vehicles in a road scene using multiple data clues captured by a single camera. Challenges that need to be carefully considered are, for example, variation in illumination, transition from a sunny scene into shade or a tunnel, light reflections, various lights at night, and the diversity of vehicle types, makes, and models. This creates a complexity which makes feature extraction and vehicle detection extremely difficult; results are in general unstable if the applied methodologies are designed for ideal indoor conditions [99].

Figure 7.3 outlines the main idea of the proposal. In contrast to research that puts more effort into a single solution for vehicle detection, we propose a data fusion approach using edge and corner features in conjunction with an adaptive classifier based on global Haar features, called *adaptive global Haar classification* (AGHaar). This enables us to detect far-away vehicles at low resolution, in a range of about 15–100 m.

We also fuse temporal and dynamic intensity information, and a complementary technique, called *virtual symmetry detection* (VSD), that covers vehicle detection at very short distances (as close as 1 m) to the ego-vehicle, even when the recorded vehicle occupies a major area of the input frame (Fig. 7.2, right).

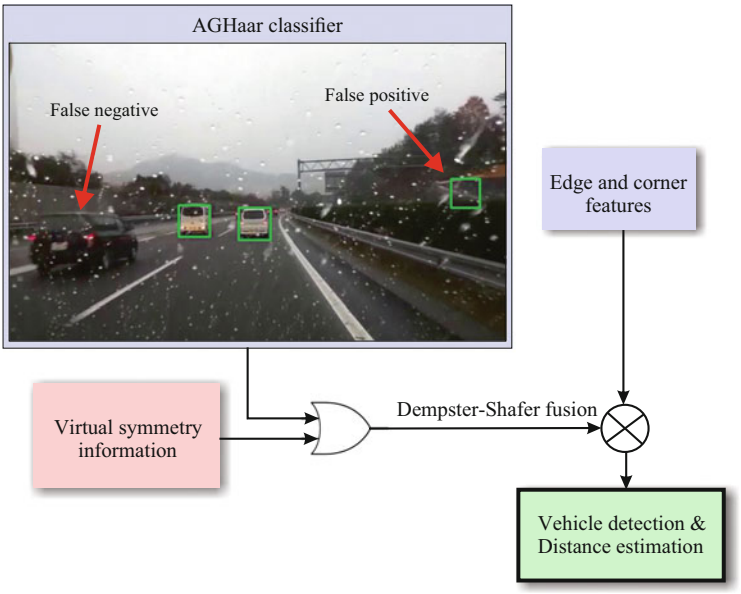

Fig. 7.3 Brief outline of the proposed detection algorithm, which combines a detection of candidate regions using *AGHaar* features with a subsequent analysis on virtual symmetry, edge, and corner features

Our algorithm is designed by following two fundamental assumptions: **(A)** the idea that regardless of a vehicles' make, model, or colour, all vehicles at a far distance have similar features and appearances in common, including occlusion edges between vehicle and road background, different light reflectance patterns on the rear windshield compared to the body of a vehicle, a tendency towards a rectangular shape, and a visible shade-bar under the vehicle's rear bumper; **(B)** for close distances, the situation is completely different; here, a vehicle shows much more details and higher resolution, with a significantly different appearance to other vehicles, different design style, different shape of bumpers, or different taillight shapes.

To the best of our knowledge, there is currently no research reported on monocular vision for the detection of very close vehicles in critical traffic scenes; for example, where a vehicle suddenly joins in at an intersection, or due to a previous occlusion. For such close-distance cases, despite the wide variety of vehicle appearances, these vehicles all still present some common features:

1. a high likelihood of a taillight pairing;
2. a constrained geometrical relationship between the size and the distance of light pairs;
3. red-colour spectrum range for taillights and brake lights.

In the next sections we detail a novel hierarchical algorithm that is capable of detecting vehicles both at far and close distances, with a substantial improvement in terms of an increase in true-positive detection rate, and a lower false-positive rate.

An *AGHaar* classifier provides initial regions and candidates of interest. The road scene and initial detections will be further processed by feature detection, taillight symmetry analysis and a final data fusion as shown in Fig. 7.3. The overall method eliminates many false-positive detections as well as retrieval of missed detections (false-negatives). We also provide an accurate distance estimation technique using a single monocular vision sensor.

7.3 Adaptive Global Haar Classifier

Extending our previous work discussed in Sects. 5.5 and 5.8 (also presented in [208] and [216]), we adopt an adaptive classifier for vehicle detection, using Global Haar features.

Figure 7.4 illustrates the concept of global features for a road scene, where we extract global intensity information for the sliding window. This can represent, for example, nearly uniform intensities on a road surface (i.e. when there is no other object shown in the reference window), or a nearly constant intensity of a vehicle (i.e. if a vehicle overlaps the reference window).

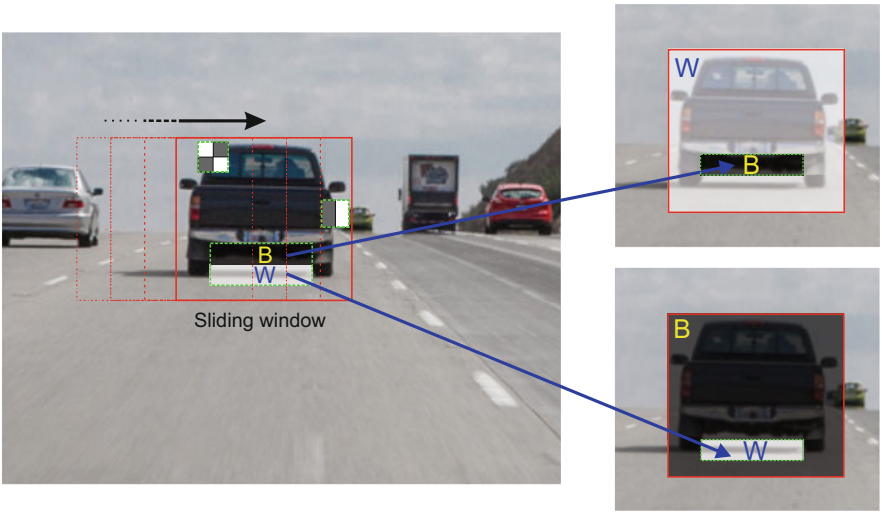

Fig. 7.4 *Left*: A sliding window with three local Haar features. *Right*: At a given window position, a local Haar feature (defined by *white* and *black regions W* and *B*) extends into two global Haar features by comparing with the sum of image values in the whole reference window

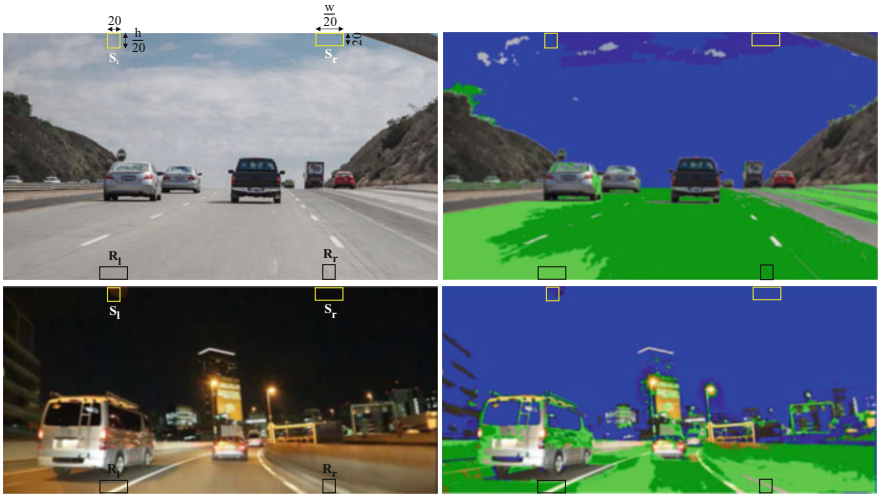

Fig. 7.5 Intensity measurements for road and sky regions under day or night conditions

In order to have an *adaptive* classifier for the intensity changes, we need to have a proper estimate of the road lighting conditions. A road is a dynamic and complex scene with different intensities due to sun, street lights, traffic lights, vehicles' lights, shades due to driving below trees, as well as shadows from moving vehicles, trees, or traffic signs. Therefore, as a preliminary requirement for a successful vehicle detection, we need to determine the conditions we are driving in, e.g. a sunny day, evening, or at night.

To assess the road-scene intensity, we cannot simply use the mean intensity of the pixels in the input frames. Figure 7.5 illustrates how we deal with intensity analysis by segmenting a road scene into two parts: "sky" and "road".

After analyzing mean intensity and standard deviation of road and sky regions from 280 samples of road scenes, taken under different weather and lighting conditions, we noticed that the top 5% of the sky region, and the bottom 5% of the road region, normally provide an acceptable intensity estimate of the whole scene, in which the measured values also fall within the scene-intensity standard deviation.

Based on this idea, we apply a 4-point intensity sampling at the expected sky and road regions, as per the defined regions S_l and S_r, and R_l and R_r shown in Fig. 7.5. We used $20 \times h/20$ and $w/20 \times 20$ patches where w and h denote the width and height of the input sequence, respectively. Then, depending on the identified lighting situation (e.g. day, night), we can adaptively adjust the classifier parameters as discussed in Sect. 5.5.3.

Since a strong reflection spot, street lights, or a very dark shade may fall in one or a few of these four patches, we applied a hybrid intensity averaging (similar to the face intensity averaging in Sect. 5.5.2) including standard *mean* and *mode* (Mo) for the road and sky region, to make sure we are measuring a balance of actual intensity in the whole scene:

$$I_s(\lambda) = \frac{1}{2}\left[\left(\lambda \cdot \mathrm{Mo}(S_l) + \frac{(1-\lambda)}{\Omega_{S_l}} \sum_{i=1}^{\Omega_{S_l}} S_l^i\right) + \left(\lambda \cdot \mathrm{Mo}(S_r) + \frac{(1-\lambda)}{\Omega_{S_r}} \sum_{j=1}^{\Omega_{S_l}} S_r^j\right)\right] \tag{7.1}$$

where $I_s(\lambda)$ is the hybrid intensity value of the *sky* region, and Ω_{S_l} and Ω_{S_r} are the total numbers of pixels in the S_l and S_r sub-regions.

Figure 7.5, on the right, shows the obtained segments of the sky and road. Dark blue and light blue segments are detected based on mean intensity measurements of S_l and S_r, with a variation of ± 10. Similarly, the green segments show the road surface based on R_l and R_r.

In the shown example of a night scene (Fig. 7.5, bottom left), despite an expectation of dark pixels, some bright pixels (due to street lights) fall into the S_l region; this influences our mean-intensity measurement for the left patch of the sky; consequently, a dark blue segmentation (bottom, right) shows regions around the street lights, instead of showing the sky region as part of the light-blue segment.

However, on the other hand, the measurement in S_r supports an "acceptable" segmentation of the sky, shown as a light-blue segment.

The *mode pixel value* (the pixel value with the highest frequency of repetition in $S_l \cup S_r$) determines which of the resulting segments (light blue or dark blue) is a better representative of the sky intensity. By assigning λ equal to 0.66,[2] we consider a *double importance factor* for the detected mode intensity compared to a standard mean; this consequently reduces the negative impact of any inappropriate segmentation. In other words, for the night scene shown at the bottom of Fig. 7.5, the final value of $I_s(\lambda)$ is automatically much closer to the intensity of light blue segments rather than to that of the dark blue (actual sky) segments. A similar approach is applied for road background intensity evaluation, $I_r(\lambda)$, which is shown by dark and light green segments.

As a final stage for defining the adaptive Haar-feature based detector, we experimentally adjusted 10 sets of optimized values for the classifier parameters SWS, SF and MNN based on 10 intensity values of $I_s(\lambda)$ and $I_r(\lambda)$ for the upper and lower part of the input video sequence.[3] This parameter adaptation is then extended for the whole intensity range of 256 values based on Lagrange and cubic interpolations as discussed in Sect. 5.5.3.

[2] Experimentally identified as the optimum value.

[3] Instead of 10, it could be any other number. The more the sets the better the interpolation results. The number 10 proved to be sufficient for obtaining an acceptable interpolation.

7.4 Line and Corner Features

Using the same training dataset, we created three vehicle classifiers using LBP, Standard Haar, and AGHaar. Samples of vehicle detections are shown in Fig. 7.6. The proposed AGHaar classifier provides more accurate *initial* vehicle detection, clearly outperforming LBP and standard Haar classifiers. However, we still consider those initial detections by AGHaar as being *vehicle candidates* or *ROIs* only. In order to obtain even more accurate results (i.e. fewer false-positives and fewer false negative) we continue our evaluation by analyzing line and corner features before confirming that an ROI is a vehicle.

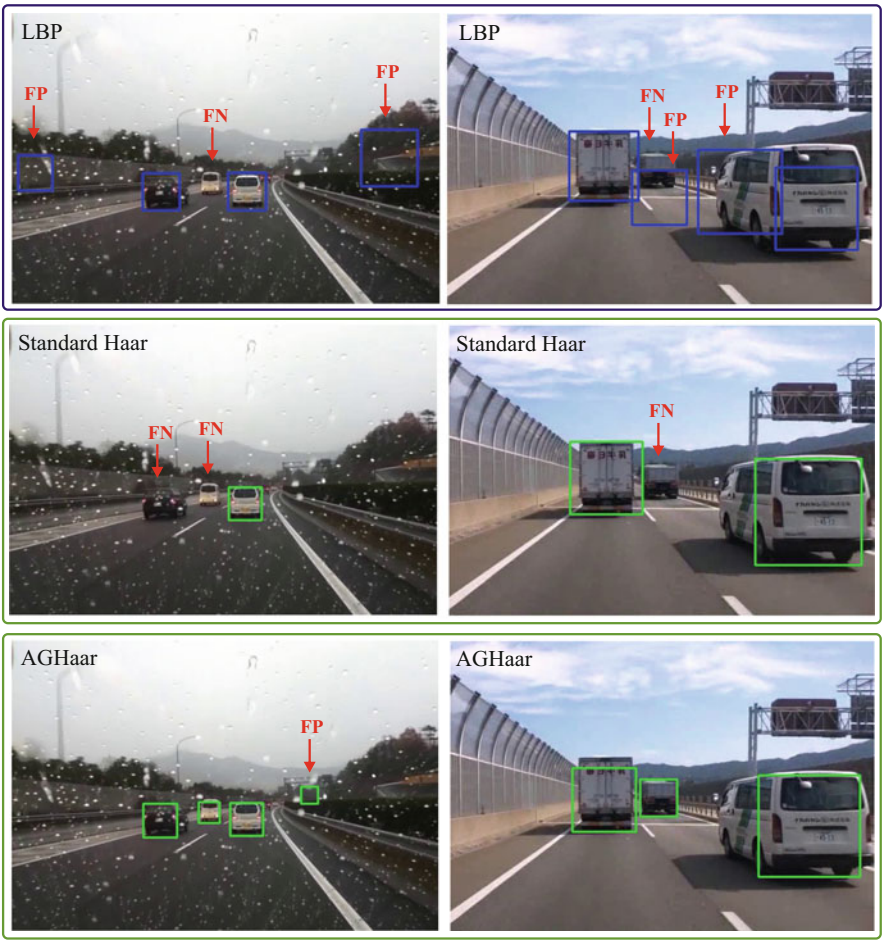

Fig. 7.6 Samples of vehicle detection based on LBP, Standard Haar, and AGHaar classification

7.4.1 Horizontal Edges

More reliable and effective than shadow-based vehicle detection (Fig. 7.1), we take *parallel horizontal edges* into account as a more reliable feature for pointing to a possible existence of a vehicle in an ROI. Our hypothesis is that horizontal edge features can be perceived due to depth differences between bumper and body of a vehicle, edges around a vehicle's registration plate, or horizontal borders of windshields.

We revise the *progressive probabilistic Hough transform* (PPHT) [155] for fast and real-time detection of horizontal edges only. The PPHT was designed following the *standard Hough transform* (SHT) as introduced by Duda and Hart [54]: a line L in the xy coordinate system can be represented by polar coordinates (θ, ρ) as below:

$$\rho = x \cdot \cos\theta + y \cdot \sin\theta. \tag{7.2}$$

Detected edge pixels $P_i = (x_i, y_i)$ in xy-space are transformed into curves in $\theta\rho$-space, also known as *Hough space*, or, in its discrete version, as *accumulator space*.

In the case of the PPHT, a voting scheme is applied to tackle the high computational cost of the SHT. While in the SHT all edge pixels are mapped into the accumulator space, the PPHT only votes based on a fraction of randomly selected pixels. There is one voting bin for each line candidate, and a minimum number of pixels (i.e. of votes) is considered as a threshold for detecting a line. For shorter lines a higher spatial density of supporting pixels is required, while for longer lines a lower spatial density of supporting pixels is sufficient. Overall, the PPHT ensures much faster line detection while the results remain almost equal in accuracy to those obtained by SHT [116].

Figure 7.7 shows a real sample of an accumulator-space calculated from a road scene. The figure illustrates that high accumulator values (red regions) are close to the leftmost or rightmost border at around $-90°$ or $+90°$. This confirms that the number of horizontal-edges in a road scene is considerably higher than the number of edges and lines in other slopes. In order to aim for horizontal lines $y \approx const$ we define two *ranges of interest* for θ:

$$1. \quad 90° - \tau \leq \theta \leq 90°. \tag{7.3}$$

$$2. \quad -90° < \theta \leq -90° + \tau. \tag{7.4}$$

Note that since ρ in PPHT may take both positive and negative values, here the θ is only in the range between $-90°$ and $+90°$.

Mapping back from Hough-space to Cartesian-space, Fig. 7.8-right shows detected horizontal lines for the given road scene. As shown, we can expect to detect one or more horizontal edges for every visible vehicle in the road.

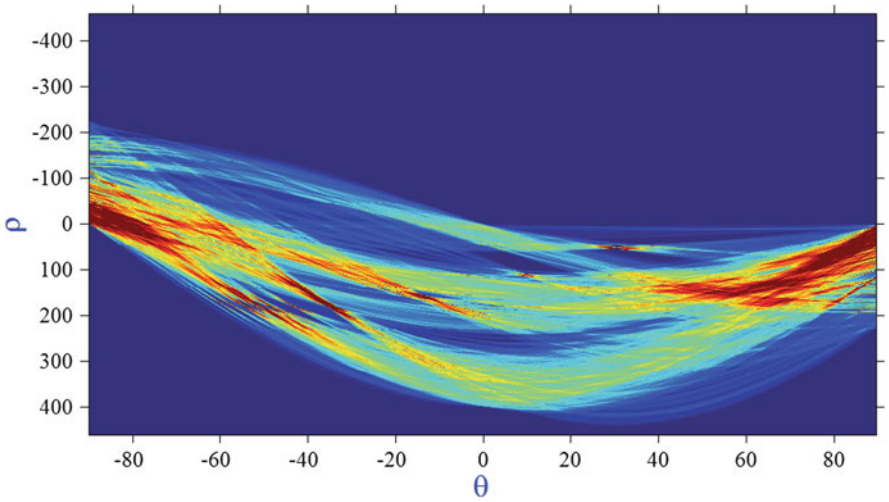

Fig. 7.7 Edge pixels of a sample road scene mapped into the $\theta\rho$-space. The accumulator values are shown using a colour key where *dark blue* is for zero, *red* is for high values, and *light blue* for low positive values

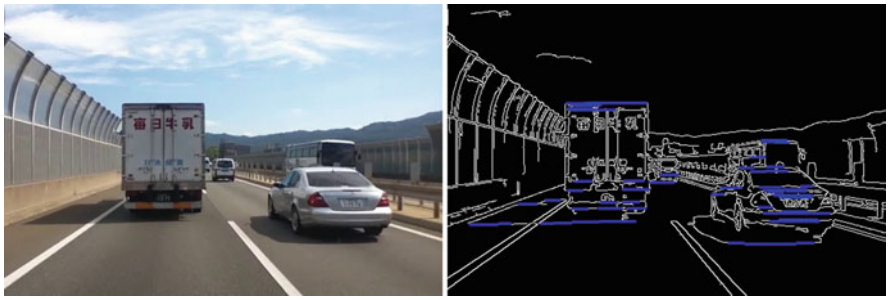

Fig. 7.8 *Horizontal line* detection by our customized PPHT

7.4.2 Feature-Point Detection

Figure 7.8, right, also illustrates that there might be a few more horizontal lines which do not belong to vehicles, for example due to shadows (of vehicles or trees), clouds, or rectangular traffic signs (e.g. large boards). However, shadow regions or non-vehicle objects like traffic signs usually have a plain or simple texture. In order to prevent false detections, we also considered analyzing corner feature-points in the scene.

Our experimental studies indicate that vehicle regions typically include a higher density of corner-points rather than road, sky, or other background regions. The visual complexity of a car's rear-view is defined by combinations of a registration

plate, taillights, a bumper, and the vehicle body. This complexity essentially defines significant corners for a vehicle, especially at the regions below the rear windshield.

Among classical and recently developed feature point detectors such as FAST [221], ORB [223], and FREAK [2], we obtained the best performance with the *Shi–Tomasi* method [235] for detecting more "appropriate" corner points in our road scene application context.

A corner or a feature point is defined by larger intensity differences to adjacent pixels in comparison to non-corner image regions. In this method, an $m \times n$ subwindow W_p is considered which slides through the input image I, defined by the reference pixel $p = (x, y)$ in the upper left corner. The weighted difference between window W_p and an adjacent window of the same size, and at reference point $p + (u, v)$, is measured as follows

$$D_p(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} [I(x_i + u, y_j + v) - I(x_i, y_j)]^2, \tag{7.5}$$

where $0 \le u \le \frac{m}{2}$ and $0 \le v \le \frac{m}{2}$, for $x_i = x + i$ and $y_j = y + j$; weights w_{ij} are used at window positions (i, j); they are either identically 1, or a sampled Gauss function. Using the linear terms of the Taylor expansion of those differences only, it follows that

$$\begin{aligned} D_p(u, v) &\approx \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} [u \cdot I_x(x_i, y_j) + v \cdot I_y(x_i, y_j)]^2 \\ &= \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \left(u^2 I_x^2 + 2uv I_x I_y + v^2 I_y^2 \right), \end{aligned} \tag{7.6}$$

where I_x and I_y stand for the derivatives of I in the x- and y-direction, respectively. By converting into a matrix format, and not including arguments (x_i, y_j), we have that

$$D_p(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \tag{7.7}$$

$$= \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}, \tag{7.8}$$

where $\mathbf{M}$ is short for the matrix defined in Eq. (7.7). Let λ_1 and λ_2 be the eigenvalues of $\mathbf{M}$, representing the differences between the original and moved window, and $R = \min\{\lambda_1, \lambda_2\}$. Corner points are selected by comparing R with a given threshold; if R is greater than this threshold then the centre pixel of W_p is selected as being a corner point [235].

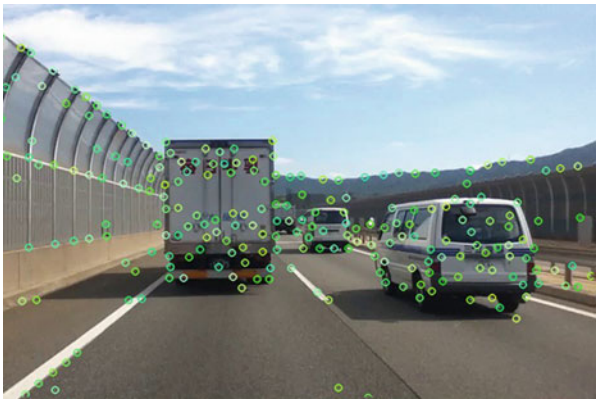

Fig. 7.9 Detected corner points are considerably more dense in a vehicle's rear regions

Figure 7.9 shows the detected feature points. This confirms the expected results of more dense feature-points for a vehicles' rear view, especially in lower parts around the registration plate, bumper, taillights, or tires.

So far we have discussed three possible clues needed to confirm an ROI as a vehicle: Initial AGHaar detection, horizontal edges, and corner features in the lower body part of a vehicle. These clues, all together, can help to prevent false positives.

7.5 Detection Based on Taillights

In contrast to Sect. 7.4, which mainly focused on methods for preventing false-positives, this section proposes a method to retrieve missing detections (false negatives).

As discussed earlier, we believe that any filtering technique not only needs to be robust for detecting vehicles at medium to far distances, it also needs to deal with cases where a vehicle suddenly appears very close, in front of the ego-vehicle (e.g. at a road intersection, or after a temporary occlusion). We identified that taillight features provide very robust support for close to mid-range distance vehicle detections [215].

7.5.1 Taillight Specifications: Discussion

Generally a trained classifier can detect the vehicles which are similar to the images in the training image database. Classifiers try to learn a few "common binary-features" among a training database in order to detect as many vehicles as possible with the highest true-positive rate and a lowest false alarm. That is why

Fig. 7.10 Failed detections or false positive detections for close-up vehicles. *Left*: Haar-based detections. *Right*: LBP-based detections

the classifiers can be more efficient for medium and far distances, as all vehicles at such distances look like a rectangle with a few "limited", "common", and "similar" features such as: windscreen rectangle part at top, rectangular lights on the leftmost and rightmost sides, and a bumper in the lower body part.

If we look at vehicles at relatively far distances, due to missing resolution all vehicles look like a plain rectangle with a few common features. For such cases, our adaptive global Haar-feature based classifier (Sect. 7.3) showed a highly successful performance in general, especially if we fuse it with the described filtering using line and corner features.

However, for vehicles at close distance we have a different situation. Figure 7.10 shows a close-up scene of two vehicles as well as missing detections and false detections by both Haar and LBP classifiers. Due to the high diversity in vehicle makes and models, a close vehicle provides much more details, which can be completely different from one vehicle to another. Such a huge diversity in detail and resolution cannot be efficiently learned or handled by a classifier; there would be numerous false positives and false negatives.

Despite such a diversity in appearances of close-distance vehicles, we hypothesize that there are still some common *geometrical features* that all vehicles have. These geometrical features cannot fit as a few binary features, so are not applicable for training a Haar-feature or LBP based classification; however, we use them for the next step of our approach: *virtual symmetry detection*.

The most visible features of vehicles at close distance are:

1. Taillights colours (all vehicles use an orange-to-red colour spectrum for tail and brake light);
2. Taillight symmetry (taillights are symmetric with the same size and shape);
3. Geometric relations (there are some inherent relationships between the size of a vehicle, the size of its lights, and the Euclidean distance between the light-pairs).

There are few publications about the analysis of taillights for vehicle detection. For example, O'Malley et al. [186] propose a method to detect vehicles based on the symmetry of rear red lights using cross correlation for symmetry detection. However, their method is particularly developed to detect vehicles under night conditions when the rear-lights are on. Also their symmetry detection, using cross correlation, only works if the recording camera (in the ego-vehicle) is exactly behind the target vehicle to ensure a perfect symmetry detection for the taillight pair. Vehicles in other lanes appear at different poses and angles to the recording camera; therefore their rear lights cannot be viewed necessarily as symmetric.

In addition, for many vehicle poses (e.g. Fig. 7.10, bottom), the width of the left light is not visually equal to the width of the right light. In consequence, any method that relies on "actual symmetry detection" will very likely to fail in real-world scenarios. In order to cope with the discussed issues, we introduce our "virtual symmetry detection" approach applicable for both day and night, and for the vehicle in different lanes.

7.5.2 Colour Spectrum Analysis

Pursuing the idea of virtual symmetry detection, we created a database of 482 images from taillights, brake lights, indicator lights, all either in the status of being on or off, under day or night conditions. We converted the collected database from RGB to HSV colour space for a better representation of rear-light pixel colour characteristics [215].

Figure 7.11 illustrates a scatter plot of the colour pixels in our collected taillight dataset. The figure shows that the rear-light pixels are generally scattered in a wide range of colour, from light orange to dark red. This already indicates a need for careful consideration, in order to prevent mis-segmentation.

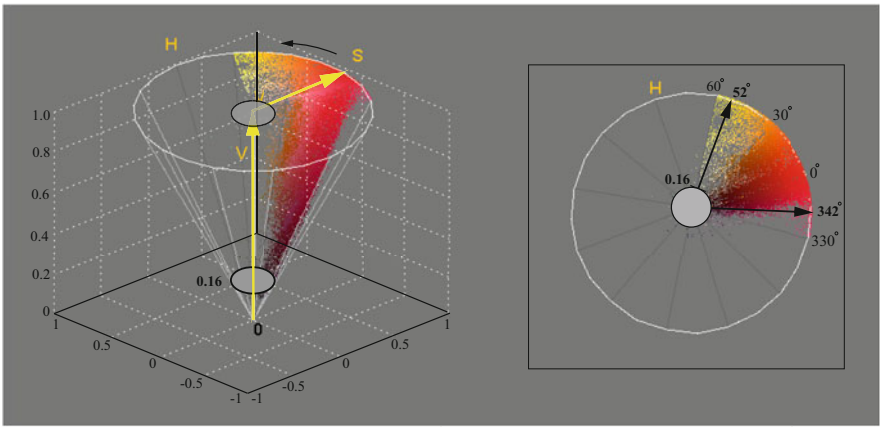

Fig. 7.11 Extracted colour pixels from the vehicle taillight database. *Left*: HSV conical scatter plot. *Right*: Top view on this 3D plot

Due to noise in the database images, some pink, black and yellowish pixels can also be seen in the scatter plot which actually do not belong to taillight pixels. Considering a Gaussian distribution function for the colour pixel distribution in the scatter plot, and excluding the tailed-distribution pixels smaller than -2σ or greater than $+2\sigma$, we remove noisy pixels that have very low density in the scatter plot. Figure 7.11, right, shows the optimized diagram that excludes noisy pixels with

1. a hue value $H \geq 52°$ (i.e. light yellow pixels),
2. a hue value $H \leq 342°$ (i.e. pink pixels), or
3. an intensity value $V \leq 0.16$ (i.e. excluding nearly black pixels).

The rest of the pixel distribution in the scatter plot is considered to be valid for the taillight segmentation procedure.

7.5.3 Taillight Segmentation

Figure 7.12 shows the steps applied for segmentation and pairing of taillights. After conversion from RGB to HSV space (Fig. 7.12a), we apply pixel matching for all three channels based on information obtained from Fig. 7.11, followed by a binary thresholding as shown in Fig. 7.12b.

Figure 7.12c depicts the detected isolated contours. In contrast to lossy techniques using encoded contours [140], we use chain coding [69] to maintain the original boundary of the contours. Detections are simply based on 8-connected components in the threshold image.

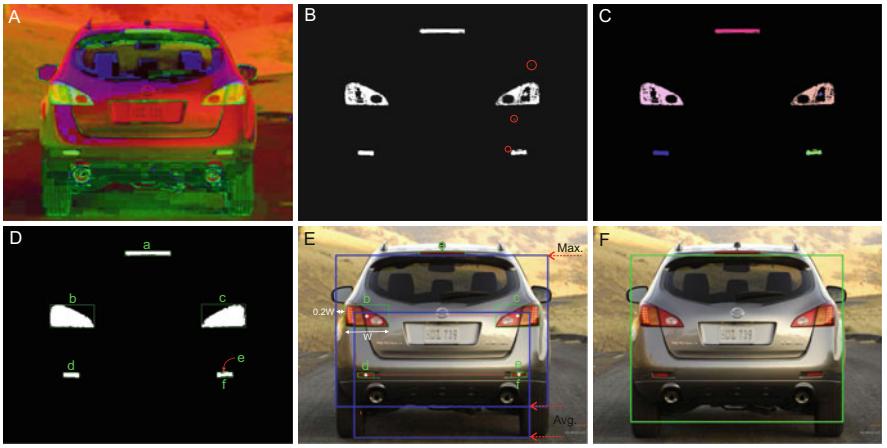

Fig. 7.12 (**a**) HSV conversion. (**b**) Binary thresholded image. (**c**) Individual contour detection and noise removal. (**d**) After hole filling. (**e**) Pairing and vehicle approximation. (**f**) Final detection

Figure 7.12d illustrates the applied procedure for filling the holes in the binary (thresholded) image, thus creating connected blobs. This aims at detecting the actual region of taillights if there are missing pixels due to noise or illumination artefacts.

The shown bounding box illustrates the overall width and height of the detected group of contours. Figure 7.12e, f illustrate the taillight pairing and the approximation of the vehicle region. The details of the taillight pairing procedure will be discussed in the next two subsections.

7.5.4 Taillight Pairing by Template Matching

Before going for taillight pairing based on our *virtual symmetry* (VS) method, we first discuss potential weaknesses of other methods such as symmetry detection based on *template matching*, as used in recent work of Gu and Lee [81]. Then we can have a better understanding of the strength of our VS method.

Let T be a detected contour in an $m \times n$ window, called the *template*. We search in the $M \times N$ image I for a contour which is similar in shape to the horizontally flipped image of T. As usual in template matching, for each location (x, y) of T (e.g. the location of the topmost, leftmost point in T over I) we calculate a *cross-correlation score*, defining a matrix $\mathbf{R}$ of size $(M - m + 1) \times (N - n + 1)$. Location (x, y) in $\mathbf{R}$ contains the cross-correlation score

$$\mathbf{R}(x,y) = \frac{\sum_{i,j}(T'(i,j) \cdot I'(x_i, y_j))}{\sqrt{\sum_{i,j} T'(i,j)^2 \cdot \sum_{i,j} I'(x_i, y_j)^2}}, \tag{7.9}$$

where $1 \leq i \leq m$, $1 \leq j \leq n$, $x_i = x + i$, $y_j = y + j$,

$$T'(i,j) = T(i,j) - \frac{1}{m \cdot n} \sum_{h,k} T(h,k), \tag{7.10}$$

$$I'(x_i, y_j) = I(x_i, y_j) - \frac{1}{m \cdot n} \sum_{h,k} I(x_h, y_k), \tag{7.11}$$

with $1 \leq h \leq m$, $1 \leq k \leq n$, $x_h = x + h$, and $y_k = y + k$. We decided to use this particular cross-correlation method due to its accuracy of matching and time performance in the given context [26].

We slide the template T over the image I by one pixel at a time, from left to right, and top to bottom. For every one-pixel sliding, the matrix $\mathbf{R}$ returns a similarity metric by comparing the *sliding patch* (i.e. the template T over the current sub-image).

Figure 7.13, upper right, illustrates the matrix $\mathbf{R}$ as a correlation map for each position of the query template T over I. Position (x, y) in the upper-left corner of the patch corresponds to a matching value in the correlation map. The brighter a pixel at position (x, y), the higher the level of similarity of I to T at that position. Normalized $\mathbf{R}$ returns values between 0 and 1, and any values greater than 0.95 are considered to indicate a potential match for taillight contour pairing. Figure 7.13, bottom right, illustrates that the result of pairing is still not accurate. This is due to the camera viewing angle, where a pair of taillights cannot always be seen as

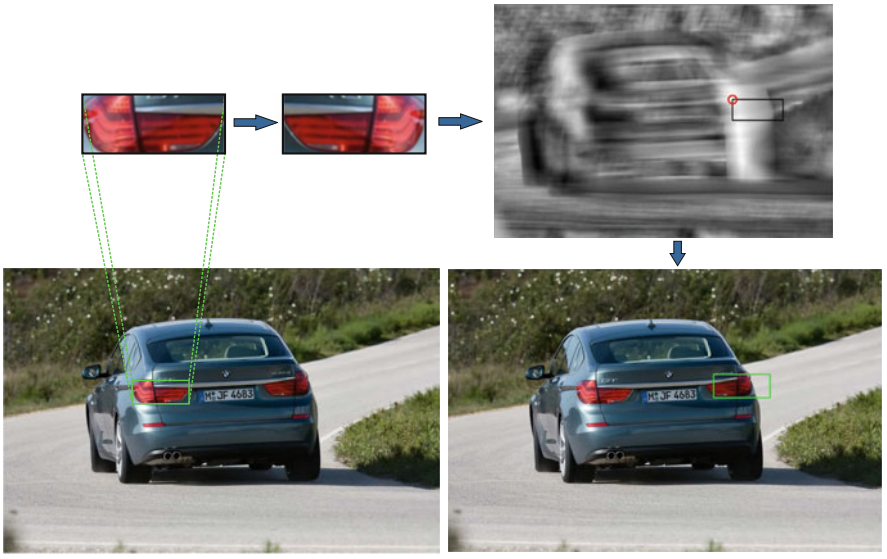

Fig. 7.13 Query template, correlation map, and template matching over the input image

fully symmetric and equal in width and height. We conclude that methods based on template-matching techniques could not be considered as a robust solution.

7.5.5 Taillight Pairing by Virtual Symmetry Detection

In this section we discuss the *virtual symmetry detection* (VSD) method [215]. For the $a = 6$ detected contours in Fig. 7.12d, there are $2^a = 64$ different ways of pairing. However, for the given example only two of those are correct pairs for the taillights: $\{b, c\}$ and $\{f, d\}$.

Furthermore, Fig. 7.13 shows that contours C_i and C_j of a pair of taillights can be asymmetric, of different width, or of different height. We cannot rely on a strict absolute symmetry; instead, we can define some geometrical rules based on a statistical analysis of a rich dataset of taillight images to manifest a *virtual symmetry* (VS) among already detected contours.

Assessing 400 selected vehicle images from the *KITTI* [117] and *EPFL* [57] datasets, we measured the baseline size of taillights, the proportion of the taillight's width to height, their mean sizes, variances and their standard deviations. Based on the studied data, we identified five optimized rules for virtual symmetry detection.

We consider C_i and C_j as being *virtually symmetric* and as the pair $\mathscr{P}_{i,j}$, if the following criteria are met:

$$1.\ \left|\mathscr{A}(C_i) - \mathscr{A}(C_j)\right| \leq 0.3 \left[\frac{\mathscr{A}(C_i) + \mathscr{A}(C_j)}{2}\right], \tag{7.12}$$

$$2.\ -15^\circ \leq \angle\alpha(C_k) \leq 15^\circ, \text{ for } k = i, j, \tag{7.13}$$

$$3.\ 0.9 \cdot [W(C_i) + W(C_j)] \leq |X(C_i) - X(C_j)| \leq 5.3 \cdot [W(C_i) + W(C_j)], \tag{7.14}$$

$$4.\ \max(H(C_i), H(C_j)) \leq 1.35 \cdot \min(H(C_i), H(C_j)); \tag{7.15}$$

$$5.\ \left|\frac{W(C_i)}{H(C_i)} - \frac{W(C_j)}{H(C_j)}\right| \leq 0.2, \tag{7.16}$$

where $\mathscr{A}(C)$ is the total number of pixels in contour C (i.e. the area of the contour C represented in pixels), $W(C)$ and $H(C)$ are the width and height of C in pixels, $X(C)$ is the x-coordinate of the centroid of C, and $\angle\alpha(C)$ is the angle of the two main axes of C.

The first condition is only true for the contours which have more than 70% similarity in terms of their area. Condition 2 allows a maximum of 15° tilt for each of the two contours (e.g. due to road angle; Figs. 7.13 and 7.14). With Condition 3, we make sure that the pair of contours has a baseline distance that falls within the range of measured standard deviation. By applying Condition 4 we check the height difference between the left and right contour, which should be less than 35% (as per

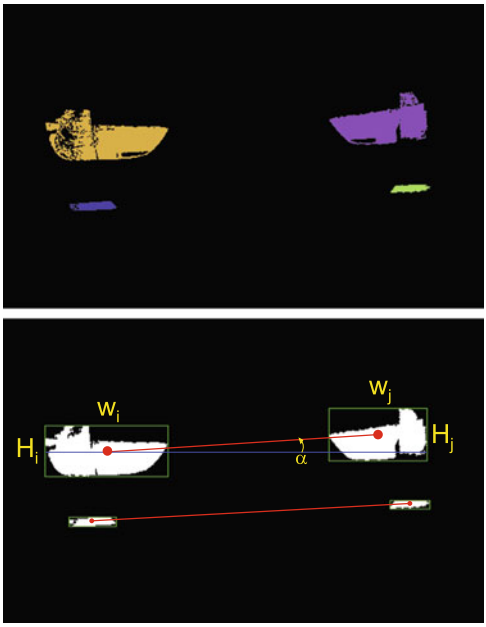

Fig. 7.14 Zoomed taillight contours from the Fig. 7.13, VSD and taillight pairing. An example of tilt and inequality in size of left and right taillights, depending on the road curvature and camera angle

the measured mean in the dataset). Finally, Condition 5 compares the ratios of width to height of left and right contours, which should not be more than 0.2.

Figure 7.15 shows some experimental results based on taillight pairing where Haar and LBP classifiers failed to detect those close-distance cars. We consider a car bounding-box approximation based on the distance between pairs of lights, taillight width and the left-most and the right-most pixels of the detected lights. In case of multiple parallel taillights, for example, main taillight pairs and other parallel light-pairs on the bumper (such as in Fig. 7.12e), we consider an average bounding-box calculated as below:

$$X_l = \min\{x_{l0}, x_{l1}, \ldots, x_{lk}\} - \Gamma \cdot W_l, \tag{7.17}$$

$$X_r = \max\{x_{r0}, x_{r1}, \ldots, x_{rk}\} + \Gamma \cdot W_r, \tag{7.18}$$

$$Y_t = \min\{y_{t0}, y_{t1}, \ldots, y_{tk}\}, \tag{7.19}$$

$$Y_b = \frac{\sum_{bi=0}^{n} y_{bi}}{k}, \tag{7.20}$$

where the values x_{li} belong to the left vertical sides of initially detected rectangles, x_{ri} belong to the right sides, y_{ti} belong to the top-horizontal sides, y_{bi} belong to the bottom-horizontal sides, and $\Gamma = 0.2$ considers a distance of $\pm 0.2W$ as the left and right margin of the car sides from the taillight pairs.

Any bounding box that falls within another bounding box is ignored as false detection, if its size is much smaller than the larger region (e.g. Fig. 7.15d). As

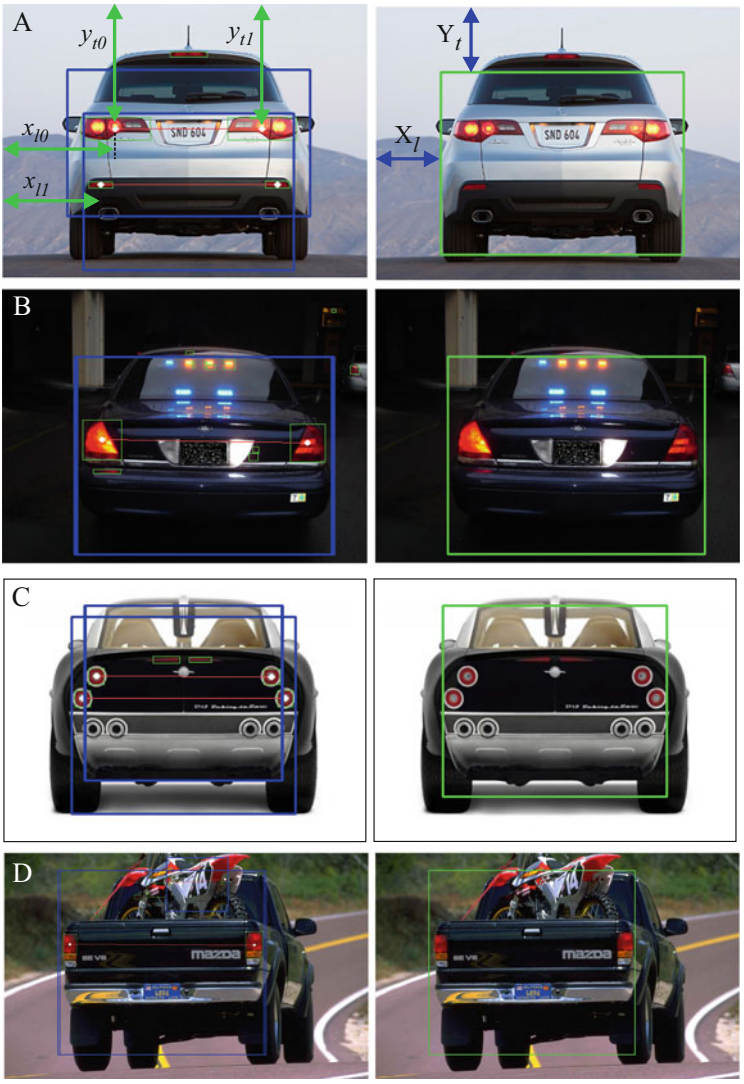

Fig. 7.15 Experimental results for taillight segmentation and pairing

shown, the proposed VSD method performs much more accurately and faster than the template matching method discussed in Sect. 7.5.4.

7.6 Data Fusion and Temporal Information

This section fuses together the already obtained results by AGHaar classification, virtual symmetry detection, and analysis of horizontal lines and corners. Please refer to Fig. 7.16 for an illustration (to be discussed further below). The ultimate goal is achieving an accurate vehicle detection and distance estimation. Since the obtained data are derived from a single sensor, obviously time-synchronized, and in the same pose, we take advantage of this to develop a robust multi-data fusion system. In such a fusion system, there is no need for time-alignment, data registration, sensor validation, or other challenges that are generally involved in common multi-sensor fusion techniques.

The novel AGHaar method alone is robust enough in the majority of road scenarios. In order to ensure an even more reliable detection, we apply data fusion on all the available information, which is what a driver is doing while driving; for example if a vehicle is not fully visible in foggy weather, an expert may consider looking for a registration plate, taillights, or other features of a vehicle in front to estimate its location and size.

Our fusion approach leads to more accurate results while increasing computation cost slightly. This does not hinder the real-time performance of the whole process.

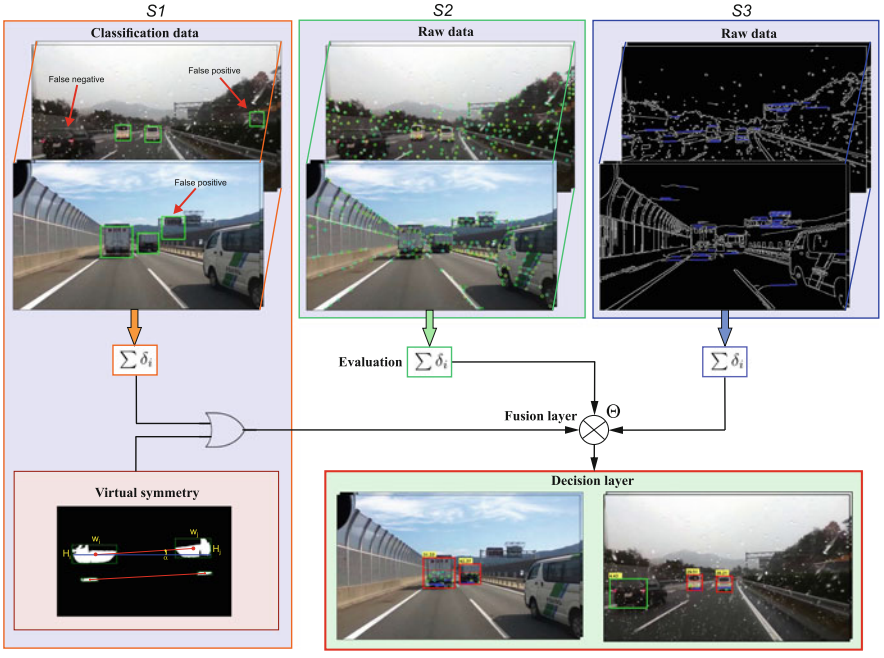

Fig. 7.16 A single-sensor multi-data fusion framework showing examples of successful vehicle detection

We considered two possible approaches for data fusion, namely the Bayesian and the *Dempster–Shafer* [234] theory. The Bayesian method interprets weights of input entities as probabilities. The *Dempster–Shafer theory* (also called the *theory of belief*, or *D-S theory* for short) assigns "masses" based on human expertise which only approximate the concept of probabilities. Since the Bayesian approach is based on "pure" statistical analysis, we also need to be "pure" (i.e. very accurate) on providing the statistical data from each source of information. This, consequently, comes with the requirement of a comprehensive initial database analysis among a wide range of recorded videos from different road scenes. Otherwise, inaccurate weight assignments in a Bayesian approach can cause completely wrong outcomes of data fusion [124].

In contrast to the Bayesian method, the D-S theory is well-known for its effectiveness in expressing uncertain judgements of experts, by serving as an alternative method of modelling evidence and uncertainty. The D-S theory is based on two ideas: (1) to define a degree of belief for a particular question, from "subjective probabilities" of a related question, and (2) Dempster's combination rule, to combine degrees of belief from independent items of evidence.

By using the D-S theory for data fusion for vehicle detection, we not only consider two categories of "vehicle" and "no-vehicle" but we also assign a degree of belief for an "unknown" status. Considering a mass for the "unknown" status we are adding a safety margin to prevent potentially wrong decisions. This automatically takes us to more rational decisions based on a combination of information consensus and human expertise; whereas in the Bayesian technique, we only have two probability values (for "existing" or "not existing"), but not a combination of both.

In the considered context we found that a D-S based fusion approach leads to more acceptable results, especially if we have incompleteness of information or in situations where the accuracy of each information source cannot be assured individually.

Let $\Psi = \{T, NT\}$ be the set representing the state of vehicle detection from each of the four available information sources described in Sects. 7.3, 7.4, and 7.5 (i.e. AGHaar, virtual symmetry, corner features, and horizontal edges) where T denotes that a target (vehicle) is detected, and NT stands for non-target (non-vehicles). Each element in the power set $2^{\Psi} = \{ \{O\}, \{T\}, \{NT\}, \{T, NT\} \}$ is considered to be a proposition concerning the actual state of the vehicle detection system.

Based on the theory of evidence, a mass m_i is assigned for each element in 2^{Ψ}, where $1 \leq i \leq 3$ stands for the three main information sources as follows:

$i = 1$ is for AGHaar, combined with virtual symmetry,
$i = 2$ for corner features, and
$i = 3$ for horizontal lines (edges).

Those three functions m_i are also called *basic belief assignments* for information sources 1, 2, or 3, satisfying

$$m_i : 2^{\Psi} \rightarrow [0, 1] \,, \tag{7.21}$$

Table 7.1 Mass assignments for three sources of information

	Source 1 (m_1)	Source 2 (m_2)	Source 3 (m_3)
Status	AGHaar/Sym. (%)	Corner features (%)	Horizontal lines (%)
T	75	55[a]	65[a]
NT	15	25	20
U	10	20	15
Total	100	100	100

[a]Maximum mass value if features match with threshold τ

with the two properties

$$m_i(O) = 0, \tag{7.22}$$

$$\sum_{A \in 2^{\Psi}} m_i(A) = 1. \tag{7.23}$$

The mass $m_i(A)$ represents the ratio of all relative and available evidence that supports the validity of state A from the ith information source.

For example, considering AGHaar and a VSD combination (AGHaar-VSD) as our main source of vehicle detection (Fig. 7.16, left), we consider $m_1(T) = 0.75$, $m_1(NT) = 0.15$, and $m_1(U) = 0.1$, which means that we have a belief in the true detection rate by AGHaar-VSD in 75% of all cases, we also have a 15% belief for false detections, and we have no opinion in 10% of the cases (unknown assignment) due to lack of knowledge or incompleteness of the analysis. Table 7.1 summarizes the masses identified based on the accuracy of the AGHaar-VSD classification in our ground-truth test dataset.

Depending on size and distance of rectangular regions selected by AGHaar as vehicle candidates, we expect a number of corners and horizontal lines that fall into the lower part of the ROI if the candidate is actually a true positive (a vehicle).

The closer the number of corners (or horizontal edges) to the chosen threshold τ (as defined above) means the higher the possibility of an ROI being confirmed as a vehicle. In other words, if the number of detected corners and horizontal lines is lower than the defined threshold, then the D-S framework decreases the level of belief by appropriately decreasing the default masses of m_2(T) and m_3(T), and, on the other hand, it increases $m_2(NT)$ and $m_3(NT)$ to reject false candidates in the fusion process. However, masses m_2(U) and m_3(U) always remain unchanged.

Also, in order to prevent incorrect updates of m_2 and m_3 due to noise, we apply weighted averaging on the masses by considering the masses allocated for the past n frames (e.g. $n = 30$ in the past second) to utilize temporal information as well:

$$\overline{m}_i = \frac{\sum_{t=1}^{n} \delta_t \, m_i}{\sum_{t=1}^{n} \delta_t}. \tag{7.24}$$

The values for n and δ_t vary depending on the ego-vehicle's speed.

Considering a processing speed of 30 frames per second, the masses in the past few frames should remain close to the actual updated values as per the previous step, or just have a 'smooth' change. Therefore, if a sudden change happens in the current frame due to considerable noise (e.g. intense light) then the weighted averaging contributes to the masses from the temporal information to maintain a moderated mass for the current frame.

Considering the masses m_i as being the confidence in each element of 2^{Ψ}, we measure the combined confidence value $m_{1,2,3}(Z)$ by fusing the information from Sources 1–3 based on Dempster's rule of combination:

$$\begin{aligned} m_{1,2,3}(Z) &= (m_1 \oplus m_2 \oplus m_3)(Z) \\ &= \frac{\sum\limits_{A \cap B \cap C = Z} m_1(A) \cdot m_2(B) \cdot m_3(C)}{1 - \sum\limits_{A \cap B \cap C = \emptyset} m_1(A) \cdot m_2(B) \cdot m_3(C)}, \end{aligned} \tag{7.25}$$

where $\oplus$ denotes the orthogonal sum which is defined by summing the mass product over all elements in the numerator part whose intersections are $A \cap B \cap C = Z$. The denominator applies normalization in the range $[0, 1]$, and it shows the amount of conflict when there is no intersection (no agreement) by those individual sources.

Figure 7.16 shows two examples of fusion results under rainy or sunny conditions based on Dempster's rule of combination.

Detections by AGHaar and VSD are technically independent of each other; however, as discussed earlier, we combine them as information Source 1 in our D-S fusion platform. The combination is represented by the logical symbol for "OR" in Fig. 7.16 and the same mass m_1 in Table 7.1. In case of an AGHaar failure (missing detections), VSD directly acts "together" with corner features and horizontal edge features. In case of detections by both AGHaar and VSD for the same vehicle, we simply apply the mean to define only one ROI per vehicle candidate, before going for data fusion with corner and horizontal edge features.

Overall, the defined multi-clue data fusion approach provides more confident detection as well as a reduced rate of false-negatives that may occur in a singular classification technique such as AGHaar only.

7.7 Inter-vehicle Distance Estimation

In the previous section we illustrated the detected vehicle(s) in a road scene in forms of bounding boxes that indicate the location of the detected vehicle(s) from a rear-view angle. As the next step, in this section, we label every bounding box by an estimate of their distance to the ego-vehicle. Using only monocular vision, it is not possible to directly obtain depth and distance information from a road scene.

However, after remapping the camera image into a 2D transformed domain we can make a distance estimate based on homogeneously distributed pixel distances in the new 2D transformed image.

Assuming an almost planar road surface, knowing the camera optic parameters, camera position, and camera angle, the *inverse perspective mapping* (IPM) can map the recorded images into a *bird's-eye view* [103]. The bird's-eye view approximates an orthogonal top-down view of the scene. As an alternative to IPM, Fig. 7.17 shows our implementation of a mapping of a recorded image into a bird's-eye view using a 4-point calibration as outlined in [318], and the subsequent distance estimation.

Measuring the pixel-distance from the target vehicle to the ego-vehicle in the bird's-eye view, and comparing it with a ground truth metric for the same camera parameters and camera installation, a distance estimation can be performed as illustrated in Fig. 7.17b, c.

Recent work by Tuohy et al. [263] also considers a similar approach for distance estimation; however, an important weakness which we have highlighted

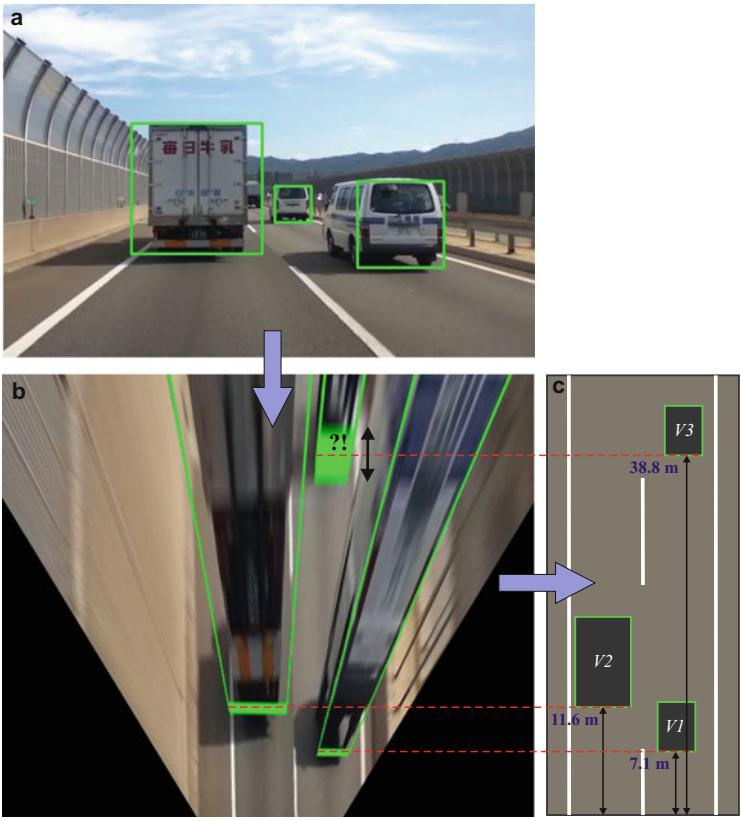

Fig. 7.17 Distance estimation based on bird's eye view

in Fig. 7.17b is neglected. Considering the bottom side of the green bounding box as our distance reference, the bird's-eye view cannot precisely show where the vehicle is located on the road; especially for distances of more than 30 m such as *V3* (Fig. 7.17, the farther vehicle).

The figure shows that every single pixel in the recorded perspective image needs to be mapped into multiple points in a bird's eye view. This transformation involves interpolation. Our evaluations show that the interpolation errors, plus the errors involved in 4-point calibration, cause a distance estimation error up to $\varepsilon = \pm 8\%$. This technique can be suitable for a basic driver-assistance system to prevent crashes at close distances; however, as the distance increases, the estimation error can increase exponentially.

We aim to improve this technique so that we have more accurate distance estimates than just using the bird's-eye view.

As illustrated in Fig. 7.18, we have a forward looking camera (e.g. close to the rear-view mirror), we know the camera field-of-view defined by $\angle\alpha$, the height H of the camera above road level, and the camera viewing angle Θ_c in the $X_cY_cZ_c$-coordinate system.

Assume a detected vehicle in the road scene at an (unknown) position (X_w, Y_w, Z_w). Let θ_v be the angle of a projection ray (for the camera) pointing to the intersection of the planar rear-part approximation of the detected vehicle with the planar road surface; Fig. 7.18, top. The actual distance D between the ego-vehicle

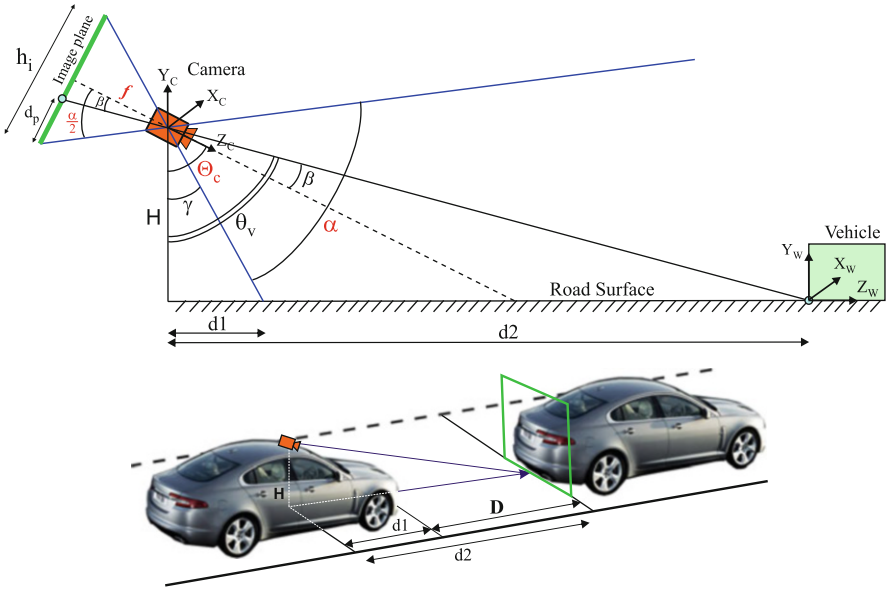

Fig. 7.18 Real-world inter vehicle distance estimation based on pixel distance information in a 2D image plane

and preceding vehicle is equal to $d_2 - d_1$ and can be computed as follows:

$$\begin{aligned} D &= H \cdot \tan(\theta_v) - H \cdot \tan(\gamma) \\ &= H \cdot \left[\tan(\Theta_c + \beta) - \tan(\Theta_c - \frac{\alpha}{2}) \right]. \end{aligned} \tag{7.26}$$

Knowing the Θ_c and α values, only β is needed to calculate D. On the other hand, we have that

$$\tan(\beta) = \frac{\frac{h_i}{2} - d_p}{f}, \tag{7.27}$$

where h_i is the height of the recorded image plane (in pixel units), d_p is the distance from the bottom side of the detected vehicle to the bottom of the image plane (also in pixel units), and f is the camera focal-length. Also we have that

$$f = \frac{h_i}{2 \cdot \tan(\frac{\alpha}{2})}. \tag{7.28}$$

Finally, including β and f in Eq. (7.26), the distance D is computed as:

$$D = H \cdot \left[\tan\left(\Theta_c + \tan^{-1}\left(\frac{\frac{h_i}{2} - d_p}{\frac{h}{2 \cdot \tan(\frac{\alpha}{2})}} \right) \right) - \tan(\Theta_c - \frac{\alpha}{2}) \right]. \tag{7.29}$$

If the ego-vehicle's shock absorbers vibrate on an uneven road then H and β may slightly change and negatively affect the actual D value. This is the only weakness of this approach known to us. Applying a weighted average of the bird's-eye view and the proposed pose-based trigonometric solution ensures a more reliable distance estimation. We provide further details and results in Sect. 7.8.1.

7.8 Experimental Results

In this section we evaluate the performance of the proposed vehicle detection and distance estimation techniques, for various traffic scenarios, weather conditions, and challenging lighting conditions.

Unfortunately, there are only a few basic publicly available datasets for comparative performance evaluation. Most of the available datasets are recorded in daylight only (e.g. KITTI data) or from some elevated positions (such as recording from traffic surveillance cameras), which are not applicable in this research.

Due to incompleteness of the public datasets in the field, we developed the comprehensive iROADS dataset [214] recorded with a 0.7 MP camera (1280×720), a 60° field of view, and 30 fps recording rate, mounted on the back of a rear-view mirror in a car, with a camera tilt angle of $\Theta_c = 82°$, and at a height of about $H = 153$ cm above the road surface. These parameters have been used for comparing ground truth information and distance estimation.

7.8.1 Evaluations of Distance Estimation

We compare distance estimation either based on bird's-eye views, or based on the proposed trigonometric technique.

Japan has one of the highest road standards in terms of consistency of road-signs and road-lane markings. We used Japan's road scenes to evaluate the accuracy of the distance estimation methods discussed in Sect. 7.7. Knowing that the length of any white marking segment in Japanese roads is 8.0 m, and the length of a gap between two white segments is 12.0 m, we extracted ground-truth distance data for about 10 km of the given road scenes.

Using the proposed fusion classifier for vehicle detection, and knowing the camera assembly and relevant pose parameters, the two distance estimation methods discussed in Sect. 7.7 (bird's eye view and trigonometry-based method) have been evaluated.

Figure 7.19 shows the error measures in calculated *distance to vehicles* in front of the ego-vehicle. Errors are defined by comparing with ground truth that is illustrated by the red line. Vehicles are at distances of 6–50 m to the ego-vehicle. We also considered a confidence interval of ± 60 cm for ground truth measurements.

Distance estimation by the camera-pose-based trigonometric method shows more accurate results compared to the bird's-eye view approach. For the camera-pose-based trigonometric method, the error is mainly within the confidence margin of our ground-truth reference. Interestingly, both approaches show a very similar error behaviour for medium distances (in a range of about 22–27 m). The error level increases significantly (up to 9%) for the bird's-eye view technique for far distances.

We identified two common sources of errors for both approaches, and a third source of errors for the bird's-eye view approach: (1) The error of vehicle localization from the vehicle classifier (the error related to the bounding box location); (2) changes in camera height H due to activities of the vehicle shock absorber; (3) an additional error for the bird's-eye view technique due to interpolation and 4-point calibration inaccuracy, as discussed in Sect. 7.7.

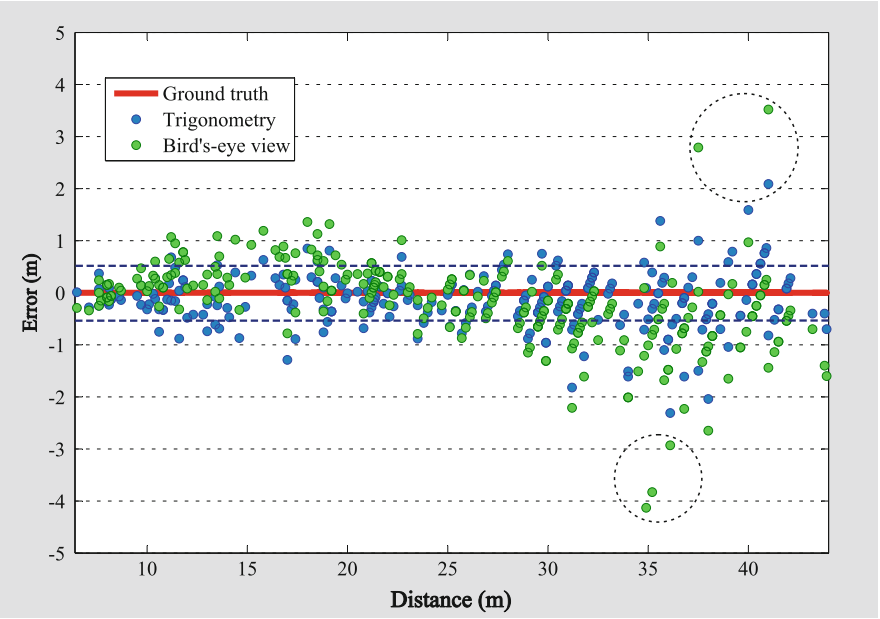

Fig. 7.19 Distance estimation errors for the bird's-eye view technique and the camera-pose-based trigonometric technique

The dashed circles in Fig. 7.19 show errors which are considerably different to others. We reviewed the videos for these moments and we observed that those cases had occurred when the ego-vehicle had suddenly braked, or the shock absorbers were highly active. In both cases, this causes changes in the camera tilt angle Θ_c and in the height H and consequently, larger unexpected errors.

Considering the standard deviations of errors for both techniques, we decided on a weighted average with a coefficient of 0.7 for the camera-pose-based trigonometric method and a coefficient of 0.3 for the bird's-eye view technique.

7.8.2 Evaluations of the Proposed Vehicle Detection

We used a combined road dataset made up of both short-distance cars (Set 1) and the iROADS dataset [214] (Set 2) for performance analysis. Figures 7.20, 7.21, 7.22, 7.23, 7.24, 7.25, 7.26, 7.27, 7.28, 7.29, 7.30, and 7.31 illustrate samples of our experimental results. The research of Klette et al. [120] discusses the variability of traffic or road conditions defined by *situations* (or *scenarios*). A *robust* technique has to perform with reasonable accuracy for different situations. In our experiments

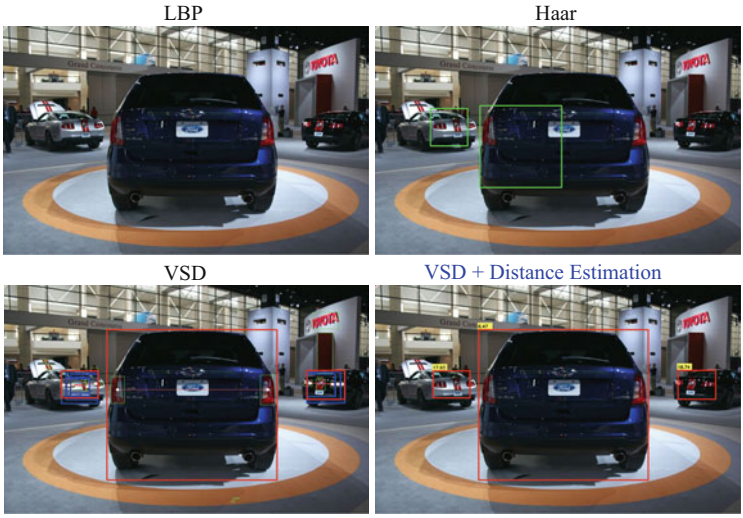

Fig. 7.20 Short-distance vehicle detection and distance estimation with different approaches (Sample 1)

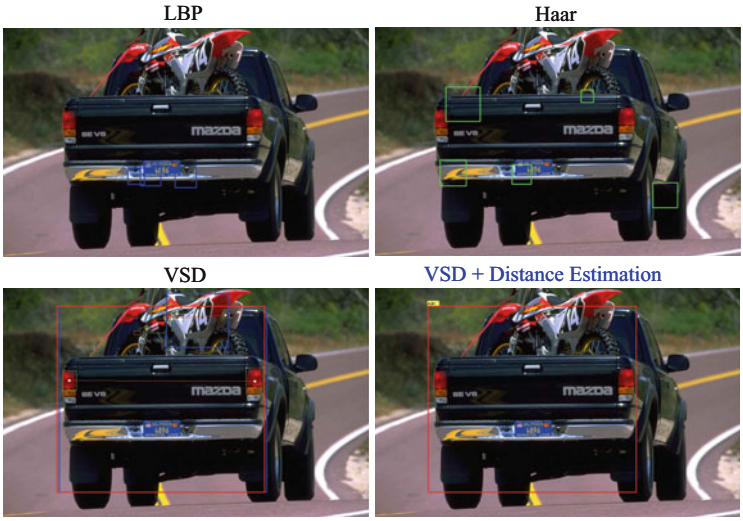

Fig. 7.21 Short-distance vehicle detection and distance estimation with different approaches (Sample 2)

Fig. 7.22 Short-distance vehicle detection and distance estimation with different approaches (Sample 3)

we perform evaluations for six different situations:

1. *close distance*: up to 1 m from the ego-vehicle;
2. *day*: Daylight situation;
3. *night*: Evening and night situation;
4. *rainy day*: Rainy weather under daylight conditions;
5. *rainy night*: Rainy weather under night conditions;
6. *snow*: Partially or fully snow-covered roads.

We apply a full analysis on true detection and false-positive rates by comparing LBP classification, standard Haar-like classification, AGHaar classification, and our proposed Dempster–Shafer data-fusion approach.

The accuracy and robustness of our detection method will be evaluated on image sequences in the six different situations listed above. First we evaluate close-by vehicle detection based on the VSD approach. Then we continue our evaluations for the five other situations for medium to far distances ranging from 10 to 100 m to the ego-vehicle. In a database of 500 car images, varying from older to modern models of vehicles, we gained 91.6% true detection, and 1.2% false alarm. Figures 7.20, 7.21 and 7.22 show three different samples where we compared our VSD method to other techniques. As discussed earlier, a weakness of other approaches is that many false positives and false negatives can be expected. Since the VSD method is only one part of our overall D-S fusion method, we continue with further discussions on the other five situations.

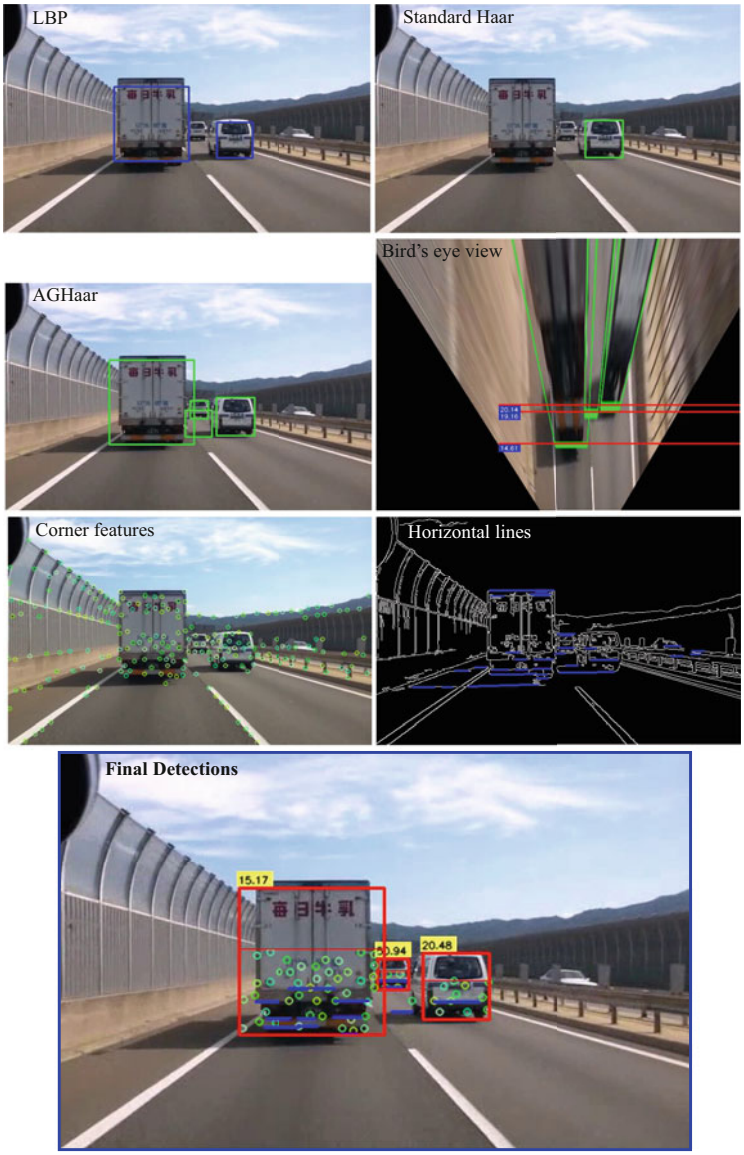

Fig. 7.23 Vehicle detection and distance estimation for situation *day*. From *left* to *right*, *top* to *bottom*: The first three images show the detection results for the discussed LBP, HAAR and AGHaar approaches, the fourth image provides bird's-eye view distance estimation for the AGHaar-based image. The fifth image provides corner features, the sixth image shows the outcome of horizontal edge detection, and the seventh image illustrates the final results of vehicle detection and distance estimation after the proposed data fusion technique. The estimated distances (in *m*) are given in the *yellow rectangles* on the top left side of the *red bounding boxes*

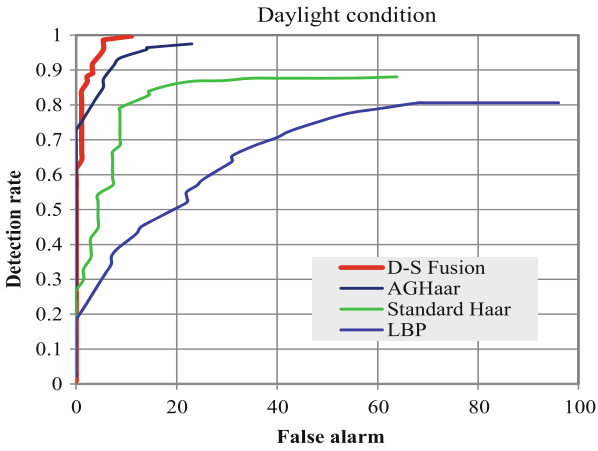

Fig. 7.24 Performance evaluation for situation *day*

Figures 7.23 and 7.24 show detection results and *receiver operating characteristic* (ROC) curves for the situation *day*. LBP-based classification shows the lowest detection rate and the highest rate of false positives. AGHaar alone and the D-S fusion-based method show relatively similar behaviour, better than LBP, while the D-S fusion-based method outperforms the best results with a smaller rate of false alarms. The estimated vehicle distances, shown in the bottom image, are slightly different to those obtained by the bird's-eye view technique, as expected, due to the weighted averaging discussed in Sect. 7.8.1.

Figures 7.25 and 7.26 illustrate experimental result for the situation *night*. The ROC plot in Fig. 7.26 shows that LBP and Standard Haar perform weakly under night conditions. Also, the two horizontal sub-curves in standard Haar and AGHaar curves (dashed ellipses) show that those algorithms had no success for some parts of the test dataset. Those parts of the curves represent cases where only false alarms, or no true detections, occur.

All ROC curves are plotted after sorting the images in the test dataset according to the number of true detections and numbers of false alarms in each image. Images with the highest score come first in the plot, causing a sharp incline in alignment with the vertical axis. In brief, the more "vertical" a curve is, the better the performance.

The LBP detector has a detection rate as low as 52% with a considerable number of false detections. Overall, the night-condition graph shows a lower performance result for LBP, standard Haar, and AGHaar compared to their corresponding ROC plots for day-light condition. This makes sense as it is less likely to capture relevant features in low-light conditions.

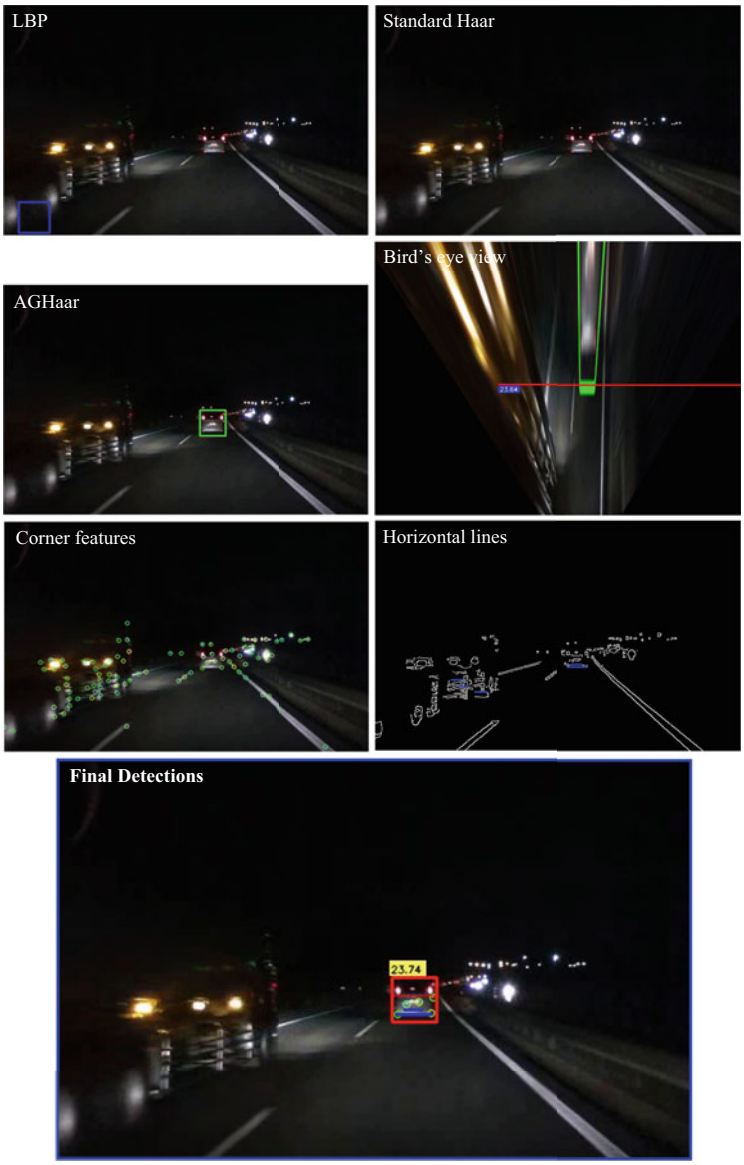

Fig. 7.25 Vehicle detection and distance estimation for situation *night*. Order of images and descriptions as per Fig. 7.23

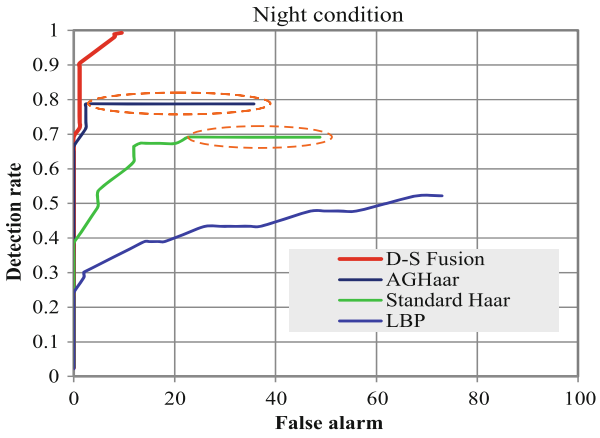

Fig. 7.26 Performance evaluation for situation *night*

However, after applying our VSD and D-S fusion techniques, the fusion-based ROC curve shows a very good detection rate (close to 1.0) with a very small rate of false alarms in both situations, *day* and *night*.

Figures 7.27, 7.28, and 7.29 provide samples of results for rainy day and rainy night conditions. These situations are challenging. For example, Fig. 7.27 shows that there are many false alarms for LBP and standard Haar methods, as well as some missing detections for AGHaar. However, the bottom image shows perfect detections after incorporating VSD and D-S fusion techniques. The green rectangle shows a detection after taillight pairing and VSD.

In contrast to results for the situation *day*, for the situation *rainy night* the AGHaar method did not perform visibly better than standard Haar. This is mainly due to reflections of street lights on rain droplets (Fig. 7.27, top) constituting strong noise which can consequently lead to false alarms. However, again the D-S fusion method shows a high true-detection rate, almost as good as for situations *day* or *night*. We can see only a minor increase in false alarms (raised from 10 to 19), which is a very small portion considering the total number of true detections in our test dataset.

Figure 7.30 shows detection results for situation *snow*, and Fig. 7.31 provides the ROC curves for our sixth dataset containing 1,200 frames from snowy road scenes. Under this condition, the LBP shows a significant increase in its false alarms. However, we can also see an increased rate of true detection for LBP (just below 70%), performing better than for *night and rainy* condition. In this case, the standard Haar shows a poorer performance than for *rain* situations, *night*, or *day*.

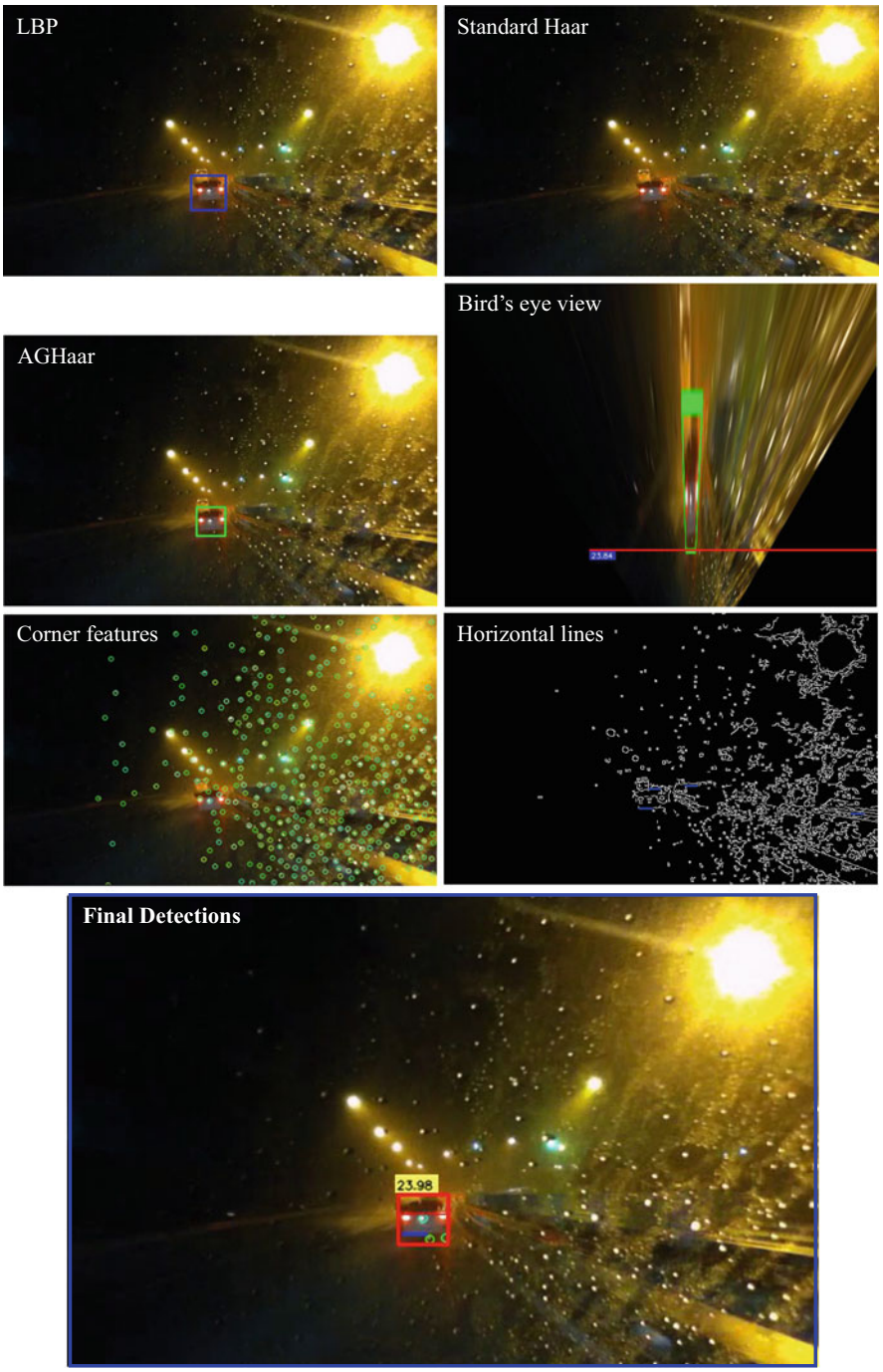

Fig. 7.27 Vehicle detection and distance estimation for the situation *rainy night*. Order of images and descriptions as per Fig. 7.23

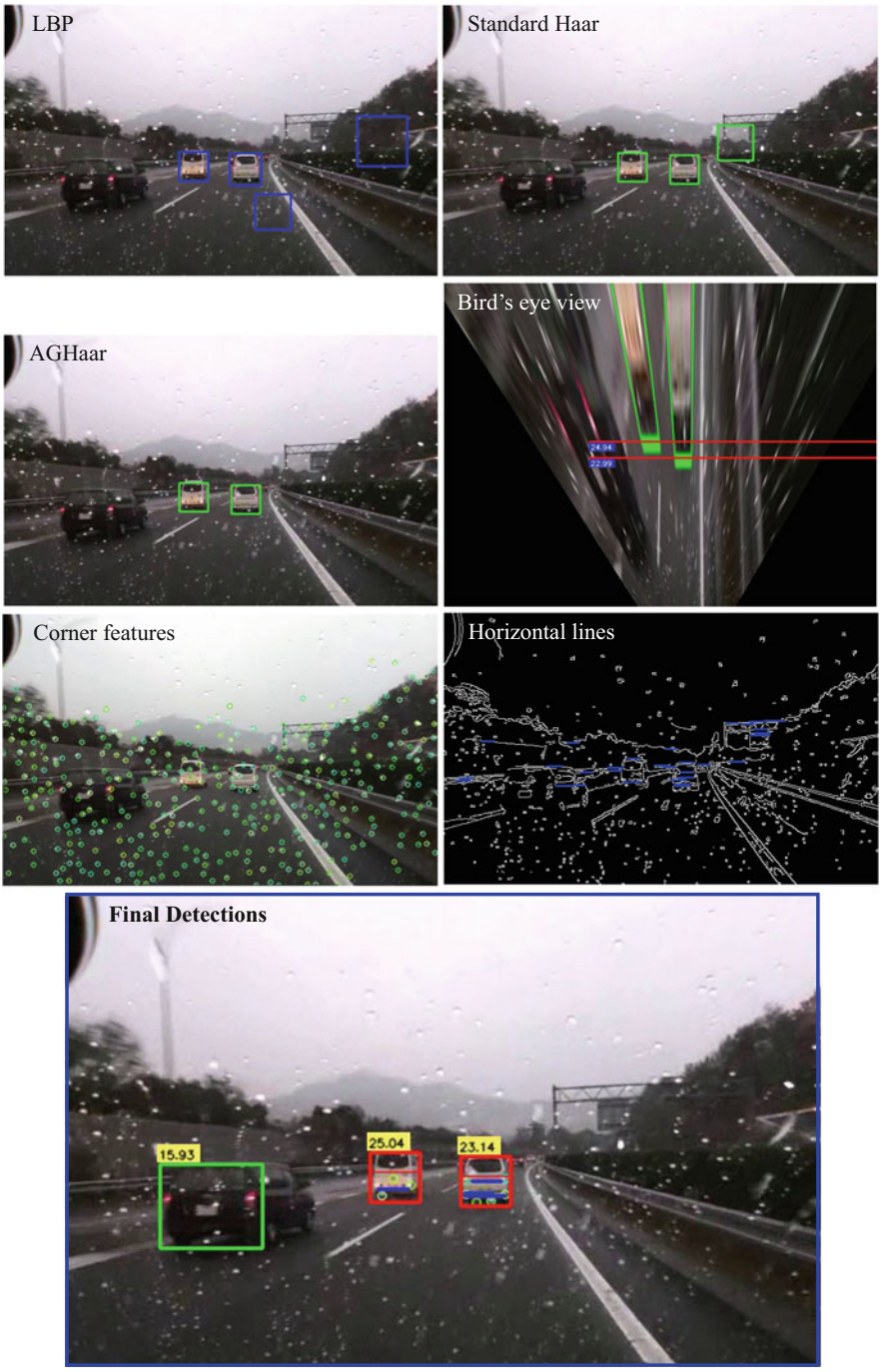

Fig. 7.28 Vehicle detection and distance estimation for the situation *rainy day*. Order of images and descriptions as per Fig. 7.23

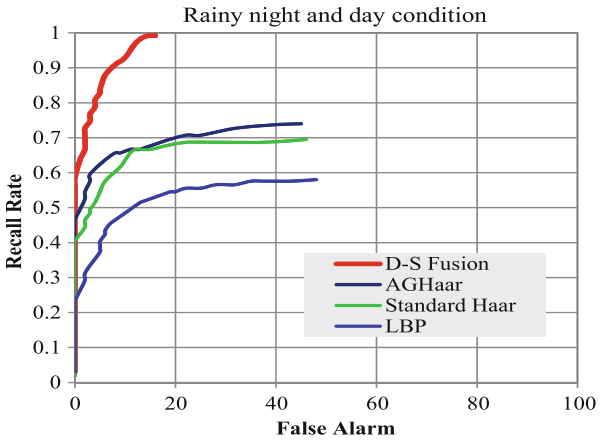

Fig. 7.29 Performance evaluation for the situations *rainy day* and *rainy night*

On the other hand, interestingly, AGHaar performs considerably better, showing the effectiveness of our adaptive global Haar-like classifier for challenging lighting conditions and dynamic environments.

With the D-S fusion approach we had a detection rate of close to 1.0 in the previous four situations. For the situation *snow*, the detection rate stops at 0.88 (almost the same as for AGHaar) but it also shows a reduction in the number of false alarms. One of the reasons for the weaker performance under snowy conditions may be due to a significant variation in illumination of the road scene. An example is the dark grey appearance of the road surface in contrast to the bright white surrounding regions covered by snow. The regions covered with snow may cause strong light reflections and camera blooming, thus making it difficult to achieve highly accurate detections.

Table 7.2 shows the precision rate and recall rate for the proposed method on the four discussed individual datasets plus a comprehensive mixed dataset including all-weather conditions, challenging lighting conditions, and close-distance vehicles. Although the standard classifiers can gain up to around a 90% recall rate for ideal daylight conditions, their detection rate dramatically decreases to under 60% on a real-world comprehensive dataset.

Except for close distance datasets, AGHaar shows a visible improvement compared to other classifiers. The data fusion method is the best performer with a detection rate of 96.8% for the multi-weather and lighting dataset. We also got an overall precision rate of 95.1%, which is acceptable for our comprehensive multi-weather dataset.

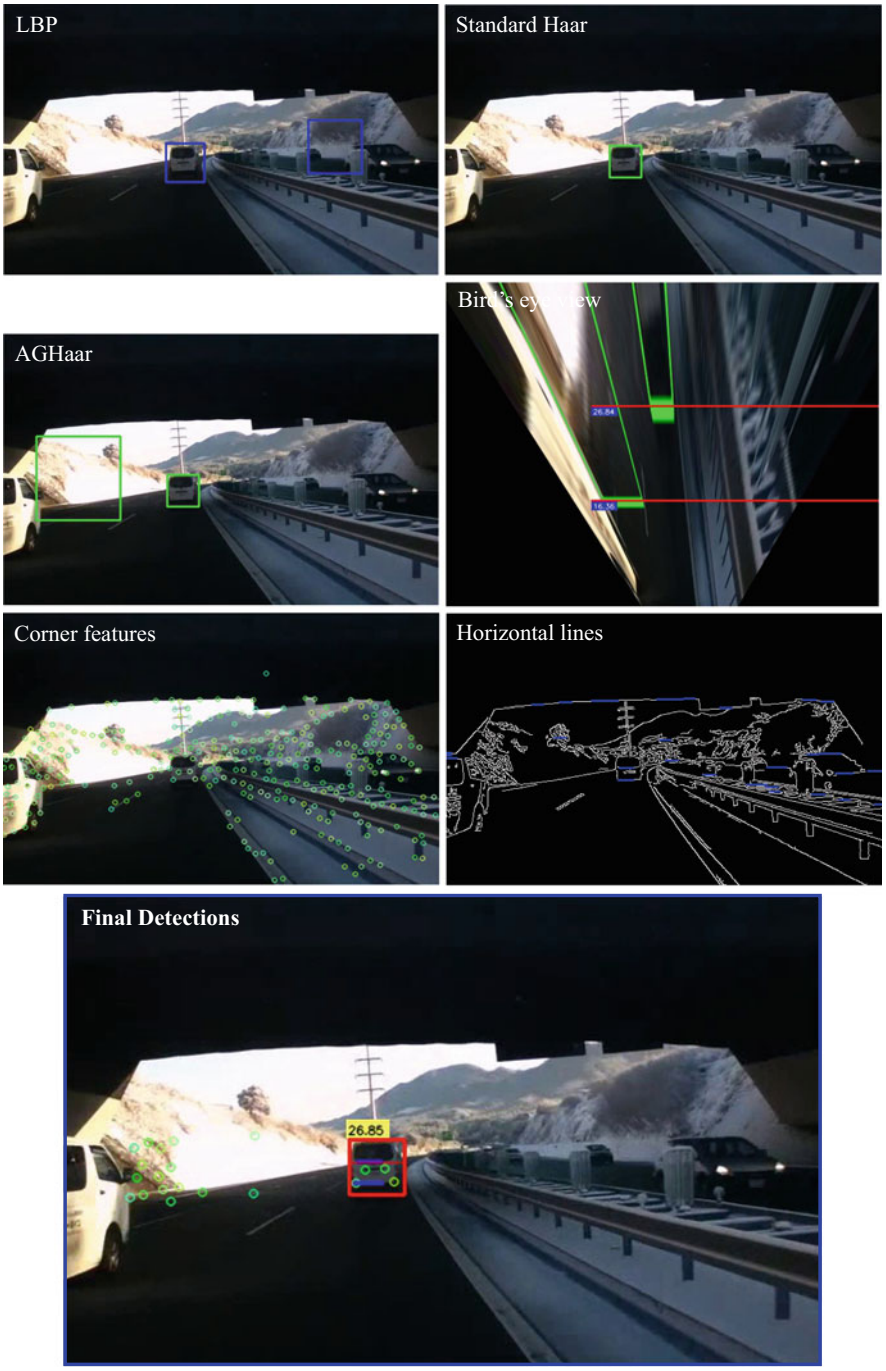

Fig. 7.30 Vehicle detection and distance estimation for the situation *snow*. Order of images and descriptions as per Fig. 7.23

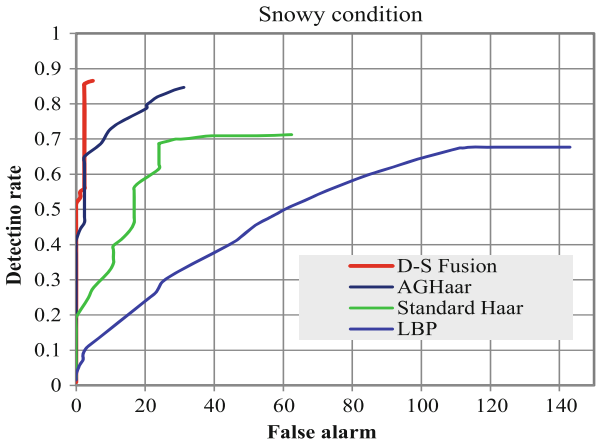

Fig. 7.31 Performance evaluation for the situation *snowy*

Table 7.2 Performance evaluation of the proposed methods on five individual datasets and one mixed comprehensive dataset

	LBP		Haar		AGHaar		VSD & D-S fusion	
	PR	RR	PR	RR	PR	RR	PR	RR (%)
Day dataset	62.8	81.0	73.4	88.2	89.0	97.5	95.2	99.6
Night dataset	63.2	52.6	73.9	69.5	81.5	79.2	95.7	99.2
Rainy dataset	70.6	57.7	75.6	69.8	78.7	73.8	91.6	99.3
Snowy dataset	48.2	67.0	69.4	71.4	84.1	84.8	97.2	87.5
Close dist. dataset	0	0	1.9	3.0	2.1	6.1	96.1[a]	98.8[a]
Mixed dataset	54.4	57.4	65.2	66.8	74.3	75.1	**95.1**	**96.8**

[a]Obtained based on VSD only

7.9 Concluding Remarks

This research proved that even for a rear-view vehicle detection, we need to deal with numerous parameters and challenges to obtain a robust result. If the research aimed at a wide range of vehicle detections, such as multi-direction vehicle detections, the results may not have been regarded as successful.

Recent research, the state-of-the-art work of Felzenszwalb et al. [61, 63], introduces a multi-directional object detection technique based on deformable part models (DPMs). The method is recognized as a generally successful approach to object detection, nevertheless, there are two main issues with this method: inaccurate bonding-box calculation to localize the detected object, and a high computational cost of 2 s per image, in a powerful PC-platform.

None of the mentioned weaknesses are acceptable or can be ignored for an ADAS application. We have already discussed the importance of an accurate bonding box calculation, in order to have a proper distance estimation from a 2D image. Furthermore, the nature of high speed driving, and the risks of crash, injuries, and fatalities do not allow us a processing time of more than few milliseconds for each

input frame. Having a fast, well-defined, feasible and wise approach is the initial requirement of any ADAS.

As another example, we refer to the latest achievements and state-of-the-art work listed on the KITTI benchmark website (in November 2013). Those results show very low detection rates ranging from 18.4 to 74.9% for multi-view vehicle detection, even under (ideal) day-light conditions [117]. The results are still far from satisfying for the needs of real-world applications or industry expectations.

In our developed research we focused on a specific detection scenario, namely rear-view vehicle detection (in order to prevent rear-end collisions). However, we aimed at covering a wide range of weather and lighting conditions. We achieved a detection rate of up to 97% with a high precision rate of 95% not only for day conditions, but also for rainy, snowy, foggy, and many other challenging real-world conditions. This is a significant step forward compared to previously reported results.

The experiments showed the superior performance of the newly proposed AGHaar vehicle detector, compared to common LBP or standard Haar-like classifiers. It also became apparent that the proposed virtual-symmetry detection is important for detecting vehicles at very close distances in addition to medium or far distances that can be covered by the AGHaar approach. The results obtained from the proposed trigonometric distance estimator are acceptable and fast enough to warn a distracted driver, timely, and before an imminent rear-end collision occurs.

The time-effectiveness of the implemented D-S fusion technique allowed us to have real-time processing for 25 fps in a Core i7 PC-platform, for the entire system, from multiple vehicle detection to distance estimation.

Comprehensive experimental studies for our all-weather database illustrate the robustness of the method across various situations. To the best of our knowledge, such a diverse set of challenging situations has neither been used in published benchmarks, nor is available in form of a public dataset. As a result of this research, we also made our accumulated dataset publicly available as Set 10, iROADS [214].

Chapter 8
Fuzzy Fusion for Collision Avoidance

8.1 Introduction

Development of multi-modal advanced driver-assistance systems (ADAS) that can simultaneously monitor the road and driver's behaviour are the objective of both computer vision research and at centres of the automotive industry. The ideal objective is to predict any potentially risky situations that can cause a traffic accident.

Crash risk assessment in a driving scenario is not a trivial task. However, availability of multiple sources of information may help to reduce the complexity of the task and to increase the level of certainty.

In related works by Fletcher et al. [66] and Mori et al. [169], the authors assess the risk based on multi-sensor information that allows them to detect road speed signs, obstacles on the road, and the time to collision *(TTC)*. An alarm is raised if the TTC is smaller than a given threshold, and the driver is looking in the opposite direction of the road.

Some of these warnings are false alarms due to inaccurate judgement, or some could be too late depending on the driver's reaction time, driving situation, or the driver's level of awareness. In addition, dealing with all the above information in a strictly mathematical sense could be complicated or non-verifiable.

Studying the driving behaviour of an expert driver, one can confirm that a driver neither thinks about accurate measurement of distance to obstacles, nor does he/she calculate the TTC. Instead, a driver uses some linguistic expressions such as *very far, far, close* or *very close* to interpret distance to hazards, or may consider *very high, high, low,* or *very low* to express the vehicle's *speed*.

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0_8

Based on such approximations, the driver decides how much to push the throttle pedal, how much to push the brake pedal, or how to adjust the steering angle to maneuver and escape a hazard. In other words, any judgement by an expert driver is based on such approximations concatenated with some IF-THEN rules in the driver's mind. The results are usually sufficiently accurate to prevent a potential crash.

Fuzzy logic reasoning [305] is a very effective approach when we have sufficient knowledge of expertise, but we have a lack of accurate measurements, imperfection of sensors, or involved uncertainties as to the nature of a given task.

We suggest that the driving behaviour can be extrapolated by the concept of *fuzzy logic* and *fuzzy fusion*. Research by Wang et al. [277] assesses the multi-ruled decision-making mechanism of a driver by studying the acceleration and deceleration changes to analyze the driver's behaviour in a car-following scenario. The research also highlights the effectiveness of fuzzy logic for modelling driving activities.

In addition to applications in control, signal processing, and electrical engineering, a great deal of research in computer vision has also used the concept of fuzzy logic for different purposes and in different ways.

Soria-Frisch et al. [242] use fuzzy sets for skin detection. Alshennawy et al. [4] propose a technique for edge detection using fuzzy operators and fuzzy rules. Naranjo et al. [177] develop a method for autonomous vehicle control using fuzzy logic. Utilizing tachometers, GPS information, vision and wireless information, the proposed system controls the steering wheel, throttle, and the brake by firing various types of fuzzy rules.

In some very recent research, as part of a project by the UK highway commission [95], the authors provide a highly accurate fuzzy inference system for crash mapping and hazardous road segmentation.

Milanes et al. [162] introduce a collision warning and avoidance system in order to prevent rear-end crashes in congested traffic conditions. The research uses the data obtained from V2I communications as the inputs of the fuzzy system.

Our review shows that no work has yet considered a crash-risk analysis based on a fuzzy fusion of driver's facial features and road conditions. Using Gaussian, trapezoid, and triangular *membership functions*, *Mamdani* rules of combination, Min/Max *operators*, and centroid *defuzzification*, we model a fuzzy inference system (*FIS*) based on the already available information from *driver-road* status. As discussed in previous sections, we introduced different techniques that enabled us to obtain up to eight types of information from both inside the cockpit (driver's behaviour) and outside (on the road). In the next sections, we describe how we can use these types of information as primary inputs of our fuzzy platform.

The proposed system analyzes the correlation of driver's attention to the potential road hazards and, based on the overall situation, it defines a risk-metric for each moment of a driving scenario, in real-time.

8.2 System Components

A fuzzy logic system (FLS) can be seen as a black box that maps the "crisp" inputs data (or information) into "crisp" outputs, within three main processes of *fuzzification*, *inference*, and *defuzzification* [122]. Figure 8.1 shows the general structure of a fuzzy logic system.

The *fuzzifier* converts a range of crisp input values into a fuzzy value (e.g. a distance range of 2–5 m into "very close"). The *knowledge base* contains the knowledge of expert(s) in the form of linguistic rules. The database includes the required definitions in order to manipulate and to control the input data. The *inference engine* (i.e. data fusion module in our system) performs the decision-making by applying fuzzy implications on the pre-defined linguistic rules. The module simulates the decision-making based on human expertise. The *defuzzifier* converts the fuzzy outputs calculated from previous steps into a crisp output in order to control external devices or parameters; for example to control a brake pedal or to warn a distracted driver in case of high risk conditions.

The fuzzifier maps a crisp real-value x_i into a fuzzy set $\underset{\sim}{\mathrm{A}}$:

$$x_i \in \mathrm{U} \subset \mathbb{R} \quad \text{and} \quad \underset{\sim}{\mathrm{A}} \subset \mathrm{U}, \tag{8.1}$$

where U is our universal set which includes all the real numbers in the range of 0 to 1. A fuzzy set $\underset{\sim}{\mathrm{A}}$ includes a membership function $\mu_{\underset{\sim}{\mathrm{A}}}$ in which for every input x_i, it defines a degree of membership in the range [0, 1]:

$$\mu_{\underset{\sim}{\mathrm{A}}}(x_i) : \mathrm{U} \rightarrow [0, 1]. \tag{8.2}$$

Thus, we can express a fuzzy set $\underset{\sim}{\mathrm{A}}$ in **U** as a set of ordered pairs in which:

$$\underset{\sim}{\mathrm{A}} = \{(x_i, \mu_{\underset{\sim}{\mathrm{A}}}(x_i)) \mid x_i \in \mathrm{U}\ ,\ \ \mu_{\underset{\sim}{\mathrm{A}}}(x_i) : \mathrm{U} \rightarrow [0, 1]\}. \tag{8.3}$$

In this chapter we only use *normal membership functions* or normal fuzzy sets. A normal fuzzy set has a maximum height of 1 (i.e. $\text{Height}(\mu_{\underset{\sim}{\mathrm{A}}}(x)) = 1$). The next section details the structure of our fuzzy-fusion method.

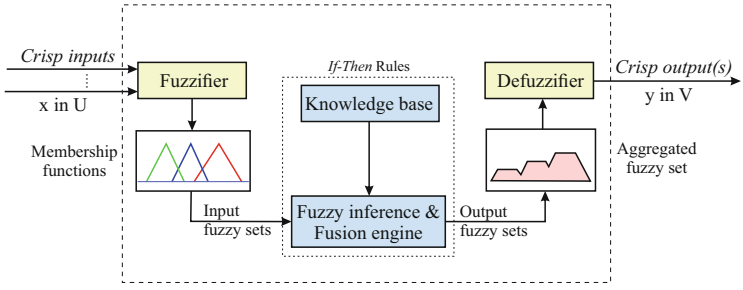

Fig. 8.1 A typical configuration of a fuzzy logic system

8.3 Fuzzifier and Membership Functions

In order to construct the fuzzifier section of our FIS, we define 23 linguistic expressions and their membership functions based on the eight possible inputs from the driver and road conditions. We have three inputs from driver's head-pose status, including yaw, pitch and roll; three inputs from driver's vigilance, including eye status, yawning status, and head nodding; and two inputs from the road scene, including the distance of the lead vehicles to the ego-vehicle, and their angular positions.

Depending on the nature of the input parameters, we considered one to five linguistic variables for every input. Table 8.1 illustrates the main inputs, crisp ranges, and the defined linguistic variables. All the "linguistic terms" and "ranges" of inputs are defined based on (a) average suggestion of expert drivers, or (b) the capability of our proposed algorithms (face/pose analysis and vehicle detection) for a robust operation in the defined range.

Table 8.1 Inputs, crisp ranges, and 23 linguistic variables to construct the fuzzifier module of the FIS

Main categories	Inputs	Crisp ranges	Linguistic variables
Driver's head-pose	Yaw	$[-65°, 0°]$	{Left}
		$[-65°, +65°]$	{Centre}
		$[0°, +65°]$	{Right}
	Pitch	$[-45°, 0°]$	{Up}
		$[-45°, +45°]$	{Centre}
		$[0°, +45°]$	{Down}
	Roll	$[-45°, 0°]$	{Left}
		$[-45°, +45°]$	{Middle}
		$[+45°, 0°]$	{Right}
Driver drowsiness	Eye status	{1 or *true*}	{Closed}
		{0 or *false*}	{Open}
	Yawning	{1 or *true*}	{Yawning}
		{0 or *false*}	{Normal}
	Head nodding	{1 or *true*}	{Nodding}
		{0 or *false*}	{Normal}
Vehicle detection	Distance (m)	[0, 10]	{Very close}
		[7, 23]	{Close}
		[20, 40]	{Medium}
		[37, 63]	{Far}
		[50, 100]	{Very far}
	Angle	$[-60°, 0°]$	{Left lane}
		$[-60°, +60°]$	{Same lane[a]}
		$[0°, +60°]$	{Right lane}

[a]*The detected vehicle is in the same lane where the ego-vehicle is travelling*

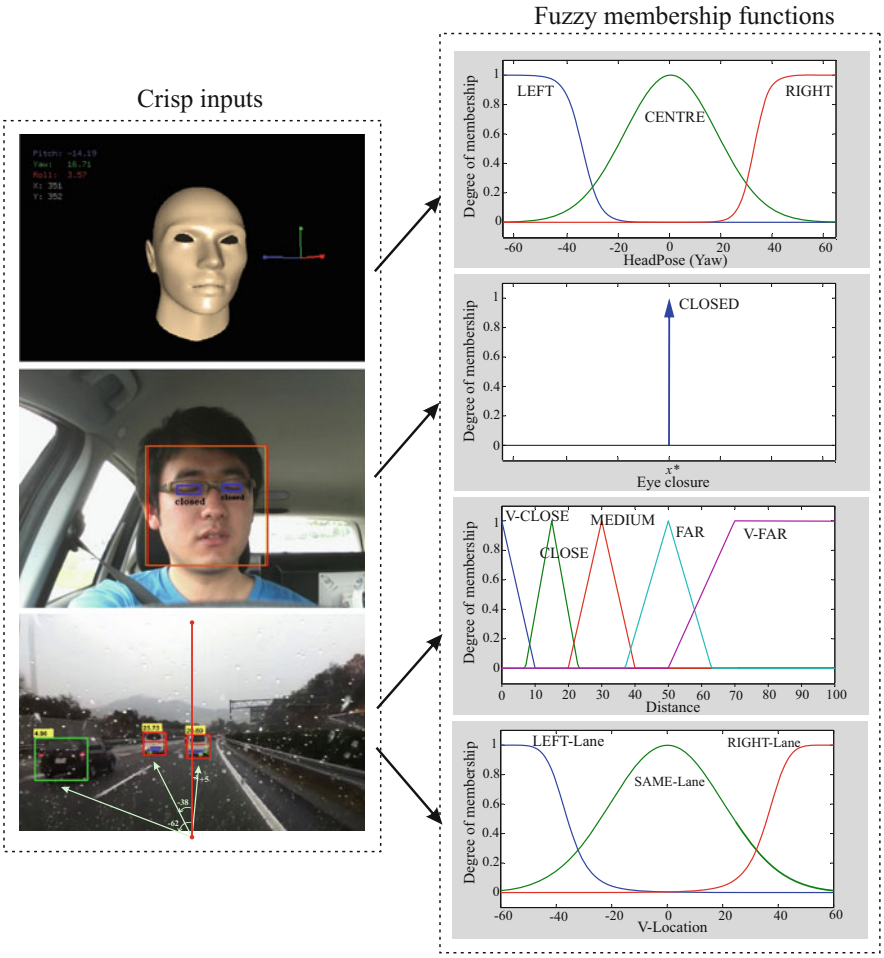

Fig. 8.2 Main inputs and the defined fuzzy membership functions considered for construction of the fuzzifier module of the proposed FIS

Figure 8.2 shows four sample inputs of driver's head (*yaw*), eye status (*eye-closure*), lead vehicle's *distance*, and lead vehicle's *angle*.

We used eight types of membership functions depending on the nature of the inputs and their characteristics [121]. For the four inputs from driver's head-pose (yaw, pitch, roll) and the angle of the leading vehicles to the ego-vehicle we used the *Gaussian* membership function:

$$\mu_{\underset{\sim}{A}_1}(x_i) = e^{-\frac{(x_i - m_1)^2}{2\sigma_1^2}}, \tag{8.4}$$

the *Z-sigmoidal* membership function:

$$\mu_{\underset{\sim}{A}2}(x_i) = \frac{1}{1 + e^{(x-m_2)\sigma_2}}\ , \tag{8.5}$$

and the *S-sigmoidal* membership function:

$$\mu_{\underset{\sim}{A}_3}(x_i) = \frac{1}{1 + e^{(-x+m_3)\sigma_3}}, \tag{8.6}$$

where m_i is the centre of the membership function and σ_i is the variance which defines the shape of the curve. For these inputs we considered a *model-driven* approach [293], therefore we used Gaussian-based functions to suppress the measurement noise involved in the input values from head-pose estimation.

We also considered three types of membership functions to map the input values of the "distance" into a fuzzy form. As shown in Fig. 8.2, for the linguistic variables of *close, medium*, and *far*, we used triangular membership functions:

$$\mu_{\underset{\sim}{A}_j}(x_i) = \begin{cases} \gamma(\frac{x_i - a_j}{m_j - a_j}) & \text{if } a_j \leq x_i \leq m_j, \\ \gamma(\frac{x_i - b_j}{m_j - b_j}) & \text{if } m_j \leq x_i \leq b_j, \\ 0 & \text{otherwise,} \end{cases} \tag{8.7}$$

and to define *very close*, and *very far*, we used left- and right-trapezoidal membership functions:

$$\mu_{\underset{\sim}{A}_j}(x_i) = \begin{cases} \gamma(\frac{x_i - a_j}{m_j - a_j}) & \text{if } a_j \leq x_i \leq m_j, \\ \gamma & \text{if } m_j \leq x_i \leq b_j, \\ 0 & \text{otherwise,} \end{cases} \tag{8.8}$$

where a_j and b_j are left and right corners of the triangle (trapezoid) on the x axis, m_j is the centre of the triangles (trapezoids), and $\gamma = 1$ in our system. We used triangular-based functions based on a *knowledge-driven* approach [293] and to suppress the inaccuracy of the input values (this could occur due to the errors involved in monocular distance estimation for non-planar roads). To define the support range of triangular and trapezoidal functions we considered the average recommendation of 10 expert drivers.

For eye-closure status, yawning detection, and head nodding inputs we simply considered singleton membership functions:

$$\mu_{\underset{\sim}{A}_j}(x_i) = \begin{cases} 1 & \text{if } x_i = x^\star, \\ 0 & \text{otherwise.} \end{cases} \tag{8.9}$$

8.4 Fuzzy Inference and Fusion Engine

This section provides the details of our fuzzy inference system, which is also called the *decision-layer* or the *fuzzy-fusion* module in this book. There are many well-known Fuzzy logic systems such as Mamdani [149], Takagi–Sugeno–Kang (TSK) [249], or Larsen [152] that interpret and combine the fuzzy rules in different manners. For our system we used the Mamdani model which receives crisp inputs and has proven advantages of simplicity, efficiency and low computational cost [229].

Fuzzy rule base is a set of IF-THEN rules which acts as the heart of the system. Let us define a fuzzy rule as follows:

$$\begin{aligned} \text{Rule}_j: \quad & \text{IF} \quad (x_1 \text{ is } \underset{\sim}{\text{A}}_1^j), (x_2 \text{ is } \underset{\sim}{\text{A}}_2^j), \ldots, \text{ AND } (x_i \text{ is } \underset{\sim}{\text{A}}_i^j) \\ & \text{THEN} \quad (y_j \text{ is } \underset{\sim}{\text{B}}_j) \end{aligned} \tag{8.10}$$

or in short:

$$\underset{\sim}{\text{R}}_j : \underset{\sim}{\text{A}}_i^j \rightarrow \underset{\sim}{\text{B}}_j \tag{8.11}$$

where

$$\underset{\sim}{\text{A}}_i^j \subset \text{U} , \tag{8.12}$$

$$\underset{\sim}{\text{B}}_j \subset \text{V} , \tag{8.13}$$

$$\mathbf{x} = [x_1, \ldots, x_i]^\top \in \text{U} \subset \mathbb{R} , \tag{8.14}$$

$$\mathbf{y} = [y_1, \ldots, y_j]^\top \in \text{V} \subset \mathbb{R} , \tag{8.15}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. $\underset{\sim}{\text{A}}_i^j$ denotes the fuzzy set j and the ith input, and $\underset{\sim}{\text{B}}_j$ is the output of the rule j. Some sample rules for our system could be as follows:

IF (Yaw-angle is *Left*) $\wedge$ (Vehicle-angle is *Right-lane*)
THEN (Risk-level is *Very high*)
IF (Eye-status is *Open*) $\wedge$ (Vehicle distance is *Very-far*)
THEN (Risk-level is *Very low*)
IF (Yawning is *True*) $\wedge$ (Vehicle angle is *Same-lane*)
THEN (Risk-level is *Medium*)
...

8.4.1 Rule of Implication

There are different methods and points of view for the implication of fuzzy rules such as the *Willmott*, *Mamdani* or *Brouwer–Gödel* implication methods [94]. If each rule in the system is performing as an independent conditional statement, then the Mamdani method that uses the *union operator* would be appropriate; if the rules are strongly connected to each other, then the Brouwer–Gödel method with the *intersection operator* is suggested.

Although all the inputs in our developed system contribute to determine the risk-level of a driving scenario, most of the defined rules in the system are independent of each other. For example, while an "eye-closure" or a "head-pose distraction" can both lead to the identification of a *high-risk* driving condition, they are acting as independent and individual inputs. Therefore we use the Mamdani-based implication.

Given rules "$\underset{\sim}{\mathrm{R}}_j$: IF $(x_1, \ldots, x_i)$ is $\underset{\sim}{\mathrm{A}}_i^j$ THEN $(y_j$ is $\underset{\sim}{\mathrm{B}}_j)$" with the Mamdani model we combine the rules as:

$$R_{\mathbf{x}}(y) = \bigcup_{j=1}^{m} \underset{\sim}{\mathrm{A}}_i^j(\mathbf{x}) \wedge \underset{\sim}{\mathrm{B}}_j(y) \tag{8.16}$$

or

$$R_{\mathbf{x}}(y) = R(x_1, \ldots, x_i, y) = \bigcup_{j=1}^{m} \left(\underset{\sim}{\mathrm{A}}_1^j(x_1) \wedge \cdots \wedge \underset{\sim}{\mathrm{A}}_n^j(x_n) \wedge \underset{\sim}{\mathrm{B}}_j(y) \right), \tag{8.17}$$

where $\bigcup$ is the *union operator* and $\wedge$ is the logical *and* that convert the antecedent part of the rule (a single number as membership degree) into an output fuzzy set using *minimum* (t-norm) operators. The output of the implication phase is actually a set of truncated functions each of which is the output for one rule.

8.4.2 Rule of Aggregation

With the aggregation process we apply an *accumulative* operator on all the truncated functions (obtained from the implication phase) in order to define the output fuzzy sets. There are two typical aggregation methods: *sum* and *maximum* [220]. Since we use normal membership functions (i.e. the maximum weight of the membership functions is 1), the sum method is not applicable for our system and we use the maximum aggregation method. The method considers the outer margin of all individually truncated membership functions to form an output fuzzy set. Figure 8.1, on the right side, illustrates a sample output fuzzy set after the aggregation of three sample *min*-truncated functions.

8.5 Defuzzification

Defuzzification is a process that produces a single "crisp output number" that best describes the truncated output fuzzy sets [121]. Considering the given output set from the previous step, we used the *centroid* or *centre of gravity* (COG) as one of the most popular methods of defuzzification:

$$z^*_{COG} = \frac{\int_b^a z \cdot \mu_{\underset{\sim}{C}}(z)dz}{\int_b^a \mu_{\underset{\sim}{C}}(z)dz}, \tag{8.18}$$

where z^* denotes the crisp centroid defuzzified output, $\int$ is the algebraic integration, $\underset{\sim}{C}$ is the output fuzzy set after the aggregation step, $\mu_{\underset{\sim}{C}}$ is the membership function of the output fuzzy set, and a and b are the left and right support of the fuzzy set $\underset{\sim}{C}$.

We also tried four other defuzzification methods, including the *bisector* method, *smallest of maxima* (SOM), *largest of maxima* (LOM), and *middle of maxima* (MOM):

$$z^*_{MOM} = \frac{1}{M_r - M_l} \sum_{j=M_l}^{M_r} z_j \quad \text{and} \quad \mu_{\underset{\sim}{C}}(z_j) = \max\{\underset{\sim}{C}\}, \tag{8.19}$$

where M_l and M_r (SOM and LOM) are the left and right support boundary of z_j and both have the maximum degree of membership function among all z and $M_l \leq M_r$.

In the bisector method, z^*_{BS} defines the output defuzzified value:

$$\int_a^{z^*_{BS}} \mu_{\underset{\sim}{C}} dz = \int_{z^*_{BS}}^b \mu_{\underset{\sim}{C}} dz \tag{8.20}$$

in which a vertical line at point z^*_{BS} divides the fuzzy set $\underset{\sim}{C}$ into two sub-regions of equal size.

In the next section we discuss the reason behind our preference for the centroid method by comparing the output results with the other four discussed defuzzification techniques.

8.6 Experimental Results

Implementation of a complete knowledge-base for the proposed system requires the definition of at least 3,240 rules ($\prod_{k=1}^{8} N_k$), where N_k is the number of linguistic variables for input k, as per Table 8.1.

In order to simplify the discussion we perform the following experiment for 4 dimensions (3 inputs, 1 output) out of our 9D system. Therefore, we define 45 fuzzy-rules for the three main inputs of the system including yaw-angle, vehicle-distance, and vehicle-angle, as shown in Fig. 8.3.

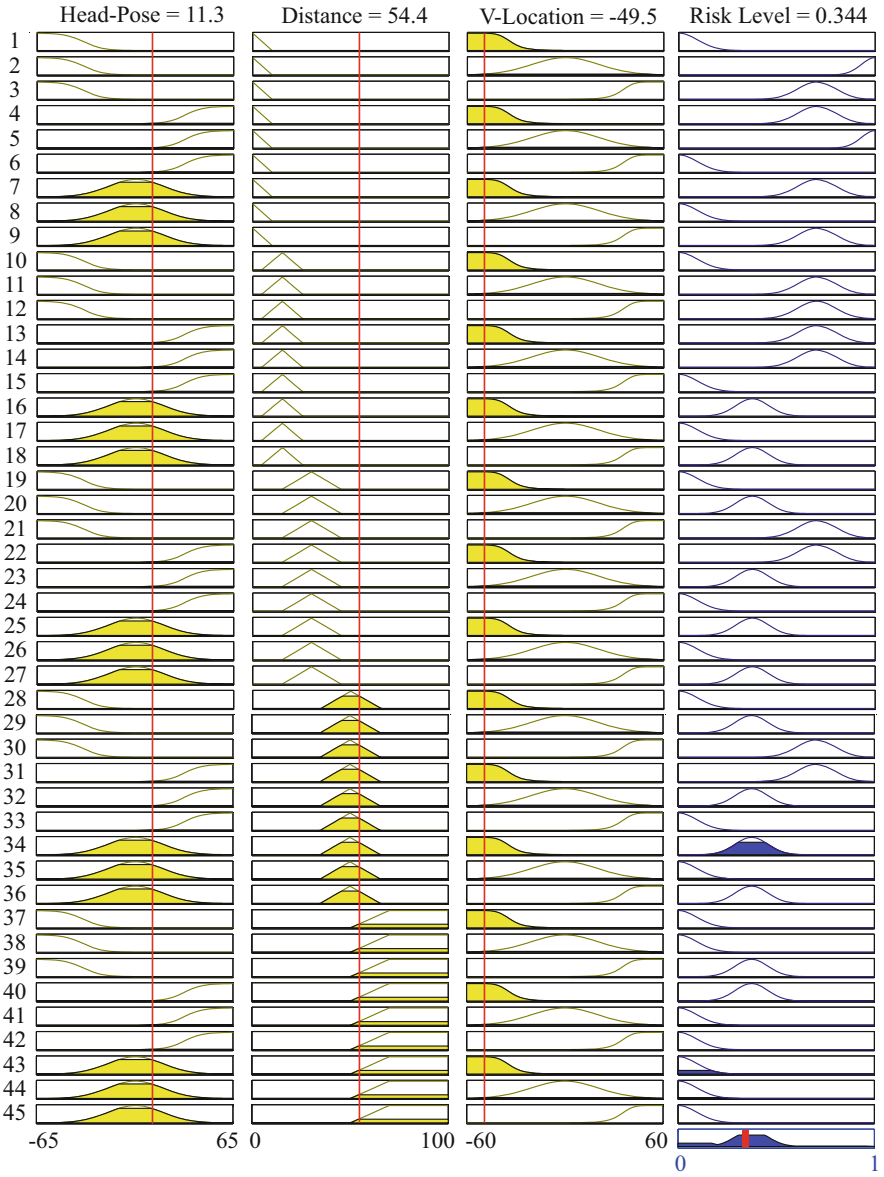

Fig. 8.3 Defuzzified risk-level calculated based on the 45 defined rules and sample input values of Head-yaw = 11.3, V-Distance = 54.4, and V-angle= −49.5

We mounted two monocular VGA-resolution cameras in our test ego-vehicle, one camera facing the driver, and the other camera facing the road. We recorded two simultaneous videos, 20 min-long each, one of the road-scene and the other of the driver-activity. Figure 8.4 shows the processed graphs obtained from a 60-s selected video sequence where the driver was involved in a high-risk driving activity. In the assessed videos, the ego-vehicle was moving in the right lane of the road. When we mention the angle of detected vehicles, our reference is also the right lane of the road.

The graph in Fig. 8.4a shows three detected vehicles in front of the ego-vehicle with respect to their distance to the ego-vehicle. As shown, within seconds 0–4, the vehicle Number 2 (V2) has a very close (apparently *high risk*) distance of $d \approx 5$ m to the ego-vehicle. At the same time ($t \in [0, 4]$), the driver also has a 2-s distraction toward the left-hand side of the road with $yaw = -30°$ (Fig. 8.4c).

However, the graph in Fig. 8.4b shows that V2 has a vehicle-angle of $\approx 42°$ which means V2 is not traversing in the same lane as the ego-vehicle (in our experiment road, vehicles driving in the same lane at a distance of less than 100 *m* cannot have an angular deviation of more than $\pm 8°$ to each other).

For the same reason, V1 also travels in the left lane (V-angle around 20°) therefore no high-risk situation for V1 is expected. V3 with a small angle of about zero is travelling in the right lane (i.e. the same lane as the ego-vehicle); however, Fig. 8.4c confirms a distance of $d \approx 15$ m for V3, thus no critical condition can be expected for V3 either.

Performing the developed fuzzy fusion system (FFS) based on the given inputs, a knowledge-base of 45 rules for the given inputs, the described Mamdani inference system, and centroid defuzzification, Fig. 8.5 depicts the surface-plot of the risk-level for the given inputs; the point shown as *A* is the calculated risk level for $t = 4$, Head-pose $= -30°$, V-angle $= +42°$, where V2 has the closest distance $= 7$ m among the three vehicles in the scene. Similar to the discussion above and our expectation, the FLS is not detecting a *high-risk* situation for $t = 4$.

Further exploring the graphs in Fig. 8.4, we can see that within the time-frame $t = 47$ to 52, the distance of V3 sharply decreases (due to sudden braking); V3 is moving in the same lane as the ego-vehicle, and at the same time the driver has a 3-s distraction to the right-hand side of the road ($yaw \approx +40°$). In this case we would expect the FFS to output a high risk-level to warn and assist the driver to avoid the possibility of an imminent crash (Fig. 8.6, point *B*).

Looking at the provided surface plots, an expert can confirm that the detected *risk-levels* by the FFS are quite rational, based on the input-values already processed from the road-scene and the driver's-status.

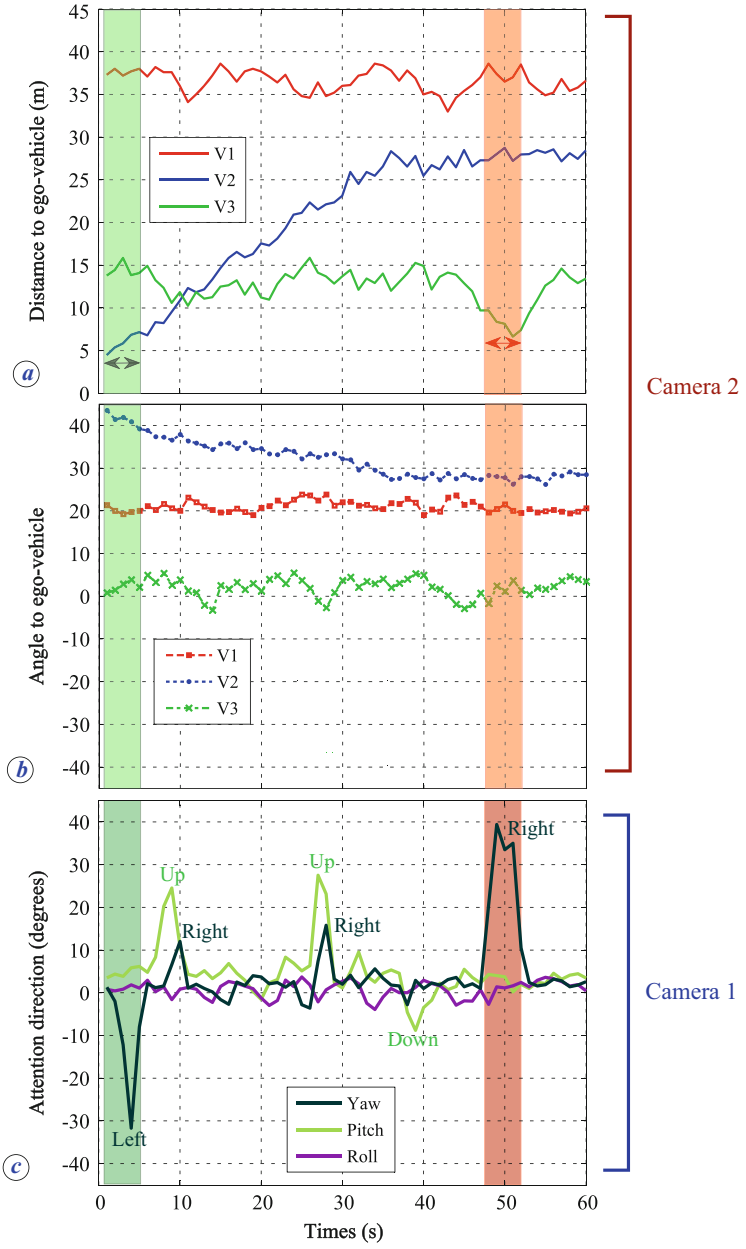

Fig. 8.4 Processed data for simultaneous *driver* and *road* monitoring: (**a**) driver's head-pose, (**b**) detected vehicles' angle to the ego-vehicle, and (**c**) detected vehicles' distance to the ego-vehicle

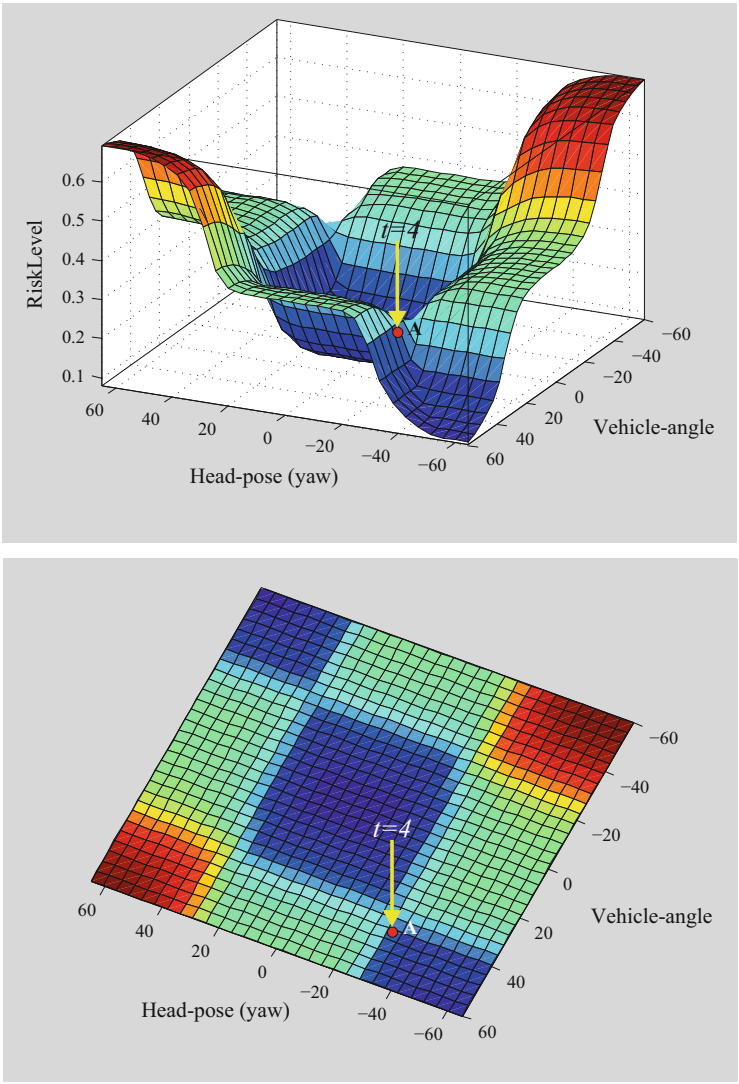

Fig. 8.5 Defuzzified surface plot of risk-level, based on driver's attention direction (head-pose) and ego-vehicle's angle to the lead vehicle

We also evaluated the system using different defuzzification approaches such as Bisector, MOM, LOM and SOM. As can be observed and compared in Figs. 8.7 and 8.8, while the Bisector-based defuzzification performs closer to the centroid output, the other three defuzzification methods underperform for our developed FFS, which is indicated by visible discontinuity and sharp changes in the risk-level

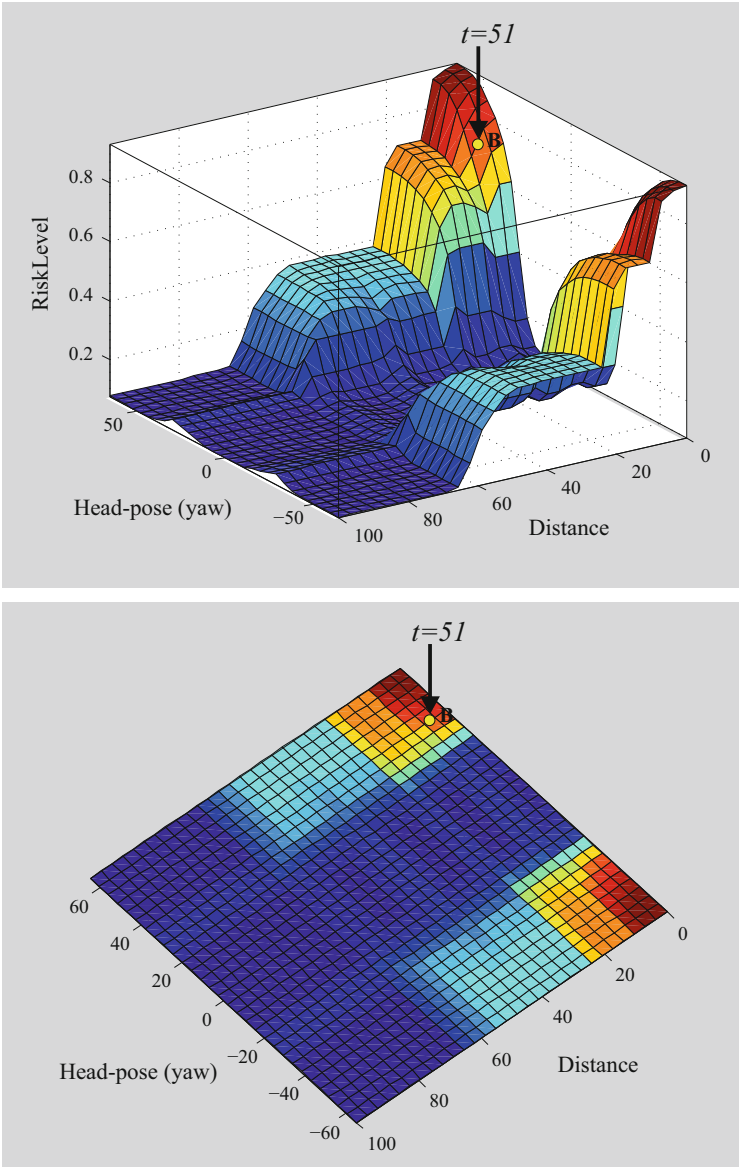

Fig. 8.6 Defuzzified surface plot of risk-level, based on driver's attention direction (head-pose) and ego-vehicle's distance to the lead vehicle

plot. The centroid defuzzification provides a smooth output and a continuity over all regions of the plot, thus the preferred defuzzification method for our FFS.

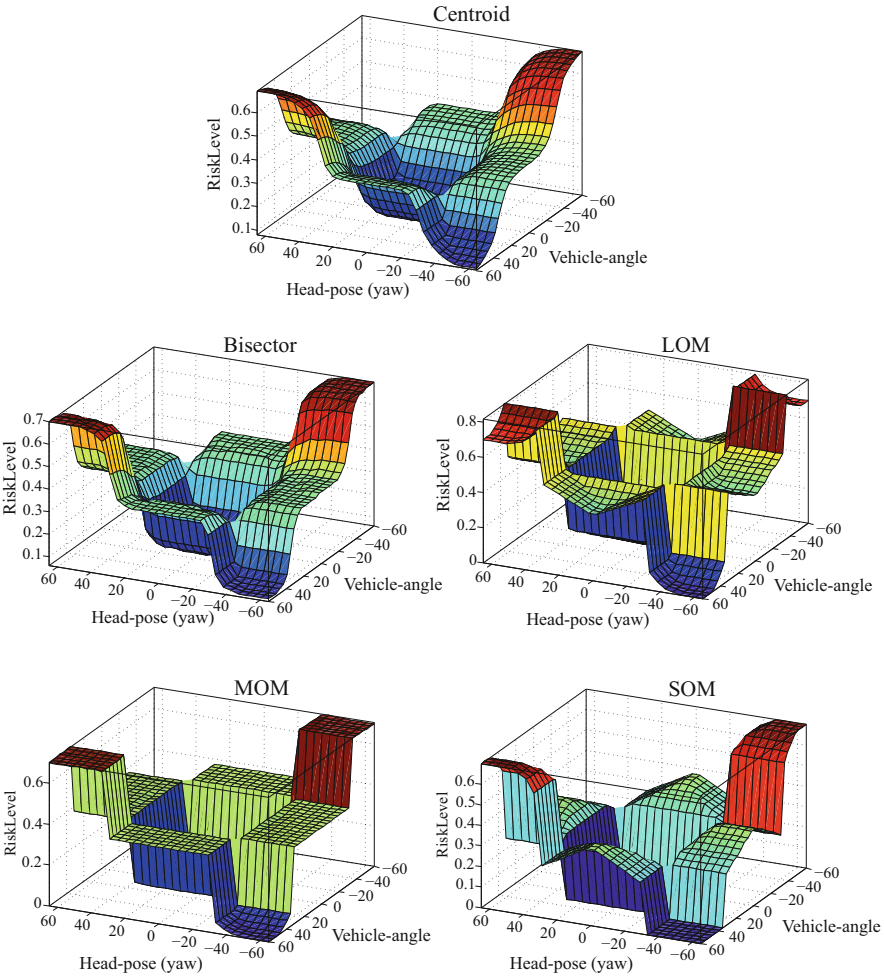

Fig. 8.7 Comparison for the centroid defuzzification method and Bisector, LOM, MOM and SOM methods. Head-pose vs. vehicle-angle

Using a 3.4 GHz Core i7 system, 8 GB RAM, in a 64-bit Windows 7 platform, the entire system (including driver's head-pose estimation, eye-status detection, yawning detection, head-nodding detection, vehicle detection, distance estimation and FFS) performs in real-time at a speed of 21 *fps*. In low-light conditions the processing speed may decrease to 15 *fps*, which is still sufficient for a real-time DAS.

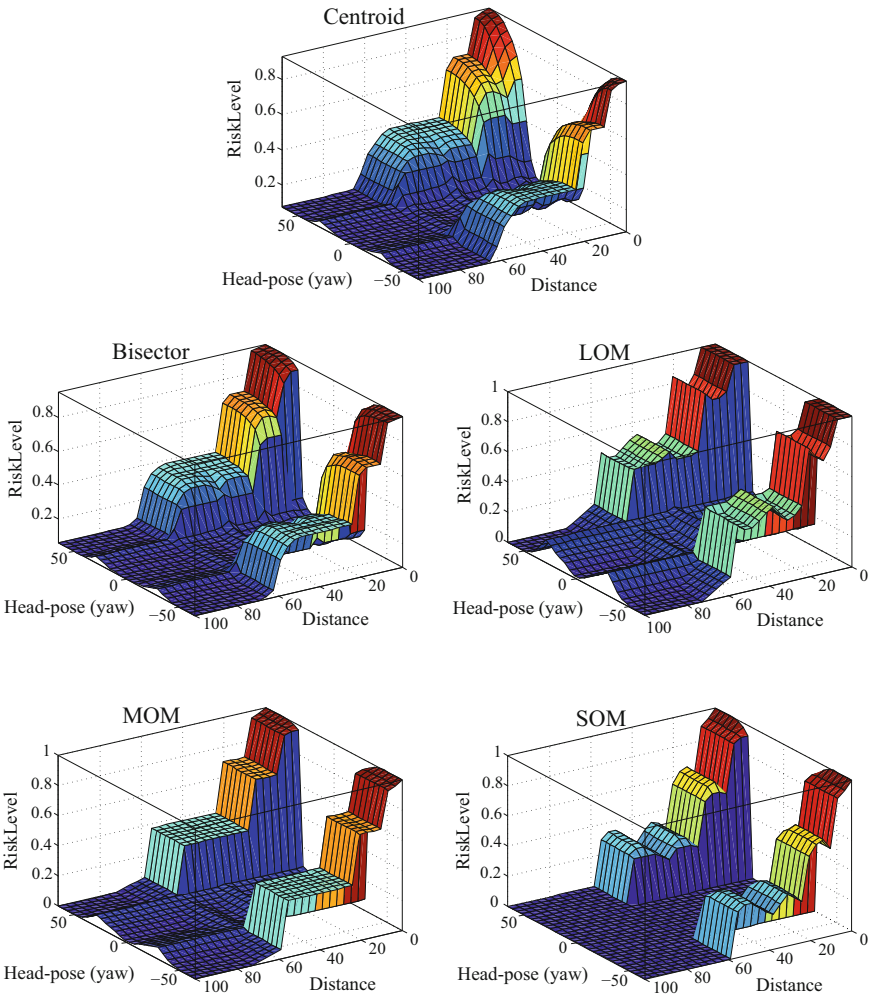

Fig. 8.8 Comparison for the centroid defuzzification method and Bisector, LOM, MOM and SOM methods. Head-pose vs. distance

8.7 Concluding Remarks

The solutions proposed in Chaps. 5, 6 and 7, allowed us to determine the location and distance of vehicles in a road scene (as road hazards), and to successfully monitor the driver's behaviour in terms of head pose (as signs of attention or distraction), eye status, head nodding, and yawning detection (as sign of drowsiness).

Aiming at early detection of collisions, this chapter provided a fuzzy-logic based fusion approach to receiving and processing the data obtained from experiments

discussed in previous chapters, in the form of eight individual inputs. Using fuzzy logic concepts and existing expert knowledge about safe driving, we transformed the crisp input data into fuzzy linguistic forms and defined appropriate membership functions categorized as either "driver behaviour" data or "road condition". We developed a rule base and Mamdani inference engine to analyze the overall state of the driving based on the driver's behaviour and the road conditions. Finally, using a centroid defuzzification, the system was able to output a risk-level in the range [0, 1] for every second of a real-world driving scenario based on the overall status of the driver and the road.

We performed experiments based on recorded videos from real-world driving scenarios with examples of low-risk and high-risk driving conditions. The calculated risk-level outputs of the FFS were satisfactory as they corresponded to our analytical discussions.

Bibliography

1. 6D Vision (2014), www.6d-vision.com
2. A. Alahi, R. Ortiz, P. Vandergheynst, FREAK: fast retina keypoint, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2012), pp. 510–517
3. A. Ali, S. Afghani, Shadow based on-road vehicle detection and verification using Haar wavelet packet transform, in *Proceedings of the IEEE Conference on Information Communication Technologies* (2005), pp. 346–350
4. A.A. Alshennawy, A.A. Aly, Edge detection in digital images using fuzzy logic technique, in *Proceedings of the World Academy of Science: Engineering & Technology* (2009), pp. 178–86
5. J.M. Alvarez, A.M. Lopez, T. Gevers, F. Lumbreras, Combining priors, appearance and context for road detection. IEEE Trans. Intell. Transp. Syst. **15**, 1168–1178 (2014)
6. A.R. Face Database (2013), www2.ece.ohio-state.edu/~aleix/ARdatabase.html
7. H. Badino, U. Franke, D. Pfeiffer, The stixel world – a compact medium level representation of the 3D-world, in *Proceedings of the DAGM – Pattern Recognition* (2009), pp. 51–60
8. Y. Ban, S. Kim, S. Kim, A. Toh, S. Lee, Face detection based on skin color likelihood. Pattern Recognit. **47**, 1573–1585 (2013)
9. A. Bar Hillel, R. Lerner, D. Levi, G. Raz, Recent progress in road and lane detection: a survey. Mach. Vis. Appl. **25**, 727–747 (2014)
10. J. Barnes, Daimler's double take sees machine vision move in-vehicle. ITS International, Nov/Dec 2013
11. N. Barnes, A. Zelinsky, Real-time radial symmetry for speed sign detection, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2004), pp. 566–571
12. A. Barth, Vehicle tracking and motion estimation based on stereo vision sequences. PhD thesis, Bonn University, 2010
13. R. Basri, D.W. Jacobs, Lambertian reflectance and linear subspaces. IEEE IEEE Trans. Pattern Anal. Mach. Intell. **25**, 218–233 (2003)
14. J. Batista, A drowsiness and point of attention monitoring system for driver vigilance, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2007), pp. 702–708
15. K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, H. Winner, Three decades of driver assistance systems: review and future perspectives. IEEE Intell. Transp. Syst. Mag. **6**, 6–22 (2014)
16. L.M. Bergasa, J. Nuevo, M.A. Sotelo, R. Barea, M.E. Lopez, Real-time system for monitoring driver vigilance. IEEE Trans. Intell. Transp. Syst. **7**, 63–77 (2006)

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0

17. D. Bétaille, R. Toledo-Moreo, J. Laneurit, Making an enhanced map for lane location based services, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2008), pp. 711–716
18. BioID Database (2012), www.bioid.com/downloads/software/bioid-face-database
19. C.M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006)
20. A. Borkar, M. Hayes, M.T. Smith, An efficient method to generate ground truth for evaluating lane detection systems, in *Proceedings of the IEEE International Conference on Acoustics Speech Signal Processing* (2010), pp. 1090–1093
21. G.R. Bradski, Computer vision face tracking for use in a perceptual user interface. Intel Technology J. 2nd Quarter (1998)
22. G. Bradski, A. Kaehler, *Learning OpenCV* (O'Reilly Media, Beijing, 2008)
23. T. Brandt, R. Stemmer, A. Rakotonirainy, Affordable visual driver monitoring system for fatigue and monotony. Syst. Man Cybern. **7**, 6451–6456 (2004)
24. L. Breiman, Random forests. Mach. Learn. **45**, 5–32 (2001)
25. A. Briassouli, I. Kompatsiaris, Change detection for temporal texture in the Fourier domain, in *Proceedings of the Asian Conference on Computer Vision*. LNCS 6492 (2010), pp. 149–160
26. K. Briechle, U.D. Hanebeck, Template matching using fast normalized cross correlation, in *Proceedings of the Aerospace Defense Sensing Simulation Controls* (2001), pp. 95–102
27. R.G. Brown, P.Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*, 2nd edn. (Wiley Publishing, New York, 1991)
28. H. Chang, A. Haizhou, L. Yuan, L. Shihong, High-performance rotation invariant multiview face detection. IEEE Trans. Pattern Anal. Mach. Intell. **29**, 671–686 (2007)
29. S.G. Charlton, P.H. Baas, Fatigue, work-rest cycles, and psychomotor performance of New Zealand truck drivers. N. Z. J. Psychol. **30**, 32–39 (2006)
30. Y. Cheng, Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. **17**, 790–799 (1998)
31. S.Y. Cheung, P.P. Varaiya, Traffic surveillance by wireless sensor networks: final report. Institute of Transportation Studies, University of California at Berkeley (2007)
32. J. Choi, Realtime on-road vehicle detection with optical flows and Haar-like feature detectors. University of Illinois at Urbana-Champaign (2007)
33. CMU Face Dataset (Carnegie Mellon University, 2013), www.vasc.ri.cmu.edu/idb/html/face/frontal_images
34. D. Comaniciu, V. Ramesh, P. Meer, Real-time tracking of non-rigid objects using mean shift, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2000), pp. 673–678
35. T.F. Cootes, G.J. Edwards, C.J. Taylor, Active appearance models. IEEE Trans. Pattern Anal. Mach. Intell. **23**, 681–685 (2001)
36. P. Corke, J. Lobo, J. Dias, An introduction to inertial and visual sensing. Int. J. Robot. Res. **26**, 519–535 (2007)
37. C. Cortes, V. Vapnik, Support-vector networks. Mach. learn. **20**, 273–297 (1995)
38. J. Crisman, C. Thorpe, Unscarf: a color vision system for the detection of unstructured roads, in *Proceedings of the IEEE Conference on Robotics Automation*, vol. 3 (1991) pp. 2496–2501
39. W. Cunningham, US requiring back-up cameras in cars by 2018. Road Show by Cnet (2014), www.cnet.com/news/u-s-requiring-back-up-cameras-in-cars-by-2018/
40. A.G. Daimler, First assistance system in the fight against vehicles driving on the wrong side of the road (2013), www.daimler.com/dccom/
41. N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2005), pp. 886–893
42. R. Danescu, S. Nedevschi, Probabilistic lane tracking in difficult road scenarios using stereovision. IEEE Trans. Intell. Transp. Syst. **10**, 272–282 (2009)
43. M. Danilum, M. Rezaei, R. Nicolescu, R. Klette, Eye status based on eyelid detection: a driver assistance system, in *Proceedings of the IEEE International Conference on Computer Vision and Graphics* (2014), pp. 171–178
44. M. Debord, 3 reasons why the auto industry should be terrified of the Google car. Business Insider Australia (2014)

45. D. Dementhon, L. Davis, Model-based object pose in 25 lines of code. Int. J. Comput. Vis. **15**, 123–141 (1995)
46. A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. **39**, 1–38 (1997)
47. H. Deusch, J. Wiest, S. Reuter, M. Szczot, M. Konrad, K. Dietmayer, A random finite set approach to multiple lane detection, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2012), pp. 270–275
48. I. Dimov, Curve fitting, interpolation, and extrapolation. Bulgarian Academy of Sciences, Institute of Information and Communication Technologies (2014)
49. P. Dollar, C. Wojek, B. Schiele, P. Perona, Pedestrian detection: an evaluation of the state of the art. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 743–761 (2012)
50. T. D'Orazio, M. Leo, C. Guaragnella, A. Distante, A visual approach for driver inattention detection. Pattern Recognit. **40**, 2341–2355 (2007)
51. A. Doshi, B. Morris, M. Trivedi, On-road prediction of driver's intent with multimodal sensory cues. IEEE Pervasive Comput. **10**, 22–34 (2011)
52. A. Doshi, M.M. Trivedi, Head and gaze dynamics in visual attention and context learning, in *Proceedings of the IEEE Computer Vision Pattern Recognition Workshops* (2009), pp. 77–84
53. DPM Virtual-World Pedestrian Dataset (CVC-07), Computer Vision Center, Universitat Autònoma de Barcelona (2014), www.cvc.uab.es/adas/site/?q=node/7
54. R.O. Duda, P.E. Hart, Use of the hough transformation to detect lines and curves in pictures. Commun. ACM **15**, 11–15 (1972)
55. M.L. Eichner, T.P. Breckon, Integrated speed limit detection and recognition from real-time video, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2008), pp. 626–631
56. The *.enpeda..* image sequence analysis test site (2014), ccv.wordpress.fos.auckland.ac.nz/eisats/
57. EPFL University, Computer Vision Laboratory: EPFL multi-view car dataset (2012), cvlab.epfl.ch/data/pose
58. S. Escalera, X. Barò, O. Pujol, J. Vitrià, P. Radeva, *Traffic-Sign Recognition Systems* (Springer, London, 2011)
59. The Face of Tomorrow Database (2014), www.faceoftomorrow.org/
60. Y. Fei, M. Adams, S. Roy, V2V wireless communication protocol for rear-end collision avoidance on highways, in *Proceedings of the IEEE Conference on Communications Workshops* (2008), pp. 375–379
61. P.F. Felzenszwalb, R.B. Girshick, D. McAllester, Cascade object detection with deformable part models, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2010), pp. 2241–2248
62. P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**, 1627–1645 (2010)
63. P.F. Felzenszwalb, R.B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**, 1627–1645 (2010)
64. FERET Face Database (2012), www.itl.nist.gov/iad/humanid/feret/
65. Fermat–Torricelli problem, Encyclopedia of Mathematics (2014), www.encyclopediaofmath.org/index.php/Fermat-Torricelli_problem
66. L. Fletcher, A. Zelinsky, Driver inattention detection based on eye gaze—road event correlation. Int. J. Robot. Res. **28**, 774–801 (2009)
67. E.W. Forgy, Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics **21**, 768–769 (1965)
68. U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, R.G. Herrtwich, Making Bertha see, in *Proceedings of the IEEE Conference on Computer Vision Workshops* (2013), pp. 214–221
69. H. Freeman, On the encoding of arbitrary geometric configurations. IRE Trans. Electron. Comput. **10**, 260–268 (1961)

70. Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, in *Proceedings of the European Conference on Computational Learning Theory* (1995), pp. 23–37
71. G.D. Furman, A. Baharav, C. Cahan, S. Akselrod, Early detection of falling asleep at the wheel: a heart rate variability approach, in *Proceedings of the Computers in Cardiology* (2008), pp. 1109–1112
72. T. Gandhi, M.M. Trivedi, Vehicle surround capture: survey of techniques and a novel omni-video-based approach for dynamic panoramic surround maps. IEEE Trans. Intell. Transp. Syst. **7**, 293–308 (2006)
73. T. Gandhi, M.M. Trivedi, Pedestrian protection systems: issues, survey, and challenges. IEEE Trans. Intell. Transp. Syst. **8**, 413–430 (2007)
74. F. Garcia, P. Cerri, A. Broggi, A. De la Escalera, J.M. Armingol, Data fusion for overtaking vehicle detection based on radar and optical flow, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2012), pp. 494–499
75. I. Garcia, S. Bronte, L.M. Bergasa, N. Hernandez, B. Delgado, M. Sevillano, Vision-based drowsiness detector for a realistic driving simulator, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2010), pp. 887–894
76. A. Geiger, J. Ziegler, C. Stiller, StereoScan: dense 3D reconstruction in real-time, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2011), pp. 963–968
77. T. Gernoth, K. Martinez, A. Gooben, R. Grigat, Facial pose estimation using active appearance models and a generic face model, in *Proceedings of the IEEE International Conference on Computer Vision Theory and Applications* (2010), pp. 499–506
78. D. Geronimo, A.M. Lopez, *Vision-Based Pedestrian Protection Systems for Intelligent Vehicles.* Springer Briefs in Computer Science (Springer, New York, 2013)
79. H. Grabner, T.T. Nguyen, B. Gruber, H. Bischof, On-line boosting-based car detection from aerial images. ISPRS J. Photogramm. Remote Sens. **63**, 382–396 (2008)
80. G. Grubb, E. Jakobsson, A. Beutner, M. Ahrholdt, S. Bergqvist, Automatic queue assistance to aid under-loaded drivers, in *Proceedings of the ITS World Congress Exhibition Intelligent Transport Systems Services* (2009), pp. 1–8
81. H.Z. Gu, S.Y. Lee, Car model recognition by utilizing symmetric property to overcome severe pose variation. Mach. Vis. Appl. **24**, 255–274 (2013)
82. R.A. Gupta, W. Snyder, W.S. Pitts, Concurrent visual multiple lane detection for autonomous vehicles, in *Proceedings of the IEEE Conference on Robotics Automation* (2010), pp. 2416–2422
83. R. Haeusler, R. Klette, Disparity confidence measures on engineered and outdoor data, in *Proceedings of the Iberoamerican Congress Pattern Recognition*. LNCS 7441 (2012), pp. 624–631
84. S. Han, Y. Han, H. Hahn, Vehicle detection method using Haar-like feature on real time system. World Acad. Sci. Eng. Technol. (2009), pp. 455–459
85. A. Haselhoff, A. Kummert, A vehicle detection system based on Haar and triangle features, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2009), pp. 261–266
86. A. Haselhoff, A. Kummert, G. Schneider, Radar-vision fusion for vehicle detection by means of improved Haar-like feature and AdaBoost approach, in *Proceedings of the European Association Signal Processing* (2007), pp. 2070–2074
87. Heidelberg robust vision challenge at ECCV (2012), hci.iwr.uni-heidelberg.de/Static/challenge2012/
88. S. Hermann, R. Klette, The naked truth about cost functions for stereo matching. MI-tech report 33, The University of Auckland (2009), www.mi.auckland.ac.nz/tech-reports/MItech-TR-33.pdf
89. S. Hermann, R. Klette, Iterative semi-global matching for robust driver assistance systems, in *Proceedings of the Asian Conference on Computer Vision*. LNCS 7726 (2012), pp. 465–478
90. H. Hirschmüller, Accurate and efficient stereo processing by semi-global matching and mutual information, in *Proceedings of the IEEE Computer Vision Pattern Recognition*, vol. 2 (2005) pp. 807–814

91. H. Hirschmüller, D. Scharstein, Evaluation of stereo matching costs on images with radiometric differences. IEEE Trans. Pattern Anal. Mach. Intell. **31**, 1582–1599 (2009)
92. B. Hongliang, W. Jianping, L. Changpin, Motion and Haar-like features based vehicle detection, in *Proceedings of the Multi-Media Modelling* (2006), pp. 1–4
93. F. Huang, R. Klette, City-scale modeling towards street navigation applications. J. Inform. Converg. Commun. Eng. **10**, 411–419 (2012)
94. I. Iancu, *A Mamdani Type Fuzzy Logic Controller*. Fuzzy Logic: Controls, Concepts, Theories and Applications (InTech, Rijeka, 2012), pp. 55–54
95. M.I. Imprialou, M. Quddus, D.E. Pitfield, High accuracy crash mapping using fuzzy logic. Transp. Res. C Emerg. Technol. **42**, 107–120 (2014)
96. INRIA Person Data Set, lear.inrialpes.fr/data (2005)
97. O.L.R. Jacobs, *Introduction to Control Theory*, 2nd edn. (Oxford University Press, Oxford/New York, 1993)
98. R. Jain, K. Rangachar, G.S. Brian, *Machine Vision* (McGraw-Hill, New York, 1995)
99. A. Jazayeri, C. Hongyuan, Z. Jiang Yu, M. Tuceryan, Vehicle detection and tracking in car video based on motion model. IEEE Trans. Intell. Transp. Syst. **12**, 583–595 (2011)
100. S.H. Jeong, C.G. Choi, J.N. Oh, P.J. Yoon, B.S. Kim, M. Kim, K.H. Lee, Low cost design of parallel parking assist system based on an ultrasonic sensor. Int. J. Automot. Technol. **11**, 409–416 (2010)
101. O. Jesorsky, K.J. Kirchberg, R.W. Frischholz, Robust face detection using the Hausdorff distance, in *Proceedings of the International Conference on Audio-and Video-Based Biometric Person Authentication* (Springer, Berlin/Heidelberg, 2001), pp. 90–95
102. X. Jian-Feng, X. Mei, Z. Wei, Driver fatigue detection based on head gesture and PERCLOS, in *Proceedings of the Wavelet Active Media Technology and Information Processing* (2012), pp. 128–131
103. R. Jiang, R. Klette, T. Vaudrey, S. Wang, New lane model and distance transform for lane detection and tracking, in *Proceedings of the International Conference on Computer Analysis Images Patterns*. LNCS 5702 (2009), pp. 1044–1052
104. W. Jianxin, S.C. Brubaker, M.D. Mullin, J.M. Rehg, Fast asymmetric learning for cascade face detection. IEEE Trans. Pattern Anal. Mach. Intell. **30**, 369–382 (2008)
105. X. Jie, H. Chen, W. Ding, C. Zhao, J. Morris, Robust optical flow for driver assistance, in *Proceedings of the Image and Vision Computing New Zealand* (2010), pp. 1–7
106. H. Jing, S.R. Kumar, M. Mitra, Z. Wei-Jing, R. Zabih, Image indexing using color correlograms, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (1997), pp. 762–768
107. B. Jun, D. Kim, Robust face detection using local gradient patterns and evidence accumulation. Pattern Recognit. **45**, 3304–3316 (2012)
108. S.-J. Jung, H.-S. Shin, W.-Y. Chung, Driver fatigue and drowsiness monitoring system with embedded electrocardiogram sensor on steering wheel. IET Intell. Transp. Syst. **8**, 43–50 (2014)
109. R.K. Jurgen (ed.), *Adaptive Cruise Control* (SAE International, Warrendale, 2006)
110. R.E. Kalman, A new approach to linear filtering and prediction problems. J. Basic Eng. **82**, 35–45 (1960)
111. A. Kasinski, A. Schmidt, The architecture and performance of the face and eyes detection system based on the Haar cascade classifiers. Pattern Anal. Appl. **13**, 197–211 (2010)
112. W. Khan, V. Suaste, D. Caudillo, R. Klette, Belief propagation stereo matching compared to iSGM on binocular or trinocular video data, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2013), pp. 791–796
113. L.W. Kheng, Mean Shift Tracking. Department of Computer Science, National University of Singapore (2012), pp. 1–28
114. A. Kheyrollahi, T.P. Breckon, Automatic real-time road marking recognition using a feature driven approach. Mach. Vis. Appl. **23**, 123–133 (2012)
115. Z. Kim, Robust lane detection and tracking in challenging scenarios. IEEE Trans. Intell. Transp. Syst. **9**, 16–26 (2008)

116. N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic hough transform. Pattern Recognit. **24**, 303–316 (1991)
117. KITTI Benchmark website: car detection benchmark (2013), www.cvlibs.net/datasets/kitti/eval_object.php
118. The KITTI Vision Benchmark Suite (2013), www.cvlibs.net/datasets/kitti/
119. R. Klette, *Concise Computer Vision: An Introduction into Theory and Algorithms* (Springer, London, 2014)
120. R. Klette, N. Krüger, T. Vaudrey, K. Pauwels, M. van Hulle, S. Morales, F. Kandil, R. Haeusler, N. Pugeault, C. Rabe, M. Lappe, Performance of correspondence algorithms in vision-based driver assistance using an online image sequence database. IEEE Trans. Veh. Technol. **60**, 2012–2026 (2011)
121. A. Klir, J. George, *Fuzzy Sets and Fuzzy Logic* (Prentice Hall, Upper Saddle River, 1995)
122. G.J. Klir, B. Yuan, *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lofti A. Zadeh* (World Scientific Publishing, Singapore/River Edge, 1996)
123. L. Kneip, M. Chli, R. Siegwart, Robust real-time visual odometry with a single camera and an IMU, in *Proceedings of British Machine Vision Conference* (2011), pp. 16.1–16.11
124. D. Koks, S. Challa, An introduction to Bayesian and Dempster–Shafer data fusion. Technical report, TR, DSTO-TR-1436, DSTO Systems Sciences Laboratory (2005)
125. P. Kovesi, Phase preserving denoising of images, in *Proceedings of DICTA* (1999), pp. 212–217
126. V. Krüger, G. Sommer, Gabor wavelet networks for efficient head pose estimation. Image Vis. Comput. **20**, 665–672 (2002)
127. O. Langner, R. Dotsch, G. Bijlstra, D.H.J. Wigboldus, S.T. Hawk, A. van Knippenberg, Presentation and validation of the Radboud faces database. Cogn. Emot. **24**, 1377–1388 (2010)
128. S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2006), pp. 2169–2178
129. L. Le, A. Festag, R. Baldessari, W. Zhang, V2X communication and intersection safety, in *Advanced Microsystems for Automotive Applications*. VDI-Buch (Springer, Berlin/Heidelberg, 2009), pp. 97–107
130. K. Lee, J. Ho, D. Kriegman, Acquiring linear subspaces for face recognition under variable lighting. IEEE Trans. Pattern Anal. Mach. Intell. **27**, 684–698 (2005)
131. V. Lepetit, F. Moreno-Noguer, P. Fua, EPnP: an accurate O(n) solution to the PnP problem. Int. J. Comput. Vis. **81**, 155–166 (2009)
132. S.Z. Li, A.K. Jain, *Handbook of Face Recognition* (Springer, New York, 2011)
133. S.Z. Li, Z. Zhang, Floatboost learning and statistical face detection. IEEE Trans. Pattern Anal. Mach. Intell. **26**, 1112–1123 (2004)
134. R. Lienhart, A. Kuranov, V. Pisarevsky, Empirical analysis of detection cascades of boosted classifiers for rapid object detection, in *Proceedings of Joint Pattern Recognition Symposium* (2003), pp. 297–304
135. R. Lienhart, J. Maydt, An extended set of Haar-like features for rapid object detection, in *Proceedings of International Conference on Image Processing*, vol. 1 (2002)
136. H. Lili, M. Barth, Tightly-coupled LIDAR and computer vision integration for vehicle detection, in *Proceedings of Intelligent Vehicles Symposium* (2009), pp. 604–609
137. Y. Lin, F. Guo, S. Li, Road obstacle detection in stereo vision based on UV-disparity. J. Inf. Comput. Sci. **11**, 1137–1144 (2014)
138. P. Lindner, G. Wanielik, 3D LIDAR processing for vehicle safety and environment recognition, in *Proceedings of IEEE Workshop Computational Intelligence in Vehicles and Vehicular Systems* (2009), pp. 66–71
139. C. Liu, H.Y. Shum, Kullback–Leibler boosting, in *Proceedings of IEEE Computer Vision Pattern Recognition* (2003), pp. 587–594
140. Y.K. Liu, B. Žalik, An efficient chain code with Huffman coding. Pattern Recognit. **38**, 553–557 (2005)

141. M. Ljung, H. Fagerlind, P. Lövsund, J. Sandin, Accident investigations for active safety at CHALMERS – new demands require new methodologies. Veh. Syst. Dyn. **45**, 881–894 (2007)
142. S.P. Lloyd, Least square quantization in PCM. Bell Telephone Laboratories Paper (1982)
143. C. Long, X. Wang, G. Hua, M. Yang, Y. Lin, Accurate object detection with location relaxation and regionlets relocalization, in *Proceedings of Asian Conference of Computer Vision* (2014), pp. 260–275
144. A.M. Lopez, J. Serrat, C. Canero, F. Lumbreras, T. Graf, Robust lane markings detection and road geometry computation. Int. J. Automot. Technol. **11**, 395–407 (2010)
145. B.S. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in *Proceedings of International Joint Conference on Artificial Intelligence*, vol. 2 (1981), pp. 674–679
146. M.J. Lyons, S. Akamatsu, M. Kamachi, J. iro Gyoba, The Japanese female facial expression database (2013), www.kasrl.org/jaffe.html
147. A. Majumder, L. Behera, V.K. Subramanian, Automatic and robust detection of facial features in frontal face images, in *Proceedings of International Conference on Computer Modelling and Simulation* (2011), pp. 331–336
148. A.M. Malla, P.R. Davidson, P.J. Bones, R. Green, R.D. Jones, Automated video-based measurement of eye closure for detecting behavioral microsleep, in *Proceedings of IEEE International Conference on Engineering Medicine Biology Society* (2010), pp. 6741–6744
149. E.H. Mamdani, Application of fuzzy logic to approximate reasoning using linguistic synthesis. IEEE Trans. Comput. **100**, 1182–1191 (1977)
150. J. Marin, D. Vazquez, A.M. Lopez, J. Amores, B. Leibe, Random forests of local experts for pedestrian detection, in *Proceedings of IEEE International Conference on Computer Vision* (2013), pp. 2592–2599
151. M. Marron, J.C. Garcia, M.A. Sotelo, M. Cabello, D. Pizarro, F. Huerta, J. Cerro, Comparing a Kalman filter and a particle filter in a multiple objects tracking application, in *Proceedings of IEEE International Symposium on Intelligent Signal Processing* (2007), pp. 1–6
152. P. Martin Larsen, Industrial applications of fuzzy logic control. Int. J. Man-Mach. Stud. **12**, 3–10 (1980)
153. P. Martins, J. Batista, Monocular head pose estimation, in *Proceedings of International Conference on Image Analysis Recognition* (2008), pp. 357–368
154. H. Masnadi-Shirazi, N. Vasconcelos, Cost-sensitive boosting. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 294–309 (2011)
155. J. Matas, C. Galambos, J. Kittler, Robust detection of lines using the progressive probabilistic Hough transform. Comput. Vis. Image Underst. **78**, 119–137 (2000)
156. J.C. McCall, M.M. Trivedi, Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. IEEE Trans. Intell. Transp. Syst. **7**, 20–37 (2006)
157. J.C. McCall, M.M. Trivedi, Driver behavior and situation aware brake assistance for intelligent vehicles. Proc. IEEE **95**, 374–387 (2007)
158. G.J. McLachlan, T. Krishnan, The EM algorithm and its extensions. J. Am. Stat. Assoc. **93**, 403–405 (1997)
159. F. Meng-Yin, H. Yuan-Shui, A survey of traffic sign recognition, in *Proceedings of International Conference on Wavelet Analysis Pattern Recognition* (2010), pp. 119–124
160. Mercedes Benz Tech Centre, Distronic Plus with Steering Assist (2013), techcenter.mercedes-benz.com/en/distronic_plus_steering_assist/detail.html
161. T.P. Michalke, F. Stein, U. Franke, Towards a closer fusion of active and passive safety: optical flow-based detection of vehicle side collisions, in *Proceedings of IEEE Intelligent Vehicle Symposium* (2011), pp. 181–188
162. V. Milanés, J. Pérez, J. Godoy, E. Onieva, A fuzzy aid rear-end collision warning/avoidance system. Expert Syst. Appl. **39**, 9097–9107 (2012)
163. M. Miyaji, M. Danno, H. Kawanaka, K. Oguri, Driver's cognitive distraction detection using AdaBoost on pattern recognition basis, in *Proceedings of IEEE International Conference on Vehicular Electronics Safety* (2008), pp. 51–56

164. M. Miyaji, H. Kawanaka, K. Oguri, Effect of pattern recognition features on detection for driver's cognitive distraction, in *Proceedings of IEEE International Conference on Intelligent Transportation Systems* (2010), pp. 605–610
165. A. Møgelmose, M.M. Trivedi, T.B. Moeslund, Vision based traffic sign detection and analysis for intelligent driver assistance systems: perspectives and survey. IEEE Trans. Intell. Transp. Syst. **13**, 1484–1497 (2012)
166. G. Monteiro, M. Ribeiro, J. Marcos, J. Batista, Wrongway drivers detection based on optical flow, in *Proceedings of IEEE International Conference on Image Processing*, vol. 5 (2007), pp. 141–144
167. S. Morales, Performance evaluation tools for stereo vision analysis in uncontrolled environments. PhD thesis, Department of Computer Science, The University of Auckland (2012)
168. S. Morales, R. Klette, A third eye for performance evaluation in stereo sequence analysis, in *Proceedings of International Conference on Computer Analysis Images Patterns*. LNCS 5702 (2009), pp. 1078–1086
169. M. Mori, C. Miyajima, P. Angkititrakul, T. Hirayama, L. Yiyang, N. Kitaoka, K. Takeda, Measuring driver awareness based on correlation between gaze behavior and risks of surrounding vehicles, in *Proceedings of IEEE Conference on Intelligent Transportation Systems* (2012), pp. 644–647
170. S. Motoyama, T. Ohta, T. Watanabe, Y. Ito, Development of lane departure warning system, in *Proceedings of ITS World Congress* (2000), pp. 1–8
171. F. Moutarde, B. Stanciulescu, A. Breheret, Real-time visual detection of vehicles and pedestrians with new efficient AdaBoost features, in *Proceedings of 2nd Workshop Planning Perception Navigation Intelligent Vehicles* (2008), pp. 1–7
172. MUCT Face Dataset (2014), www.milbo.org/muct/
173. S. Müller-Schneiders, C. Nunn, M. Meuter, Performance evaluation of a real time traffic sign recognition system, in *Proceedings of IEEE Conference on Intelligent Vehicles Symposium* (2008), pp. 79–84
174. E. Murphy-Chutorian, M.M. Trivedi, Head pose estimation in computer vision: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **31**, 607–626 (2009)
175. E. Murphy-Chutorian, M.M. Trivedi, Head pose estimation and augmented reality tracking: an integrated system and evaluation for monitoring driver awareness. IEEE Trans. Intell. Transp. Syst. **11**, 300–311 (2010)
176. W. Murray, Improving work-related road safety in New Zealand – a research report. Department of Labour, Wellington (2007)
177. J.E. Naranjo, M.A. Sotelo, C. Gonzalez, R. Garcia, M.A. Sotelo, Using fuzzy logic in automated vehicle control. Intell. Syst. **22**, 36–45 (2007)
178. National Highway Traffic Safety Administration, Traffic safety facts, U.S. Department of Transportation (2013)
179. New Zealand Ministry of Transport, Motor vehicle crash fact sheets (2010)
180. J. Ocken, V2X telematics: taking ADAS to the next level (2011), analysis.telematicsupdate.com/v2x-safety/v2x-telematics-taking-adas-next-level
181. E. Ohn-Bar, A. Tawari, S. Martin, M.M. Trivedi, On surveillance for safety critical events: in-vehicle video networks for predictive driver assistance systems. Comput. Vis. Image Underst. **134**, 130–140 (2015)
182. E. Ohn-Bar, M.M. Trivedi, Beyond just keeping hands on the wheel: towards visual interpretation of driver and motion patterns, in *Proceedings of IEEE Conference on Intelligent Transportation Systems* (2014), pp. 1245–1250
183. E. Ohn-Bar, M. Trivedi, Fast and robust object detection using visual subcategories, in *Proceedings of IEEE Computer Vision Pattern Recognition Workshops* (2014), pp. 179–184
184. T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 971–987 (2002)
185. R. O'Malley, M. Glavin, E. Jones, Vehicle detection at night based on tail-light detection, in *Proceedings of International Symposium on Vehicular Computing Systems*, vol. 2224 (2008)

186. R. O'Malley, E. Jones, M. Glavin, Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions. IEEE Trans. Intell. Transp. Syst. **11**, 453–462 (2010)
187. M. Osadchy, Y. Le Cun, M.L. Miller, Synergistic face detection and pose estimation with energy-based models. J. Mach. Learn. Res. **8**, 1197–1215 (2007)
188. H. Pan, Y. Zhu, L. Xia, Efficient and accurate face detection using heterogeneous feature descriptors and feature selection. Comput. Vis. Image Underst. **117**, 12–28 (2013)
189. C. Papageorgiou, T. Poggio, A trainable system for object detection. Int. J. Comput. Vis. **38**, 15–33 (2000)
190. M. Pavlic, G. Rigoll, S. Ilic, Classification of images in fog and fog-free scenes for use in vehicles, in *Proceedings of IEEE Intelligent Vehicles Symposium* (2013), pp. 481–486
191. M.T.R. Peiris, R.D. Jones, P.R. Davidson, P.J. Bones, Detecting behavioral microsleeps from EEG power spectra, in *Proceedings of IEEE Conference on Engineering Medicine Biology Society* (2006), pp. 5723–5726
192. M.T.R. Peiris, R.D. Jones, P.R. Davidson, G.J. Carroll, P.J. Bones, Frequent lapses of responsiveness during an extended visuomotor tracking task in non-sleep-deprived subjects. J. Sleep Res. **15**, 291–300 (2006)
193. F. Peyret, J. Laneurit, D. Bétaille, A novel system using enhanced digital maps and WAAS for a lane-level positioning, in *Proceedings of World Congress Intelligent Transport Systems* (2008), pp. 1–12
194. M.T. Pham, T.J. Cham, Fast training and selection of Haar features using statistics in boosting-based face detection, in *Proceedings of IEEE International Conference on Computer Vision* (2007), pp. 1–7
195. M.T. Pham, Y. Gao, V.T.D. Houng, T.J. Cham, Fast polygonal integration and its application in extending Haar-like features to improve object detection, in *Proceedings of IEEE Computer Vision Pattern Recognition* (2010), pp. 942–949
196. P.J. Phillips, H. Moon, S.A. Rizvi, P.J. Rauss, The FERET evaluation methodology for face-recognition algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **22**, 1090–1104 (2000)
197. PICS Image Database, Psychology Department, University of Stirling (2011), pics.psych.stir.ac.uk/
198. D. Ponsa, A.M. Lopez, F. Lumbreras, J. Serrat, T. Graf, 3D vehicle sensor based on monocular vision, in *Proceedings of IEEE Conference on Intelligent Transportation Systems* (2005), pp. 1096–1101
199. D. Ponsa, A.M. Lopez, J. Serrat, F. Lumbreras, T. Graf, Multiple vehicle 3D tracking using an unscented Kalman filter, in *Proceedings of IEEE Conference on Intelligent Transportation Systems* (2005), pp. 1108–1113
200. E. Portouli, E. Bekiaris, V. Papakostopoulos, N. Maglaveras, On-road experiment for collecting driving behavioural data of sleepy drivers. Somnologie Schlafforschung Schlafmedizin **11**, 259–267 (2007)
201. R. Preiß, C. Gruner, T. Schilling, H. Winter, Mobile vision – developing and testing of visual sensors for driver assistance systems, in *Proceedings of Advanced Microsystems Automotive Applications*. VDI-Buch (2004), pp. 95–107
202. C. Premebida, G. Monteiro, U. Nunes, P. Peixoto, A Lidar and vision-based approach for pedestrian and vehicle detection and tracking, in *Proceedings of IEEE Conference on Intelligent Transportation Systems Conference* (2007), pp. 1044–1049
203. Z.M. Qian, H.X. Shi, J.K. Yang, Video vehicle detection based on local feature. Adv. Mater. Res. **186**, 56–60 (2011)
204. L. Qiong, P. Guang-zheng, A robust skin color based face detection algorithm, in *Proceedings of International Asian Conference on Informatics Control Automation Robotics* (2010), pp. 525–528
205. H. Rein-Lien, M. Abdel-Mottaleb, A.K. Jain, Face detection in color images. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 696–706 (2002)
206. M. Rezaei, R. Klette, 3D cascade of classifiers for open and closed eye detection in driver distraction monitoring, in *Proceedings of Computer Analysis Images Patterns*. LNCS 6855 (2011), pp. 171–179

207. M. Rezaei, R. Klette, Simultaneous analysis of driver behaviour and road condition for driver distraction detection. Int. J. Image Data Fusion **2**, 217–236 (2011)
208. M. Rezaei, R. Klette, Adaptive Haar-like classifier for eye status detection under non-ideal lighting conditions, in *Proceedings of International Conference on Image Vision Computing New Zealand* (ACM, 2012), pp. 521–526
209. M. Rezaei, R. Klette, Artistic rendering of human portraits paying attention to facial features, in *Proceedings of International Conference on Arts Technology*. LNCS 101 (Springer, 2012), pp. 90–99
210. M. Rezaei, R. Klette, Novel adaptive eye detection and tracking for challenging lighting condition, in *Proceedings of Asian Conference on Computer Vision Workshops*. LNCS 7729 (2013), pp. 427–440
211. M. Rezaei, R. Klette, Look at the driver, look at the road: No distraction! No accident! in *Proceedings of IEEE Computer Vision Pattern Recognition* (2014), pp. 129–136
212. M. Rezaei, J. Lin, R. Klette, Hybrid filter blending to maintain facial expressions in rendered human portraits. Int. J. Arts Technol. **7**, 128–147 (2014)
213. M. Rezaei, M. Terauchi, Vehicle detection based on multi-feature clues and Dempster–Shafer fusion theory, in *Proceedings of Pacific-Rim Symposium on Image Video Technology* (2013), pp. 60–72
214. M. Rezaei, M. Terauchi, iROADS dataset (Intercity Roads and Adverse Driving Scenarios), EISATS, Set 10 (2014), ccv.wordpress.fos.auckland.ac.nz/eisats/set-10/
215. M. Rezaei, M. Terauchi, R. Klette, Monocular vehicle detection and distance estimation under challenging lighting conditions. IEEE Trans. Intell. Transp. Syst. **16**, 2723–2743 (2014)
216. M. Rezaei, H. Ziaei Nafchi, S. Morales, Global Haar-like features: a new extension of classic Haar features for efficient face detection in noisy images, in *Proceedings of Pacific-Rim Symposium on Image Video Technology*. LNCS 8333 (2013), pp. 302–313
217. B. Ristic, S. Arulampalam, N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications* (Artech House, London, 2004)
218. G. Ros, A. Sappa, D. Ponsa, A.M. Lopez, Visual SLAM for driverless cars: a brief survey, in *Proceedings of IEEE Intelligent Symposium on Workshop* (2012), pp. 1–6
219. G. Ros, S. Ramos, M. Granados, A.H. Bakhtiary, D. Vazquez, A.M. Lopez, in *Proceedings of IEEE Winter Conference on Applications Computer Vision* (2015), pp. 231–238
220. T.J. Ross, *Fuzzy Logic with Engineering Applications* (Wiley, Hoboken 2009)
221. E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in *Proceedings of European Conference on Computer Vision* (2006), pp. 430–443
222. J.C. Rubio, J. Serrat, A.M. Lopez, D. Ponsa, Multiple target tracking for intelligent headlights control. IEEE Trans. Intell. Transp. Syst. **13**, 594–609 (2012)
223. E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: an efficient alternative to SIFT or SURF, in *Proceedings of IEEE International Conference on Computer Vision* (2011), pp. 2564–2571
224. H. Ryu, J. Yoon, S. Chun, S. Sull, Coarse-to-fine classification for image-based face detection, in *Proceedings of International Conference on Image Video Retrieval* (2006), pp. 291–299
225. M.J. Saberian, N. Vasconcelos, Boosting classifier cascades, in *Proceedings of the Neural Information Processing Systems* (2010)
226. M.J. Saberian, N. Vasconcelos, Learning optimal embedded cascades. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 2005–2018 (2012)
227. D. Santos, P.L. Correia, Car recognition based on back lights and rear view features, in *Proceedings of the Image Analysis Multimedia Interactive Services* (2009), pp. 137–140
228. M. Sarshar, M. Rezaei, A novel system for advanced driver assistance systems, in *Proceedings of the IEEE Systems Conference* (2013), pp. 529–534
229. S. Schmidt, R. Steele, T.S. Dillon, Towards usage policies for fuzzy inference methodologies for trust and QoS assessment, in *Proceedings of the Computational Intelligence Theory Applications* (2006) pp. 263–274

230. M. Schreier, V. Willert, Robust free space detection in occupancy grid maps by methods of image analysis and dynamic B-spline contour tracking, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2012), pp. 514–521
231. Y. Sekimoto, Y. Matsubayashi, H. Yamada, R. Imai, T. Usui, H. Kanasugi, Light weight lane positioning of vehicles using a smartphone GPS by monitoring the distance from the center line, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2012), pp. 1561–1565
232. A. Selloum, D. Bétaille, E. Le Carpentier, F. Peyret, Robustification of a map aided location process using road direction, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2010), pp. 1504–1510
233. R. Senaratne, B. Jap, S. Lal, A. Hsu, S. Halgamuge, P. Fischer, Comparing two video-based techniques for driver fatigue detection: classification versus optical flow approach. Mach. Vis. Appl. **22**, 597–618 (2011)
234. G. Shafer, *A Mathematical Theory of Evidence* (Princeton University Press, Princeton, 1976)
235. J. Shi, C. Tomasi, Good features to track, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (1994), pp. 593–600
236. B.-S. Shin, D. Caudillo, R. Klette, Evaluation of two stereo matchers on long real-world video sequences. Pattern Recognit. **48**, 113–1124 (2014)
237. B.-S. Shin, Z. Xu, R. Klette, Visual lane analysis and higher-order tasks: a concise review. Mach. Vis. Appl. **25**, 1519–1547 (2014)
238. J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, Real-time human pose recognition in parts from single depth images. Stud. Comput. Intell. **411**, 119–135 (2013)
239. M.H. Sigari, Driver hypo-vigilance detection based on eyelid behavior, in *Proceedings of the International Conference on Advances Pattern Recognition* (2009), pp. 426–429
240. S. Sivaraman, M.M. Trivedi, Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking and behavior analysis. IEEE Trans. Intell. Transp. Syst. **14**, 1773–1795 (2013)
241. S. Sivaraman, M.M. Trivedi, Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking, and behavior analysis. IEEE Conf. Intell. Transp. Syst. **14**, 1773–1795 (2013)
242. A. Soria-Frisch, R. Verschae, A. Olano, Fuzzy fusion for skin detection. Fuzzy Sets Syst. **158**, 325–336 (2007)
243. M.Á. Sotelo, J. Barriga, Blind spot detection using vision for automotive applications. J. Zhejiang Univ. Sci. A **9**, 1369–1372 (2008)
244. G.P. Stein, Y. Gdalyahu, A. Shashua, Stereo-assist: top-down stereo for driver assistance systems, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2010), pp. 723–730
245. V. Struc, B. Vesnicer, F. Mihelic, N. Pavesic, Removing illumination artifacts from face images using the nuisance attribute projection, in *Proceedings of the IEEE Conference on Acoustics Speech Signal Processing* (2011), pp. 846–849
246. V.B. Subburaman, S. Marcel, Alternative search techniques for face detection using location estimation and binary features. Comput. Vis. Image Underst. **117**, 551–570 (2013)
247. J. Sun, N.N. Zheng, H.Y. Shum, Stereo matching using belief propagation. IEEE Trans. Pattern Anal. Mach. Intell. **25**, 787–800 (2003)
248. Synthetic Lane Data (2013), www.cvc.uab.es/adas/projects/lanemarkings/IJAT/videos.html
249. T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **1**, 116–132 (1985)
250. J. Tao, B.-S. Shin, R. Klette, Wrong roadway detection for multi-lane roads, in *Proceedings of the International Conference on Computer Analysis Images Patterns* (2013), pp. 50–58
251. A. Tawari, K.H. Chen, M.M. Trivedi, Where is the driver looking: analysis of head, eye and iris for robust gaze zone estimation, in *Proceedings of the IEEE Intelligent Transportation Systems Conference* (2014), pp. 988–994

252. A. Tawari, S. Sivaraman, M.M. Trivedi, T. Shanon, M. Tippelhofer, Looking-in and looking-out vision for urban intelligent assistance: estimation of driver attention and dynamic surround for safe merging and braking, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2014), pp. 115–120
253. Technology award for BLIS in 2006 (2006), www.roadsafetyawards.com/national/view.aspx?winnerid=107
254. L. Teijeiro-Mosquera, J.L. Alba-Castro, Recursive pose dependent AAM: application to drivers' monitoring, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2011), pp. 356–361
255. K. Teknomo, *k*-means clustering tutorials. Medicine **100**, 1–12 (2005)
256. C. Tomasi, T. Kanade, Detection and tracking of point features. Technical report CMU-CS-91-132. Int. J. Comput. Vis. (1991)
257. G. Toulminet, M. Bertozzi, S. Mousset, A. Bensrhair, A. Broggi, Vehicle detection by means of stereo vision-based obstacles features extraction and monocular pattern analysis. IEEE Trans. Image Process. **15**, 2364–2375 (2006)
258. C. Tran, A. Doshi, M.M. Trivedi, Modeling and prediction of driver behavior by foot gesture analysis. Comput. Vis. Image Underst. **116**, 435–445 (2012)
259. B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, Bundle adjustment – a modern synthesis, in *Proceedings of the Vision Algorithms Theory Practice* (2000), pp. 298–375
260. M.M. Trivedi, S.Y. Cheng, E. Childers, S. Krotosky, Occupant posture analysis with stereo and thermal infrared video: algorithms and experimental evaluation. IEEE Trans. Veh. Technol. **53**, 1698–1712 (2004)
261. M.M. Trivedi, T. Gandhi, J. McCall, Looking-in and looking-out of a vehicle: computer-vision-based enhanced vehicle safety. IEEE Trans. Intell. Transp. Syst. **8**, 108–120 (2007)
262. W. Tsao, A.J.T. Lee, Y. Liu, T. Chang, H. Lin, A data mining approach to face detection. Pattern Recognit. **43**, 1039–1049 (2010)
263. S. Tuohy, D. O'Cualain, E. Jones, M. Glavin, Distance determination for an automobile environment using inverse perspective mapping in OpenCV, in *Proceedings of the Signals and Systems Conference* (2010), pp. 100–105
264. TurboSquid (2013), www.turbosquid.com/Search/3D-Models/face
265. U.S. Department of Transportation, National Highway Traffic Safety Administration. The impact of driver inattention on near-crash/crash risk. DOT HS 810 594 (2006)
266. U.S. Department of Transportation, Research and Innovation Technology Administration, Vehicle-to-vehicle (V2V) communications for safety (2013), www.its.dot.gov/research/v2v.htm
267. M. Vargas, J.M. Milla, S.L. Toral, F. Barrero, An enhanced background estimation algorithm for vehicle detection in urban traffic scenes. IEEE Trans. Veh. Technol. **59**, 3694–3709 (2010)
268. N. Vinh Dinh, N. Thuy Tuong, N. Dung Duc, J. Jae Wook, Toward real-time vehicle detection using stereo vision and an evolutionary algorithm, in *Proceedings of the Vehicular Technology Conference* (2012), pp. 1–5
269. P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in *Proceedings of the IEEE Computer Vision Pattern Recognition*, vol. 1 (2001), pp. 511–518
270. P. Viola, M. Jones, Robust real-time face detection, in *IEEE International Conference on Computer Vision* (2001), pp. 1–8
271. P. Viola, M. Jones, Robust real-time face detection. Int. J. Comput. Vis. **57**, 137–154 (2004)
272. Virginia Tech Transportation Institute, 100-car naturalistic driving study fact sheet (2005)
273. Visage Tech (2014), www.visagetechnologies.com/
274. C. Wang, H. Zhang, M. Yang, X. Wang, L. Ye, C. Guo, Automatic parking based on a bird's eye view vision system. Adv. Mech. Eng. **6**, 1–13 (2014)
275. H. Wang, L.B. Zhou, Y. Ying, A novel approach for real time eye state detection in fatigue awareness system, in *Proceedings of Robotics Automation Mechatronics* (2010), pp. 528–532
276. R. Wang, L. Guo, B. Tong, L. Jin, Monitoring mouth movement for driver fatigue or distraction with one camera, in *Proceedings of IEEE International Conference on Intelligent Transportation Systems* (2004), pp. 314–319

277. W. Wang, Y. Mao, J. Jin, X. Wang, H. Guo, X. Ren, K. Ikeuchi, Driver's various information process and multi-ruled decision-making mechanism: a fundamental of intelligent driving shaping model. Int. J. Comput. Intell. Syst. **4**, 297–305 (2011)
278. X. Wang, M. Yang, S. Zhou, Y. Lin, Regionlets for generic object detection, in *Proceedings of the IEEE Confernce on Computer Vision* (2013), pp. 17–24
279. A. Wedel, H. Badino, C. Rabe, H. Loose, U. Franke, D. Cremers, B-spline modeling of road surfaces with an application to free-space estimation. IEEE Trans. Intell. Transp. Syst. **10**, 572–583 (2009)
280. A. Wedel, U. Franke, H. Badino, D. Cremers, B-spline modeling of road surfaces for freespace estimation, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2008), pp. 828–833
281. Y. Wei, U. Stilla, Comparison of two methods for vehicle extraction from airborne LiDAR data toward motion analysis. Geosci. Remote Sens. Lett. **8**, 607–611 (2011)
282. W. Wen, C. Xilin, Y. Lei, Detection of text on road signs from video. IEEE Trans. Intell. Transp. Syst. **6**, 378–390 (2005)
283. WHO 2013 Global status report on road safety (2013), www.who.int/violence_injury_prevention/road_safety_status/2013/en/
284. W.W. Wierwille, L.A. Ellsworth, Evaluation of driver drowsiness by trained raters. Accid. Anal. Prev. **26**, 571–581 (1994)
285. W.S. Wijesoma, K.R.S. Kodagoda, A.P. Balasuriya, Road-boundary detection and tracking using ladar sensing. IEEE Trans. Robot. Autom. **20**, 456–464 (2004)
286. Wikipedia, Automobile safety (2013), en.wikipedia.org/wiki/Automobile_safety
287. Wikipedia, *k*-means clustering (2016), en.wikipedia.org/wiki/K-means_clustering
288. Wikipedia, Radar gun (2013), en.wikipedia.org/wiki/Radar_speed_gun
289. P.I. Wilson, J. Fernandez, Facial feature detection using Haar classifiers. J. Comput. Sci. Coll. **21**, 127–133 (2006)
290. J. Wiśniewska, M. Rezaei, R. Klette, Robust eye gaze estimation, in *Proceedings of the IEEE International Conference on Computer Vision Graphics* (2014), pp. 636–644
291. World Health Organization (WHO), Road traffic injuries. Fact sheet No. 358 (2013)
292. World recognized Haar classifier contributors for face and eye detection (2014), code.opencv.org/projects/opencv/wiki/Contributors?version=3
293. D. Wu, Twelve considerations in choosing between Gaussian and trapezoidal membership functions in interval type-2 fuzzy logic controllers, in *Proceedings of the IEEE Conference on Fuzzy Systems* (2012), pp. 1–8
294. Z. Xiangxin, D. Ramanan, Face detection, pose estimation, and landmark localization in the wild, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2012), pp. 2879–2886
295. J. Xiao, S. Baker, I. Matthews, T. Kanade, Real-time combined 2D+3D active appearance models, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2004), pp. 535–542
296. J. Xiao, T. Fang, P. Zhao, M. Lhuilier, L. Quan, Image-based street-side city modeling. ACM Trans. Graph. **28**, article no.114 (2009)
297. R. Xiao, L. Zhu, H.J. Zhang, Boosting chain learning for object detection, in *Proceedings of the IEEE Conference on Computer Vision* (2003), pp. 709–715
298. H. Xinwen, L. Cheng-Lin, T. Tieniu, Learning boosted asymmetric classifiers for object detection, in *Proceedings of the IEEE Computer Vision Pattern Recognition* (2006), pp. 330–338
299. W. Xuezhi, Y. Huai, Y. Chunyang, S. Chunyan, D. Bobo, Z. Hong, Improved Haar wavelet feature extraction approaches for vehicle detection, in *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (2007), pp. 1050–1053
300. YALE Face Database (2013), vision.ucsd.edu/~iskwak/ExtYaleDatabase/Yale20Face20Database.htm
301. M.H. Yang, D. Kriegman, N. Ahuja, Detecting faces in images: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 34–58 (2002)

302. W. Yan-Wen, A. Xue-Yi, Face detection in color images using AdaBoost algorithm based on skin color information, in *Proceedings of the International Workshop Knowledge Discovery Data Mining* (2008), pp. 339–342
303. J.J. Yebes, L.M. Bergasa, R. Arroyo, A. Lazaro, Supervised learning and evaluation of KITTI's cars detector with DPM, in *Proceedings of the IEEE Intelligent Vehicle Symposium* (2014), pp. 768–773
304. M. Yu, G. Ma, 360° surround view system with parking guidance. SAE Int. J. Commer. Veh. **7**, 19–24 (2014)
305. L.A. Zadeh, Is there a need for fuzzy logic? Inf. Sci. **178**, 2751–2779 (2008)
306. O.R. Zaiane, Principals of knowledge discovery in data. University of Alberta (2016)
307. E. Zaytseva, S. Segui, J. Vitria, Sketchable histograms of oriented gradients for object detection, in *Proceedings of the Iberoamerican Congress Pattern Recognition* (2012), pp. 374–381
308. S. Zehang, G. Bebis, R. Miller, On-road vehicle detection: a review. IEEE Trans. Pattern Anal. Mach. Intell. **28**, 694–711 (2006)
309. Y. Zeng, R. Klette, Multi-run 3D streetside reconstruction from a vehicle, in *Proceedings of the International Conference on Computer Analysis Images Patterns*. LNCS 8047 (2013), pp. 580–588
310. C. Zhang, Z. Zhang, A survey of recent advances in face detection. Microsoft Research. Technical Report MSR-TR-2010-66 (2010)
311. C. Zhang, Z. Zhang, *Face Detection and Adatation*. Synthesis Lectures Computer Vision, vol. 2 (Morgan and Claypool Publishers, San Rafael, 2010)
312. X. Zhang, Y. Gao, Face recognition across pose: a review. Pattern Recognit. **42**, 2876–2896 (2009)
313. Z. Zhang, Y. Shan, Incremental motion estimation through local bundle adjustment Microsoft Research. Technical report MSR-TR-01-54 (2001)
314. K. Zhao, M. Meuter, C. Nunn, D. Muller, S. Muller-Schneiders, J. Pauli, A novel multi-lane detection and tracking system, in *Proceedings of the IEEE Intelligent Vehicles Symposium* (2012), pp. 1084–1089
315. N. Zhiheng, S. Shiguang, Y. Shengye, C. Xilin, G. Wen, 2D cascaded AdaBoost for eye localization, in *Proceedings of the International Conference on Pattern Recognition* (2006), pp. 1216–1219
316. Z. Zhu, Q. Ji, Robust real-time eye detection and tracking under variable lighting conditions and various face orientations. Comput. Vis. Image Underst. **98**, 124–154 (2005)
317. P. Zoratti, Automotive driver assistance systems: using the processing power of FPGAs. *EE Catalog*, vol. 1.0 (2011), pp. 1–8
318. K. ZuWhan, Robust lane detection and tracking in challenging scenarios. IEEE Trans. Intell. Transp. Syst. **9**, 16–26 (2008)

Index

M. Rezaei, R. Klette, *Computer Vision for Driver Assistance*, Computational Imaging and Vision 45, DOI 10.1007/978-3-319-50551-0