Multi-Cue Pedestrian Recognition

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Dipl.-Inf. Stefan Munder
aus Potsdam

Mannheim, 2007

Dekan:	Professor Dr. Matthias Krause, Universität Mannheim
Referent:	Professor Dr. Christoph Schnörr, Universität Mannheim
Korreferent:	Professor Dr. Dariu M. Gavrila, Universität Amsterdam

Tag der mündlichen Prüfung: 21. Juni 2007

Abstract

This thesis addresses the problem of detecting complex, deformable objects in an arbitrary, cluttered environment in sequences of video images. Often, no single best technique exists for such a challenging problem, as different approaches possess different characteristics with regard to detection accuracy, processing speed, or the kind of errors made. Therefore, multi-cue approaches are pursued in this thesis. By combining multiple detection methods, each utilizing a different aspect of the video images, we seek to gain detection accuracy, robustness, and computational efficiency.

The first part of this thesis deals with texture classification. In a comparative study, various combinations of feature extraction and classification methods, some of which novel, are examined with respect to classification performance and processing speed, and the relation to the training sample size is analyzed.

The integration of shape matching and texture classification is investigated. A pose-specific mixture-of-experts architecture is proposed, where shape matching yields a probabilistic assignment of a texture pattern to a set of distinct pose clusters, each handled by a specialized texture classifier, the local expert. The reduced appearance variability that each local expert needs to cope with leads to improved classification performance. A slight further performance gain could be achieved by shape normalization.

The second multi-cue approach deals with cascade systems that employ a sequence of fast-to-complex system modules in order to gain computational efficiency. Three optimization techniques are examined that adjust system parameters so as to optimize the three performance measures detection rate, false positive rate, and processing cost. A combined application of two techniques, a novel fast sequential optimization scheme based on ROC (receiver operating characteristics) frontier following, followed by an iterative gradient descent optimization method, is found to work best.

The third method investigated is a Bayesian combination of multiple visual cues. An integrated object detection and tracking framework based on particle filtering is presented. A novel object representation combines mixture models of shape and texture, the former based on a generative point distribution model, the latter on discriminative texture classifiers. The associated observation density function integrates the three visual cues shape, texture, and depth.

All methods are extensively evaluated on the problem of detecting pedestrians in urban environment from within a moving vehicle. Large data sets consisting of tens of thousands of video images have been recorded in order to obtain statistically meaningful results.

Zusammenfassung

Diese Dissertation befasst sich mit dem Problem, komplexe, deformierbare Objekte in beliebig strukturierter Umgebung aus Videobildern zu detektieren. Für solche herausfordernden Problemstellungen gibt es in der Regel nicht nur eine einzelne beste Lösung, sondern es existieren verschiedene Ansätze mit unterschiedlichen Charakteristiken bezüglich Detektionsgüte, Verarbeitungsgeschwindigkeit oder der Art der auftretenen Fehler. Daher werden in dieser Arbeit Multimerkmalsansätze verfolgt. Durch die Kombination mehrerer Methoden, die jeweils unterschiedliche Aspekte der Videobilder ausnutzen, wird eine erhöhte Detektionsgüte, Robustheit, und Recheneffizienz angestrebt.

Der erste Teil dieser Arbeit beschäftigt sich mit der Texturklassifikation. In einer Vergleichsstudie werden verschiedene Kombinationen von Merkmalsextraktion und Klassifikationsmethode, von denen einige neuartig sind, bezüglich Klassifikationsleistung und Rechenaufwand untersucht, sowie die Relation zur Größe der Lernstichprobe analysiert.

Die Integration von Formenabgleich (Shape Matching) und Texturklassifikation ist anschließend Gegenstand der Untersuchungen. Es wird eine posenspezifische Mixture-of-Experts-Architektur vorgeschlagen, bei der durch Formenabgleich eine probabilistische Zuordnung von Texturmustern zu bestimmten Posenklassen vorgenommen wird, die dann jeweils durch einen spezialisierten Texturklassifikator, dem sogenannten lokalen Experten, gehandhabt werden. Da jeder Experte nur noch mit einer reduzierten Erscheinungsvielfalt zurechtkommen muss, verbessert sich die Klassifikationsleistung. Eine weitere Leistungssteigerung konnte durch Formnormalisierung erreicht werden.

Der zweite hier untersuchte Multimerkmalsansatz setzt auf Kaskadensysteme, die zwecks Recheneffizienz aus einer Sequenz von zunächst einfachen und dann schrittweise immer komplexeren Systemmodulen bestehen. Drei Optimierungstechniken werden in dieser Arbeit analysiert, die Systemparameter so justieren, dass die drei Leistungsmaße Detektionsrate, Fehlalarmrate und Rechenaufwand optimiert werden. Eine kombinierte Anwendung zweier Techniken, ein neuartiges, schnelles sequentielles Optimierungsschema basierend auf der Verfolgung von ROC-Fronten gefolgt von einer iterativen Gradientenabstiegsmethode, stellte sich als am erfolgreichsten heraus.

Der dritte untersuchte Ansatz ist die Bayes'sche Kombination mehrerer visueller Merkmale. Vorgestellt wird ein integriertes Objektdetektions- und Tracking-Framework basierend auf Partikelfiltern. Eine neuartige Objektrepräsentation kombiniert Mischmodelle für Form und Textur, ersteres basierend auf einem generativen Punktverteilungsmodell, zweiteres basierend auf diskriminativen Texturklassifikatoren. Die zugehörige Beobachtungsdichtefunktion integriert die drei visuellen Merkmale Form, Textur und Tiefe.

Alle Methoden werden auf das Problem, Fußgänger in innerstädtischer Umgebung aus einem fahrenden Fahrzeug heraus zu detektieren, angewandt und ausführlich bewertet. Um statistisch aussagekräftige Ergebnisse zu erhalten, werden große Datensätze bestehend aus zehntausenden Videobildern eingesetzt.

Acknowledgments

First of all, I would like to express my gratitude to Professor Dariu M. Gavrila who gave me the opportunity to work on the exciting field of pedestrian recognition, who taught me how to conduct scientific research, and who guided me through the process of writing my dissertation. Without his continued input of ideas, inspiring discussions, and technical support, this work would not have been possible.

I would like to thank Professor Christoph Schnörr for supervising my dissertation, for patiently supporting my work and for providing me with the necessary scientific background.

I am grateful to all colleagues, fellow students, and friends at DaimlerChrysler Research for providing a friendly and inspiring atmosphere. In particular, many thanks go to Jan Giebel for his introductory support. I have benefitted a lot from his preceding work. Secondly, I want to thank Markus Enzweiler for our close and fruitful collaboration, for his support on data preparation and software implementation, and for proofreading this thesis. I am also grateful to Ulrich Kreßel for providing all the computing equipment to conduct my work.

Finally, I wish to thank my parents who always supported me in what I was doing.

Contents

Abstract . . . iii
Zusammenfassung . . . iv
Acknowledgments . . . v

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Thesis Outline . . . 2
1.3 Thesis Contribution . . . 3

2 Related Work **5**
2.1 ROI Generation . . . 5
2.2 Generative vs. Discriminative Pedestrian Models . . . 6
2.3 Pedestrian Classification . . . 7
2.4 Pedestrian Tracking . . . 10
2.5 System Optimization . . . 11

3 Texture Classification **15**
3.1 Feature Extraction . . . 16
3.1.1 PCA Coefficients . . . 16
3.1.2 Haar Wavelets . . . 17
3.1.3 Local Receptive Fields . . . 17
3.2 Classification Methods . . . 18
3.3 Methods for Increasing the Training Sample Size . . . 19
3.3.1 Bootstrapping . . . 19
3.3.2 Cascade . . . 19
3.3.3 Boosted Cascade of Haar-like Features . . . 19
3.4 Benchmark Data Set . . . 20
3.4.1 Data Sets . . . 20
3.4.2 Test Procedure . . . 21
3.5 Experimental Results . . . 22
3.5.1 Combinations of Feature Extraction and Classification Methods . . . 22
3.5.2 Increasing the Training Sample Size . . . 25
3.5.3 Summary . . . 28

4 Integration of Shape and Texture **29**
4.1 Shape Representation . . . 31
4.1.1 Exemplar-based Shape Representation . . . 32
4.1.2 Multi Point Distribution Model (MPDM) . . . 32
4.2 Pose-Specific Mixture of Experts . . . 34
4.2.1 Shape-Based Gating Function . . . 35
4.2.2 Expert Training and Combination . . . 36
4.2.3 Implementation Details . . . 36
4.3 Shape Normalization . . . 37
4.4 Experiments . . . 39
4.4.1 Data Sets . . . 40
4.4.2 Pose Clustering . . . 41
4.4.3 Cluster Association . . . 43
4.4.4 Validation of Design Choices . . . 45
4.4.5 Shape Normalization . . . 45
4.4.6 Results . . . 47

5 Cascade Optimization **49**
5.1 Problem Formulation . . . 50
5.2 Sequential ROC Optimization . . . 53
5.2.1 Sequential Optimization Algorithm . . . 53
5.2.2 Discussion . . . 55
5.3 Generic Optimization Techniques . . . 57
5.4 Independent Optimization . . . 59
5.5 Cascade Classification Experiments . . . 60
5.5.1 Synthetic Data Sets . . . 60
5.5.2 Real Example . . . 62
5.6 Pedestrian Recognition Application . . . 65
5.6.1 PROTECTOR System Modules . . . 65
5.6.2 Test Methodology and Data Sets . . . 69
5.6.3 Parameter Optimization Experiments . . . 69
5.6.4 System Evaluation . . . 71

6 Bayesian Detection and Tracking **75**
6.1 Particle Filtering Framework . . . 76
6.2 Multi-Cue Object Representation . . . 78
6.2.1 Shape Cue . . . 79
6.2.2 Texture Cue . . . 80
6.2.3 Depth Cue . . . 81
6.2.4 Cue Integration . . . 81
6.3 Application System . . . 82
6.3.1 Target Object Detector . . . 83

6.3.2 Proposal Density 83
6.3.3 Track Initialization and Termination 84
6.4 Experiments 84
6.4.1 Model Generation 84
6.4.2 System Evaluation 87

7 Conclusion **91**
7.1 Summary 91
7.2 Future Work 93

Bibliography **95**

1 Introduction

1.1 Motivation

Detecting a class of objects in video images is a fundamental topic in computer vision research. At the core, we are faced with the pattern recognition problem of discriminating patterns of the target class from the vast space of "everything else". In this thesis, we are specifically interested in those cases, where (a) the target object is deformable and of variable appearance, (b) no prior object model (such as a CAD drawing) is available, but where a representation of the target class needs to be learned from training data, and where (c) the target object is located in a dynamic, cluttered environment, so that simple segmentation techniques such as background subtraction are not applicable. The list of typical properties that such an object detector is required to possess include detection accuracy, i.e. a high detection rate at low false positives, robustness to environmental conditions such as global illumination, and computational efficiency.

In general, there is no single best technique that serves all of these purposes, since different approaches exhibit different characteristics. Therefore, multi-cue approaches that combine multiple detection methods, each utilizing a different aspect of the input image, are pursued in this thesis. Such combinations of multiple visual cues have the potential of higher accuracy and robustness than single cue approaches, since errors in one cue, caused for example by a certain background or lighting condition, may be compensated for by the others.

A particularly interesting combination is that of shape and texture, as these two visual cues represent two fairly independent constituents of object appearance. Shape describes the pose and articulation of an object projected to the image plane, while texture is determined by the object's surface structure and the lighting conditions. Utilizing shape for object detection is attractive because its observation is largely independent of unwanted appearance variations due to lighting and background. But suppressed texture details might make the difference between target and non-target objects. On the other hand, utilizing the richer texture cue requires more complex representations. One topic of this thesis is, hence, to investigate how prior shape knowledge, given by a shape matching step, can be utilized to simplify the texture classification problem.

Computational efficiency is an important aspect, as object detectors are often required to run in real-time. In a multi-cue system consisting of multiple system modules, computational efficiency is attained by not running all modules in parallel but sequentially. Fast modules are applied first to prune out the search space, so that the more powerful

but computationally costly modules can be applied to the remaining candidates only. This approach is referred to as a cascade architecture. Whereas traditional training algorithms deal with the optimization of single modules, the problem of how to adjust each module so as to optimize the overall system performance is addressed in this thesis.

Summarizing, the goal of this thesis is to gain object detection accuracy, robustness, and computational efficiency by the integration of multiple visual cues.

The target application of this work is the detection of pedestrians in urban environment from a moving vehicle, which serves as the testbed for the experiments throughout this thesis. This application combines the difficulties of a complex appearance variation of the target object due to varying clothing, pose, and articulation, varying lighting conditions, and a changeable, cluttered background. But the techniques presented here apply to the detection of humans in other scenarios as well, e.g. for in surveillance applications, advanced human-machine interfaces, or sports activity analysis, and could conceivably also be applied to the recognition of other objects such as faces, hand gesture, or vehicles viewed from an arbitrary angle.

1.2 Thesis Outline

This thesis first examines methods on pedestrian classification (Chapter 3). A selection of feature extraction and pattern classification methods is made that covers the majority of previous techniques. Experiments are made on a common large public data set to yield useful insight into the problem at hand, and to analyze the pros and cons of the underlying methodical components.

The integration of the two particular visual cues shape and texture is analyzed in Chapter 4, with the goal of simplifying the texture classification task by utilizing explicit prior shape knowledge. A shape matching technique is first applied to an input image to provide information about object pose and contour. The outcome triggers a weighted combination of pose-specific texture classifiers, after optionally shape-normalizing the input pattern.

In Chapter 5, the issue of computational efficiency is addressed by considering a cascade of system modules, each utilizing a different visual cue. The problem to be solved is how to adjust parameters of each system module so as to optimize overall system performance, taking detection rate, false positive rate, and processing costs into account. Three optimization techniques are discussed and evaluated experimentally based on synthetic and real-world data.

Multi-cue integration based on a Bayesian framework for object tracking using particle filtering is examined in Chapter 6. A joint object representation and associated observation density function integrating the three visual cues shape, texture, and depth are proposed. The decision about the presence of a target object is based on a sequence of observations instead of a single one, which is shown to improve detection performance.

1.3 Thesis Contribution

The original contribution of this thesis is as follows.

- A thorough comparative study of texture-based pedestrian classification techniques is presented. Different combinations of feature extraction and pattern classification methods, some of which novel, are examined experimentally, and the relation between classification performance and training sample size is studied. Statistically meaningful results are obtained by analyzing performance mean and variance over varying training and test data sets. (Chapter 3)

- A two-step approach for integrating explicit prior shape knowledge into texture classification is presented. First, the shape matching outcome activates a weighted combination of texture classifiers, each attuned to a particular body pose. Secondly, shape normalization eliminates shape variation within each pose cluster from the input patterns. (Chapter 4)

- The optimization of a complex cascade detection system is addressed. Three optimization techniques are discussed, including a new sequential optimization scheme that makes use of the cascade architecture to successively compute optimal parameters combinations. Novel is the consideration of the three optimization objectives detection rate, false positive rate, and processing cost, as well as a thorough experimental analysis of all methods based on synthetic and real-world data. (Chapter 5)

- The joint shape-texture object representation is integrated into a Bayesian tracking framework based on particle filtering. Combining a generative shape model for accurate shape matching and a discriminative texture classifier is shown to enable a tracker to simultaneously infer object class and configuration. (Chapter 6)

- A system for real-time pedestrian detection from a moving vehicle is presented with leading edge performance. Extensive experiments were performed both offline, using large data sets recorded in urban environment under different lighting conditions, capture times and locations, and online, integrated in a demonstrator vehicle.

The work on this thesis has lead to the following publications:

- D. M. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection: the PROTECTOR+ system. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 13–18, Parma, Italy, 2004.

- S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, 2006.

- D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73(1):41–59, 2007.
- S. Munder, C. Schnörr, and D. M. Gavrila. Pedestrian detection and tracking using a mixture of view-based shape-texture models. Submitted to *IEEE Transactions on Intelligent Transportation Systems*, 2007.

2 Related Work

The complexity of pedestrian appearance, combined with a cluttered background, has lead most authors to pursue learning-based detection approaches where a representation of the pedestrian class is learned from a set of example images instead of hand-crafting pedestrian models. Such representations either involve explicit, generative models that describe how pedestrians appear in an image given some configuration parameters (see Section 2.2), or involve discriminative models that describe the decision boundary between pedestrian and non-pedestrian images (see Section 2.3). In a pedestrian detection system on board a vehicle, application of these models is typically restricted to certain regions of interest (ROIs) in order to enhance processing speed. Standard background subtraction, as frequently used in surveillance applications, is unsuitable because of the moving camera. A number of viable alternatives for ROI generation exist as summarized in Section 2.1.

A separate field of research is that of pedestrian tracking, which basically involves inserting a pedestrian model into a framework of generic object tracking. Techniques previously used range from very simple models (e.g. a cylinder moving with constant velocity) attached to a pedestrian detection module to sophisticated applications that integrate detection and tracking of multiple pedestrians using complex appearance models. An overview is given in Section 2.4.

The combination of multiple system modules, e.g., modules for ROI generation, classification, and tracking, requires some adjustments to be made on each module so as to optimize overall system performance. A large body of literature exists on the generic optimization of non-convex, non-smooth objective functions, but due to the inherent complexity, serious attempts so simultaneously optimize all components of object recognition systems are surprisingly rare, see Section 2.5.

2.1 ROI Generation

Powerful but computationally expensive classifiers require a pre-processing step to reduce the number of image locations at which the classifier is applied, in order to reach an acceptable processing speed.

This can be done by detecting objects that stick out of the ground plane. One way to detect those objects is by stereo vision. For example, Zhao and Thorpe [105] obtain potential pedestrian regions by clustering in disparity space. A refinement of the foreground region is obtained by iteratively applying disparity pixel clustering and object

classification. Broggi et al.[10] and Grubb et al.[39] consider the x- and y-projections of the disparity space following the *V-disparity* technique [57]. Alternatively, optical flow techniques were used if only monocular images are available. This approach typically involves detecting independently moving objects by comparing the observed optical flow field with the assumed camera motion [21, 75, 92]. Some authors combined stereo vision and optical flow techniques to achieve more accurate object segmentation [26].

Within a tracking framework, possible positions of once detected pedestrians in subsequent video frames can be restricted by the dynamical models of pedestrian and camera motion.

2.2 Generative vs. Discriminative Pedestrian Models

For representing a class of objects, two types of visual models can roughly be distinguished: *Generative* (explicit) models, which describe how objects look like and often allow to synthesize new object views, and *discriminative* models that describe the decision boundary between objects of the target class and non-target objects.

Regarding generative pedestrian models, shape models, either in 2D or 3D, are attractive because appearance variations that arise from varying lighting or clothing are eliminated, and because efficient matching techniques exist. While building 3D models requires elaborate equipment or user interaction [31], 2D models can be directly learned from examples images. For instance, the manifold of 2D outer contours of pedestrians has been represented by a set of example shapes [37, 93, 88]. Such models are particularly suited for object detection due to their specificity, i.e., no false object shapes are included in the model, but need to handle a large number of examples. Efficient coarse-to-fine matching strategies are typically employed, often in combination with distance-transformed images [37, 93, 87]. Gaps in the shape model can be filled by using parametric representations of deformable contours [14, 41, 35, 5]. But since simple models (e.g. a single Gaussian [14]) are too unspecific for the given problem, more complex models (e.g. mixture of Gaussians [41]) require large training sets as well. A possible drawback of parametric shape models is their need for usually iterative parameter estimation procedures for model matching as, for example, the Dynamic Programming-based, snake-like *Active Contours* technique by Cootes et al.[14], so that these models are better suited for tracking applications where matching starts with a good initial shape hypothesis [38]. Statistical field models have recently been proposed for representing object shape. For example, a two-layer field representation has been presented [102] consisting of a Markov field in the hidden layer that captures the shape prior, and an observation layer that represents the conditional likelihood of associated image observations. This approach is particularly suited for handling partial occlusions, since shapes are represented by a locally connected field model instead of a centralized shape vector.

Irrespective of the particular representation, a weak point of purely contour-based

approaches is their susceptibility to background clutter, i.e., random background structure may lead to similar shape observations as the object class. Besides probabilistic approaches for handling this uncertainty, authors have built richer representations by incorporating texture models. Compound linear models of shape and texture have been proposed by multiple authors [15, 48, 24], see [62] for an overview. Linear texture models are typically built by warping the example images to a common reference shape using point correspondences given by the shape model, and by using PCA (principal component analysis) to obtain a compact representation. However, the associated assumption of a normal (Gaussian) distribution of the combined shape-texture representation is too rough and does not adequately represent the pedestrian class. Alternatives are available, such as kernel-based densities or mixtures of Gaussians, see [16] for a discussion. A second drawback is the fitting of these models to an input image, which requires the combined estimation of shape and texture parameters by means of iterative, gradient descent-like methods, and is, hence, time consuming. As a way out, separate models of shape and texture have been built, and their observations have been combined in a joint observation density function (e.g. [85, 45]).

Due to these difficulties, we refrain from using generative texture models in this work. Instead, efficient and effective pattern classification techniques are employed that directly model the decision boundary between pedestrian and non-pedestrian images, see next section. Motion is another visual cue that has been used to represent pedestrians. In particular, models of the typical gait pattern have been built [23, 106]. However, these are restricted to pedestrians walking parallel to the image plane, and are therefore not considered in this thesis.

Increased accuracy and robustness has been reported by the combination of multiple complementary visual cues, where failures of one cue are compensated by the others [84, 86, 85], and by using component-based approaches [84, 104, 78, 101] that decompose the complex object appearance into simpler models of body parts. Another approach for reducing the high complexity of pedestrian appearance involves pose clustering. For example, Zhang et al.[104] and Wu & Nevatia [101] manually categorize training examples by viewing angle (left, right, frontal, and back views), and build separate object models for each viewing direction. Heap and Hogg [41] proposed a "Hierarchical PDM" approach to 2D shape modeling, where a set of locally linear shape models is found by a k-means clustering of the training data. An adaptation of this approach in [38] has been used for our shape model in Section 4.1.2, which also forms the basis of the proposed mixture of pose-specific texture classifiers in Section 4.2.

2.3 Pedestrian Classification

Insightful surveys on statistical pattern recognition were given by Jain et al.[47] and Duda et al.[20]. With regard to pedestrian classification, many different combinations of feature extraction and actual classification techniques were applied in the literature, see

Table 2.1 for an overview. One line of research involves neural networks. For example, Wöhler and Anlauf [100] train a feed-forward neural network with local receptive fields directly on (size normalized) pedestrian images. Zhao and Thorpe [105] apply a fully connected feed-forward neural network to high-pass filtered images. Another line of research has involved over-complete sets of Haar wavelet features in combination with a Support Vector Machine (SVM). This approach was pioneered by Papageorgiou and Poggio [72] and later adapted by Elzein et al.[21] and others.

Instead of shifting all the work to a single powerful, hence computationally expensive classifier, Viola et al.[98] proposed an efficient detector cascade, where simpler detectors are placed earlier in the cascade and more complex later. At each stage, AdaBoost [27] adds simple appearance filters (similar to Haar wavelets) to the classifier until a user-defined performance target is reached. Originally developed for the face detection domain, this approach was later applied to pedestrian detection [99].

Apart from reducing processing costs, techniques for combining multiple classifiers have also been used to simplify the classification problem. One such line of research pursues multi-class approaches where the target class is separated into a number of simpler sub-classes or clusters. Multiple classifiers are generated, each one to distinguish only one sub-class from the non-target class. Online, all classifiers are applied to a test pattern in parallel, and the final decision is given by the sub-class of the highest classification score. Examples include the work of Grubb et al.[39] who employed two SVM classifiers, one for front/back views and one for side views, or Nakajima et al.[67], who trained SVMs for four different poses. Shimizu and Poggio [83] estimate the walking direction of pedestrians by employing 16 competing classifiers, each trained for a certain angular range. A different combination strategy was pursued by Shashua et al.[82]. After manually dividing the training set into 9 clusters of particular pose and illumination and creating relatively simple classifiers on each cluster, AdaBoost combines the multiple discriminant values into a final decision.

Common to these approaches is the fixed combination rule where the final classification result is determined exclusively by the outputs of the component classifiers. In contrast, the mixture-of-experts architecture, first introduced by Hampshire and Waibel [40] and later made popular by Jacobs et al.[46, 49], employs a dynamic combination rule by means of a dedicated gating network that assigns an input pattern to one of the local experts. This gating network is generated either prior to [40], in parallel with [46, 49], or subsequent to [55] to the training of the local experts. Though the mixture-of-experts approach proved beneficial for many applications (e.g., speech recognition), we are not aware of previous applications in the pedestrian classification domain.

An alternate way of reducing the complexity of pedestrian appearances are component-based approaches. Shashua et al.[82], for instance, extract a feature vector from each of 9 fixed sub-regions. Other approaches try to directly identify certain body parts. Mohan et al.[64], for example, extend the work of [72] to four component classifiers for detecting heads, legs, and left/right arms separately. Individual results are combined by a second classifier, after ensuring proper geometrical constraints. Mikolajczyk et al.[63]

Table 2.1: Overview of previous work on pedestrian classification.

Authors	Features	Classifier	Training Set (ped. / non-ped.)	Test Set (ped. / non-ped.)	Det. Rate / False Pos. Rate
Zhao & Thorpe, 2000 [105]	edge pixels	neural network	1012 / 4306 ex.	8400 ex. (total)	85.2% / 3.1%
Wöhler & Anlauf, 1999 [100]	local receptive fields	neural network (SVM)	3926 / 4426 ex.	1000 / 1337 ex.	85.8% / 1.6%
Papageorgiou & Poggio, 2000 [72]	Haar wavelets	SVM (quadratic)	924 ex. / -	123 ex. / 50 img.	70% / 0.15, 70% / 3
Mohan et al., 2001 [64]	Haar wavelets	SVM	866 / 9315 ex.	123 / 796,904 ex.	90% / 0.08
Elzein et al., 2003 [21]	Haar wavelets	SVM	600 ex. / -	39 ex. / -	69% / 1.06
Viola, Jones & Snow, 2005 [99]	Haar wavelets	AdaBoost Cascade	2250 ex. / -	2 × 2000 img.	80% / 0.5
Shashua et al., 2004 [82]	edge orientation hist.	Linear weak learner, AdaBoost	≈ 27,100 ex. each	≈ 7,600 ex. each	93.5% / 8%
Mikolajczyk et al., 2004 [63]	edge directions	AdaBoost cascade	200...300 / 100K ex.	400 ex. / 200 img.	87% / 0.55
Grubb et al., 2004 [39]	edge	SVM			
Gavrila & Giebel, 2002 [33]	edge	chamfer matching	2661 ex. / 0	2101 ex. / -	80% / 5
Szarvas et al., 2005 [91]	local receptive fields	neural network, SVM	3,699 / 30K ex.	1,655 / 30K ex.	90% / 1.2%
Bertozzi et al., 2004 [6]	symmetry	thresholding	- / -	1897 ex. / -	83% / 0.46
Dalal & Triggs, 2005 [18]	histogram of oriented gradients	SVM (linear or Gaussian K.)	2478 ex. / 1218 img.	1132 ex. / 453 img.	89% / 0.01%
Leibe et al., 2005 [58]	Implicit Shape Model + chamfer matching + MDL verification		210...420 ex. / -	595 ex. / 209 img.	71.3% / 0.82

"ex." denotes the number of example windows in a data set, "img." denotes the number of video images from which non-pedestrian examples have been extracted. The corresponding false positive rate is either given as the percentage of non-pedestrian examples incorrectly classified, or as the absolute (average) number of false positives per image.

train a cascade of AdaBoost classifiers, similar to [99], for each of 7 body parts, and probabilistically assemble the component results considering position and classification confidence.

Apart from differences in feature extraction and classification methods, we can characterize previous work by the amount and type of image data utilized. For instance, Papageorgiou and Poggio [72] used 924 frontal and rear views of pedestrians for training (plus several thousands of negative examples), and demonstrated performance on 123 pedestrian test examples. Viola et al.[99] deployed a training set of 2250 pedestrian examples. The data sets of Shashua et al.[82] consists of 54,282 training and 15,244 test examples, about equally distributed between positive and negative examples. "Bootstrapping" [90] (cf. Section 3.3.1) was used by some authors to increase the number of negative training examples, e.g. [105].

There are some striking differences in the reported classification performance on the above data sets. The variation in the number of false classifications at a particular correct classification rate can exceed one order of magnitude across multiple sequences of the same study [99], and can run as high as several orders of magnitude when considering multiple studies (e.g. [105, 82] vs. [72]). These large performance variations are mainly the result of the (limited) size of the data sets used and their composition, in particular with respect to the negative examples. Data sets which draw the negative examples randomly from images containing large uniform image regions (e.g. sky, pavement) lead to much better classification performance than data sets where the negative examples are generated by some pre-filtering method and contain pedestrian look-a-like vertical structures.

This lack of comparability has motivated our experimental study in Chapter 3. By evaluating different combinations of feature extraction and classification methods of a common large, public data set, our aim is to establish a clearer picture of what underlying methodical components are particularly worthwhile.

2.4 Pedestrian Tracking

See Table 2.2 for an overview of previous work on pedestrian tracking. Particle filtering has evolved as the standard tool for pedestrian tracking because of its ability to estimate complex multi-modal posterior pdfs that arise in cluttered environments. Following the seminal work of Isard and Blake [43] who re-introduced particle filtering to computer vision, many extensions have been proposed regarding mixed discrete/continuous state spaces [41], improved sampling strategies [44, 19], and the integration of multiple visual cues [44, 61, 86]. An excellent overview of particle filter variants is given in [1].

In many early papers on pedestrian tracking, track initialization is performed manually in an initial video frame or given by some known prior distribution. This is impractical in most applications, where typically pedestrians randomly enter or leave the field of view. From a theoretical point of view, the straightforward solution is to construct a joint

state space of variable dimension and to infer the number of objects in parallel with each object's configuration. However, means for reducing the computational burden of the increased state dimension need to be found, such as the grid-based observation model with pre-computed likelihood values in [45], or the use of the Metropolis-Hastings algorithm for sampling in [51].

If complex object representations or complex observation models are involved, authors have generally refrained from joint state spaces but rather ran multiple (single target) tracker instances in parallel. Some heuristics are then typically used to handle track initialization and termination, and to implement target interactions. For example, Kang and Kim [50] perform object detection by a dedicated particle filter with uniform prior distribution, and proposed a competition rule to handle multiple proximate tracks. Other authors [44, 70] employ an independent object detector process for track initialization. Its output is also incorporated into the proposal distribution of the particle filter, in order to "guide" particle sampling towards the more likely image regions, which speeds up computation and increases robustness to gross object motions. Although good practical results were obtained with these multi-tracker approaches, object detection performance is clearly limited by the initialization heuristic, since the decision about the presence of a target object is solely made by the detector process. In contrast, joint particle filters make inference about the presence (or the number) of target objects from a sequence of observations in sound probabilistic (Bayesian) manner, but suffer from their exponential complexity.

The tracking framework employed in Chapter 6 represents a combination of both approaches. While we follow a multi-target approach to infer object class by the particle filter, the number of targets per particle filter is limited to one in order to cope with our high dimensional state space. Multiple particle filters are used to track multiple targets.

2.5 System Optimization

A large body of literature exists for the optimization of non-convex, non-smooth objective functions. But most such approaches are impractical for the given problem because of its lack of a closed-form mathematical representation of the overall system performance with respect to parameter values of individual system components. Empirical system performance is measured on large sample sets, which is relatively costly. Recently, a particular constrained optimization technique has been proposed by Vanden Berghen [3] that suits the given problem. This method is employed in Section 5.3; a short description is given there. Alternatively, authors have either tried to model system behavior by functions for which established analysis techniques exist, e.g. Bayesian networks [80], or developed specialized solutions for the particular problem at hand. Within this work, we aim to exploit the cascade coupling of system modules.

Cascaded classifiers [99] have recently received increasing interest due to their computational efficiency, and a number of publications addressed their optimization. For

Table 2.2: Overview of previous work on the visual tracking of pedestrians (humans)

Authors	Object Model	Visual Cues				Tracking Approach
		Shape	Texture	Motion	Others?	
Deutscher et al., 2000, [19]	3D assembly of cylinders	edge pixels			BG subtr.	"Annealed" PF
MacCormick and Blake 2000, [60]	2D shape	edge pixels				PF
Isard and MacCormick, 2001 [45]	3D generalized cylinder	Mexican hat filter	color			PF
Soto and Khosla, 2001 [85]	2D appearance	edge pixels	color histogram		stereo	PF
Toyama and Blake, 2002 [93]	2D shape exemplars	edge pixels				PF
Fablet and Black, 2002 [23]	2D appearance			optical flow		PF
Sidenbladh and Black, 2003 [84]	3D assembly of truncated cones	edge, ridge		intensity diff.		PF
Spengler and Schiele, 2003 [86]	2D appearance		skin color		BG subtr.	Kalman or PF
Zhao and Nevatia, 2004 [106]	3D shape and locomotion			optical flow	BG subtr.	Kalman
Ning et al., 2004 [68]	3D shape and motion	edge pixels			BG subtr.	PF
Roth et al., 2004 [79]	2D appearance	gradient pixels				PF
Okuma et al., 2004 [70]	2D appearance		color histogram			"Mixture" PF
Kang and Kim, 2005 [50]	2D shape (SOM)	edge pixels				PF
Ramanan et al., 2005 [78]	component-based 2D shape	edge pixels	color classifier			MAP search by DP
Wu and Yu, 2006 [102]	2D shape (Markov field)	edge pixels				PF
Wu and Nevatia, 2006 [101]	2D component appearance	"Edgelets"				data association or meanshift
This thesis, Chapter 5	2D shape exemplar and texture mixture	edge pixels	texture classifier		stereo	data association and α-β filter
This thesis, Chapter 6	mixture of 2D shape (PDM) and texture	edge pixels	texture classifier		stereo	PF

BG subtr.: background subtraction; PF: particle filtering; PDM: Point Distribution Models [14], SOM: Self-Organizing Map, DP: Dynamic Programming. Object models denoted "appearance" use learned filter responses of the used visual cues to represent the target class.

example, Sun *et al.* [89] and Luo [59] observed that the overall cascade performance is optimal if the slope of the log-scale ROC curve is equal for all nodes, given that the individual cascade nodes are statistically independent. No such assumption is made by Huo and Chen [42], who recently proposed a ROC "frontier-following" heuristic to successively adjust the thresholds of a classifier cascade. The idea of analyzing the optimal front of ROC points was first utilized by Provost and Fawcett [77] in the context of classifier comparison. They showed that, for any misclassification costs, the optimal classifiers are located on the ROC convex hull. Here, we extend both ideas [42, 77] by developing a technique to sequentially optimize a cascade of complex system modules, each controlled by a number of parameters, with respect to ROC performance and processing time (see Section 5.2).

3 Texture Classification

The capability of discriminating image patterns of the target object class from those of the non-target (background) class is one of the key components of every object recognition system. The pedestrian recognition application is particularly challenging because of the great variability of both the target and the non-target class. Advances in machine learning theory coupled with improvement in computer technology (processing speed, storage) increasingly favor techniques that do not rely on manually crafted models, but which, instead, use learning approaches with the corresponding large training sets to distinguish whether an image region contains an object or not.

Many interesting pedestrian classification approaches have been proposed in the literature; an overview was given in Section 2.3. However, the amount of training and test data used in these publications, and their distribution in terms of capture times and locations, differ substantially. This prohibits a meaningful quantitative performance comparison and offers little insight in the relative merits of the underlying methodical components. This chapter provides a thorough experimental study of pedestrian classification techniques on a large, common data set.

The actual pattern classification step is typically preceded by a feature extraction step, which serves two purposes. The first aim is to suppress irrelevant components of texture variation such as noise or global illumination variations that do not possess discrimination information. Secondly, the transformation into a higher dimensional feature space allows the use of simple classification rules. In this chapter, multiple combinations of feature extraction and classification techniques, some of which novel, are examined empirically.

Another important factor in pattern classification is the size of the training set. While

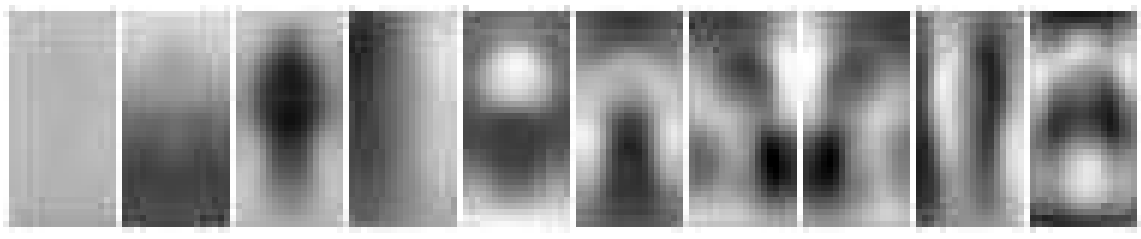


Figure 3.1: Illustrating example of principle components obtained on the training data set introduced in Section 3.4.1. First 10 principal components (according to maximum eigenvalue) are shown.

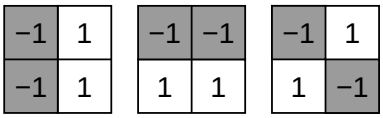


Figure 3.2: Haar wavelets of three different orientations – vertical, horizontal, and diagonal – as utilized by Papageorgiou and Poggio [72]

it is well-known that classification performance scales with the training sample size, theoretical studies exist for generalized or simplified problems only, e.g. [96], and are often not directly applicable to real-world problems. Here, we empirically study the correlation of classification performance with training sample size and investigate two techniques for the automatic generation of new training examples.

3.1 Feature Extraction

Based on the variety of techniques listed in Section 2.3, this section provides a description of the feature extraction techniques selected for experimental evaluation. We distinguish global and local features and further differentiate between adaptive and non-adaptive features among the latter. These categories are exemplified by PCA coefficients, local receptive fields (LRF), and Haar wavelets below. Associated parameters are subject to optimization via cross-validation on the training set (see Section 3.4.2).

3.1.1 PCA Coefficients

The probably best known (linear) feature extraction method is principal component analysis (PCA) [47]. It effectively reduces dimensionality by identifying the most expressive features, i.e., the eigenvectors with the largest eigenvalues, while those with small eigenvalues are assumed to contain noise and are cut off accordingly. PCA coefficients can be regarded as global features as each coefficient describes a certain property of the full input pattern, whereas local details are smoothed out by the dimensionality reduction (see Figure 3.1).

Two approaches exist in the literature: The “Eigenobjects” (most notably “Eigenfaces”) approach computes principal components on the target class only in order to extract typical features of the target objects. Alternatively, the principal components can be computed from all training examples to obtain a compact representation of the overall appearance variability. In preliminary experiments, we found both approaches to perform equally well, so that the second one is used here.

The number of principal components to remain is typically user-defined. We consider values that capture 80, 90, 95, or 100 percent of the variance during parameter optimization.

3.1.2 Haar Wavelets

The most popular features for pedestrian classification found in the literature are Haar wavelets, or extensions thereof, e.g., [72, 99]. Their use is motivated by the fact that they encode local image features, i.e., intensity differences, at multiple scales, thus allowing for a balance between compactness and expressivity.

We adopt the over-complete dictionary of Haar wavelets by Papageorgiou and Poggio [72], where "over-completeness" arises from wavelets of three different orientations (see Figure 3.2) shifted by 1/4 of the size of the support of each wavelet in both directions. Domain knowledge about the target class is incorporated by using only two medium scales of wavelets. Wavelets of the finest scale are assumed to represent noise and are, hence, discarded, as well as very coarse scale wavelets which have support as large as the object itself. Given our input images of size 18×36, we selected wavelets of scales 4×4 and 8×8, from which we obtained 15×33 and 6×15 features, respectively, for each orientation; hence, a total of 1,755 features. Furthermore, the signs of the coefficients, i.e., of the intensity differences, are considered irrelevant: only their magnitude is encoded in the feature vectors.

In addition, we pursue the approach by Viola and Jones [99], who build a cascade of AdaBoost classifiers, based on a much greater dictionary of features (see Section 3.3.3).

3.1.3 Local Receptive Fields

Instead of manually crafting a set of features, multilayer perceptrons provide an adaptive approach for feature extraction by means of their hidden layer, so that the features are tuned to the data during training [47]. Feed-forward neural networks with local receptive fields (NN/LRF), introduced by Fukushima et al. [28] and later applied to pedestrian classification by Wöhler and Anlauf [100], are a particularly attractive approach for classifying 2D images. In contrast to standard multilayer perceptrons, neurons in the hidden layer are only connected to a restricted local region of the input image, referred to as their local receptive fields (see Figure 3.3). The hidden layer is divided into a number of branches, with all neurons within one branch sharing the same set of weights. Each branch encodes some local image feature. Local connectivity and weight-sharing effectively reduce the number of weights to be determined during the training stage, thus allowing for relative small training sets for the (high) dimension involved.

We further investigate the concept of LRFs by extracting the output of the hidden layer of a (once trained) NN/LRF as features subject to classification by generic classification methods (other than neural networks). Preliminary experiments have shown receptive fields of size 5×5 to be optimal, shifted at a step size of 2 pixels over the input image of size 18×36. The number of branches is varied within the values of {8,16,24,32,48} during parameter optimization.

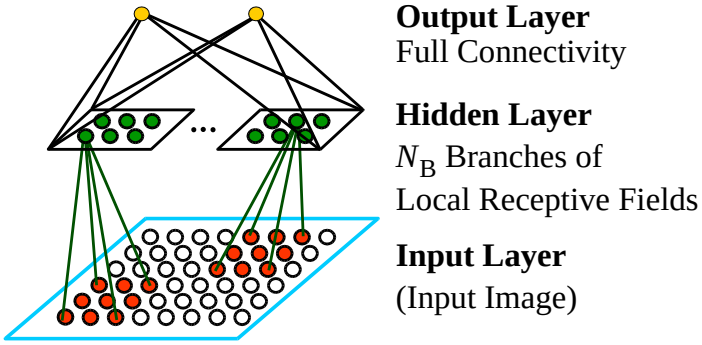


Figure 3.3: Architecture of a neural network with local receptive fields as employed by Wöhler and Anlauf [100]

3.2 Classification Methods

We now turn our attention to suitable methods for classification. We focused our selection on pattern classifiers that directly construct the decision boundary, rather than density estimation approaches [47] (e.g., Bayes Decision Theory or Parzen Classifier), given that the latter seem less suited for modeling the non-target class, which is, in a sense, not a real class but comprises the vast feature space of "everything else".

The generation of the LRF features inherently involves the training of a neural network. We consequently apply a *feed-forward neural network* to PCA and Haar wavelet features as well. The architecture chosen here is the simple but most common form of a (fully connected) three-layer network, where the number of hidden units N_{hidden} is adjusted via cross-validation.

Support Vector Machines (SVM) [97] have evolved as a standard tool for a broad range of classification tasks, including pedestrian classification [72, 64]. A possible advantage is the direct optimization of the margin of the decision boundary, hence, the classification error, opposed to the minimization of some artificial error term as, e.g., mean squared error for neural networks. The complexity of the decision boundary is determined by the kernel function. For our experiments, two different kernel functions are examined:

- polynomial: $K(x, y) = (x^T y + 1)^d$,
- radial basis function (RBF): $K(x, y) = \exp(-\gamma \|x - y\|^2)$.

Parameters d and γ of the kernel function and the error penalty term C of the SVM optimization objective (cf. Eq. (24) in [17]) are determined via cross-validation. Experiments are based on the LIBSVM implementation found in [11]. Note that the combination of Haar wavelet features and quadratic SVM closely resembles the system by Papageorgiou and Poggio [72].

Finally, a *k-nearest neighbor classifier* (k-NN) serves as a baseline classifier as it is able to handle arbitrary distributions without parameter adaptation, except for the number

k.

3.3 Methods for Increasing the Training Sample Size

Classification performance, in general, is known to scale with the training sample size [47]. We quantify this effect empirically in Section 3.5.2 with respect to our training sets, feature extraction, and classification methods. Yet, the acquisition of additional training examples is often limited by possibility and expense. For the problem at hand, for instance, pedestrian examples are obtained from manual labeling. On the other hand, non-pedestrian patterns, randomly extracted by some preprocessing module from a set of images not containing any pedestrians, come almost for free. We, hence, study two techniques known from the literature on how to iteratively select and utilize additional non-target examples based on an initial classifier, denoted as *bootstrapping* and *cascade*.

3.3.1 Bootstrapping

Sung and Poggio [90] employ a *bootstrapping* strategy to incrementally construct a training set of relevant non-target examples: False positives of an existing classifier are collected from a set of randomly extracted non-target patterns and added to the training set. A new classifier is then trained on the so-augmented training set, replacing the old one. This procedure is repeated until no further performance gain can be achieved.

3.3.2 Cascade

Viola et al. [99] employ a *cascade* strategy for combining multiple classifiers, where test patterns are successively classified by each stage of the cascade until the outcome of one stage is "non-pedestrian". Consequently, a test pattern is only assigned to the pedestrian class if all classifiers agree on that decision. The cascade is constructed iteratively: For each stage of the cascade, a new training set is generated by collecting false positives of the existing cascade out of a set of randomly extracted non-pedestrian examples, plus the original set of pedestrian examples. The classifier obtained from this new training set is then appended to the cascade.

3.3.3 Boosted Cascade of Haar-like Features

In addition to applying the cascade approach to the feature-classifier combinations described above, we also evaluate the cascade system of Viola and Jones [99] for comparison. Their system is based on a rich dictionary of simple appearance filters, similar to Haar wavelets. For each stage of the cascade, AdaBoost [27] iteratively constructs a weighted linear combination of simple classifiers, each made by thresholding one feature value. Iterations are stopped when a certain user-defined performance target is reached

Figure 3.4: Pedestrian and non-pedestrian samples from the benchmark data set (upper versus lower row, respectively).

and the training process continues with the next stage of the cascade. Our experiments are conducted using the implementation found in the Intel Open Source Computer Vision Library [71], with the target performance for each stage set to 50 percent false positive rate at a detection rate of 99.5 percent.

3.4 Benchmark Data Set

3.4.1 Data Sets

Figure 3.4 shows a few examples of pedestrian and non-pedestrian samples of the benchmark data set. Pedestrian examples were obtained from manually labeling (and extracting) the rectangular positions of pedestrians in video images, in a rather tedious and time consuming process. Images were recorded at various (day) times and locations with no particular constraints on pedestrian pose or clothing, except that pedestrians are standing in upright position and are fully visible. In order to make maximum use of these (valuable) labels, pedestrians images were mirrored and the bounding boxes were shifted randomly by a few pixels in horizontal and vertical directions. The latter is to account for small errors in ROI localization within an application system. Six pedestrian examples are thus obtained from each label.

As non-pedestrian examples, we extracted patterns representative of typical preprocessing steps within a pedestrian classification application from video images known not to contain any pedestrians. Examples of such preprocessing are background subtraction for surveillance applications or stereo-based object detection for in-vehicle applications. For our case of static, monocular images, we chose a shape-based pedestrian detector

Table 3.1: DaimlerChrysler pedestrian benchmark data set.

	Training Sets	Test Sets
Number of Data Sets	3	2
Pedestrian Labels Per Set	800	800
Pedestrian Examples Per Set	4800	4800
Non-Pedestrian Examples Per Set	5000	5000
Additional Non-Ped Images	$\geq$ 1200	

"Pedestrian Labels" denotes the number of pedestrians manually labeled, whereas "Pedestrian Examples" denotes the number of pedestrian examples in each data set derived from the pedestrian labels by mirroring and shifting.

[37] that matches a given set of pedestrian shape templates to distance transformed edge images. We included those patterns as negative samples to our classification training set, where the shape detector resulted in a match with the associated pixel-averaged chamfer-2-3 distance [7] to one of the given pedestrian shape templates below 2.5 (this corresponds to a maximum average per-pixel deviation of roughly 1.25 pixel) (see bottom row in Figure 3.4). Given the bounding box locations of interest in video images, examples were cut out after adding a border of 2 pixels to preserve contour information and scaled to common size 18×36, which was found optimal in preliminary experiments.

We split the resulting data base into five fully disjoint sets, three for training and two for testing (see Table 3.1), which allows for a variation of training and test sets during the experiments. Examples recorded at the same time and location are kept within the same set, so that, e.g., a pedestrian captured in a sequence of images does not show up in multiple data sets. This ensures truly independent training and test sets, but also implies that examples within a single data set are not independent – a fact taken into account in the test procedure below.

3.4.2 Test Procedure

Classification performance is evaluated by means of ROC curves, which quantify the trade-off between detection rate (the percentage of positive examples correctly classified) and the false positive rate (the percentage of negative examples incorrectly classified).

In order to compare the performance of two classifiers, we need a confidence interval to decide whether performance differences are significant or represent noise. Although the variance of test results obtained from a finite sample size has well been studied in the literature, this theory fails here because of (unknown) dependencies amongst the test examples. In fact, much larger performance variations have been observed in practice than one would expect from test samples of size 4,800 and 5,000 (see Table 3.1).

Consequently, we decided to empirically determine the ROC variance by varying train-

ing and test sets. While this is commonly done via cross-validation, we prefer not to interchange training and test data and to use a partition of the training data for parameter tuning. Parameters to be specified prior to training and testing of a classifier have been introduced above for each feature extraction and classification method. Cross-validation over the three training sets is used to determine optimal settings for these parameters.

Performance is then analyzed on the test sets as follows: For each experiment, three different classifiers are generated, each by selecting two out of the three training sets. Testing all three classifiers on both test sets yields six different ROC curves, i.e., six different detection rates for each possible number of false positives. (ROC points are interpolated where necessary.) When taken as six independent tests which follow a normal distribution, a confidence interval of the true mean detection rate is given by the t distribution as

$$\bar{y} \pm t_{(\alpha/2,N-1)} \frac{s}{\sqrt{N}} \approx \bar{y} \pm 1.05s, \tag{3.1}$$

where $\bar{y}$ and s denote the estimated mean and standard deviation, respectively, $1-\alpha = 0.95$ is the desired confidence interval, and $N = 6$ is the number of tests. Hence, the estimated standard deviation of the detection rate approximately represents a 95 percent confidence interval. Although this analysis is somewhat optimistic as it assumes independency of the individual ROC curves, it still provides a reasonable indicator for performance comparison.

3.5 Experimental Results

This section provides comparative experimental results of the techniques described in Sections 3.1, 3.2, and 3.3. In a first batch of experiments, we apply each classification method to each type of features, whenever appropriate, in order to allow for a separate investigation into the effectiveness of features and classifiers. The benefit of increased training sample sizes is then evaluated in a second batch of experiments, based on the best two feature-classifier combinations identified so far.

3.5.1 Combinations of Feature Extraction and Classification Methods

All experiments in this section are conducted using two (out of three) training sets for training and the remaining one for validation. After parameters have been optimized via cross-validation (see Table 3.2), an evaluation of the mean and variance of ROC performance is done on the two test sets as described above.

Individual results for each feature type are given in Figures 3.5(A), 3.5(B), and 3.5(C). Figure 3.5(D) provides a comparison of the different feature types by selecting the best performing classifier for each feature. Two observations can be made: First, global

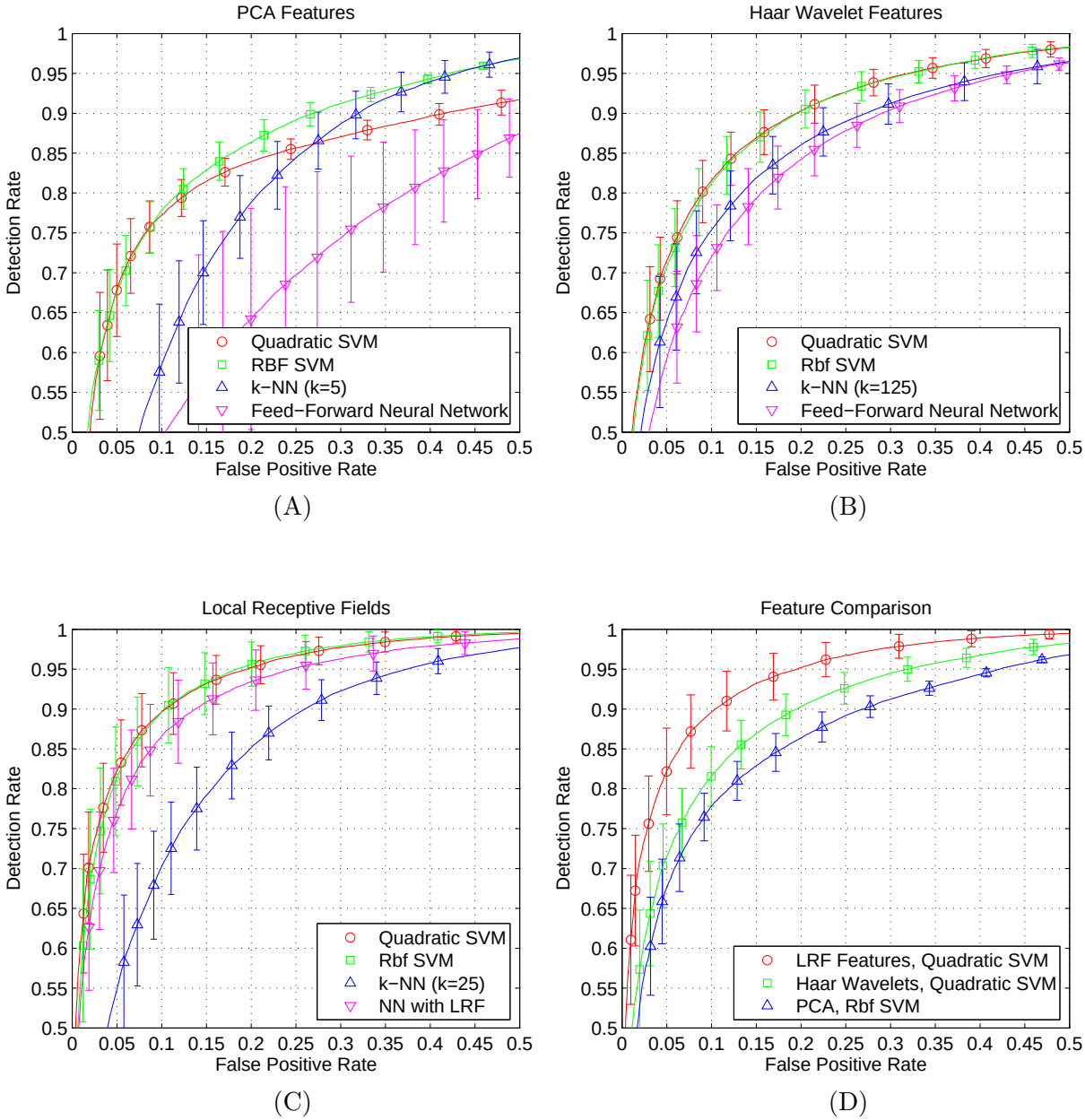


Figure 3.5: A comparison of different feature extraction and classification methods. Performance of different classifiers on (A) PCA coefficients, (B) Haar wavelet and (C) LRF features. (D) A performance comparison of the best classifiers for each feature type.

Table 3.2: List of the parameter settings that have been found optimal for each feature-classifier combination via cross-validation.

PCA Features	
Polynomial SVM	95%, $d = 2$, $C = 1$
RBF SVM	90%, $\gamma = 10^{-2}$, $C = 10$
k-NN	90%, $k = 5$
Neural Network	90%, $N_{hidden} = 20$

Haar Wavelet Features	
Polynomial SVM	$d = 2$, $C = 1$
RBF SVM	$\gamma = 10^{-3}$, $C = 100$
k-NN	$k = 125$
Neural Network	$N_{hidden} = 40$

Percentages denote the fraction of variance to be captured, see Section 3.1.1.

Local Receptive Fields	
Polynomial SVM	$N_B = 16$, $d = 2$, $C = 1$
RBF SVM	$N_B = 16$, $\gamma = 10^{-3}$, $C = 100$
k-NN	$N_B = 16$, $k = 25$
Neural Network	$N_B = 16$

Table 3.3: Number of support vectors and processing time for SVMs on each of the three feature types, obtained from the first two training data sets. Numbers are given for the best parameter setting and SVM kernel function found.

Feature Type	Feature Dimension	Kernel Type	Number of Support Vectors	Processing Time Per Example
PCA (90%)	61	RBF	4,704	3.7 ms
Haar Wavelets	1,755	quadratic	7,042	41 ms
LRF ($N_B = 16$)	1,792	quadratic	5,160	33 ms

features, represented by PCA coefficients, are inferior to local features (Haar wavelets, LRFs). The reason for this may lie in the fact that sometimes very small details such as hands, feet, or the form of the head make the difference between pedestrians and other objects. Such details are smoothed out by PCA dimensionality reduction. Second, adaptive features (LRFs), which have been tuned to the data during the training process, outperform non-adaptive ones (Haar wavelets).

Regarding classifiers, SVMs generally perform best. This holds even for LRF features that have been generated by a neural network. However, SVMs require the largest processing time of the classifiers tested due to the high number of support vectors (see Table 3.3). On LRF features, the training process yielded 5,160 support vectors on average, leading to a processing time of about 33 ms per example. For comparison, the neural network with LRFs only takes about 1 ms per example (both implementations in C/C++ on a 3.2GHz Pentium IV PC).

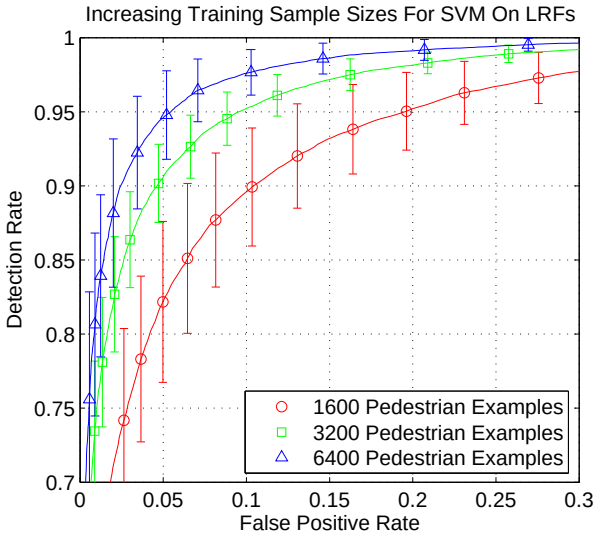
3.5.2 Increasing the Training Sample Size

The two best classification techniques identified above, quadratic SVM on local receptive fields and quadratic SVM on Haar wavelet features, are employed for these experiments. We first evaluate the benefit of manually increasing the training sample size from an auxiliary data set. The number of training examples is doubled two times, so that the training sets consist of 3,200 and 6,400 pedestrian and 20,000 and 40,000 non-pedestrian examples, respectively. Again, classifier parameters are first optimized via threefold cross-validation and the mean and variance of ROC performance is evaluated on three different training and two different test sets. Resulting ROC curves are given in Figure 3.6(A,B) for both classifiers. An additional experiment of doubling the training sample size three times has been performed with the neural network with local receptive fields, see Figure 3.6(C), where the same effect has been observed.

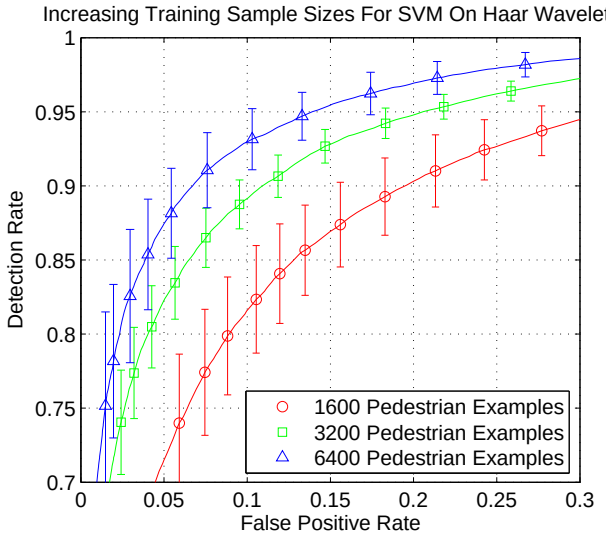
Interestingly enough, classification errors are reduced by approximately a factor of two whenever the training sample size is doubled; no saturation effects are yet observed. Notice, furthermore, that the performance differences caused by increasing the number of training examples exceed the differences between different feature extraction methods. The relative performance difference between the feature types remains the same, i.e., LRFs maintain their superiority.

We now evaluate to what extent the benefit of additional training examples can be achieved by the automatic extraction of new non-pedestrian patterns by means of *bootstrapping* and *cascade*. Both techniques are applied iteratively, generating 10,000 new non-pedestrian examples in each iteration, which equals the number used for the initial classifier. In all combinations considered, the maximum performance was reached after the third iteration. Results are given in Fig 3.7. Though both approaches quickly reached their limits, a consistent performance improvement was achieved. A comparison of both strategies reveals a small advantage of the bootstrapping approach. This benefit is, however, paid with higher computational costs, as incrementally more complex training sets imply incrementally more complex classifiers.

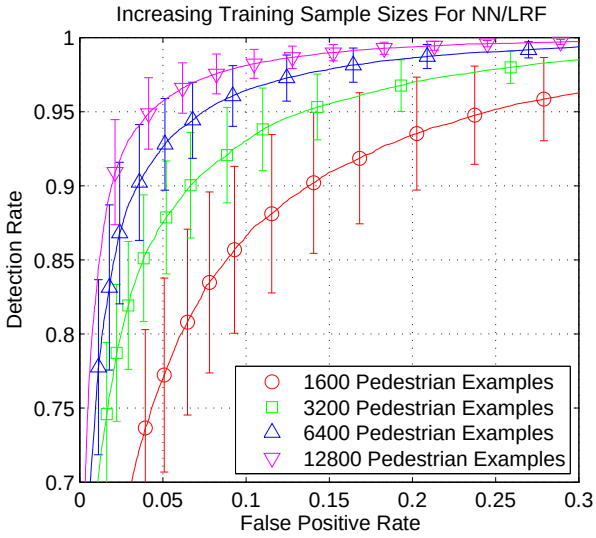
Results of the AdaBoost cascade system by Viola and Jones are given in Figure 3.8(A). The performance of the initial cascade stages is limited by the user-defined training termination criterion (set to 50 percent false positive rate at a detection rate of 99.5 percent). The entire eight-stage cascade, however, achieves about the same performance as the cascaded SVM applied to Haar wavelet features, see Figure 3.8(B). Although adding more stages to the cascade further reduces the training set error, performance on the validation and test sets was observed to run into saturation. The main advantage of this approach, though, is processing speed. In our implementation, the cascade of eight AdaBoost classifiers runs, on average, at 0.4ms per test sample, whereas the 4 stage SVM cascade requires a significantly higher 250ms on average (both implementations in C/C++ on a 3.2GHz Pentium IV PC).



(A)



(B)

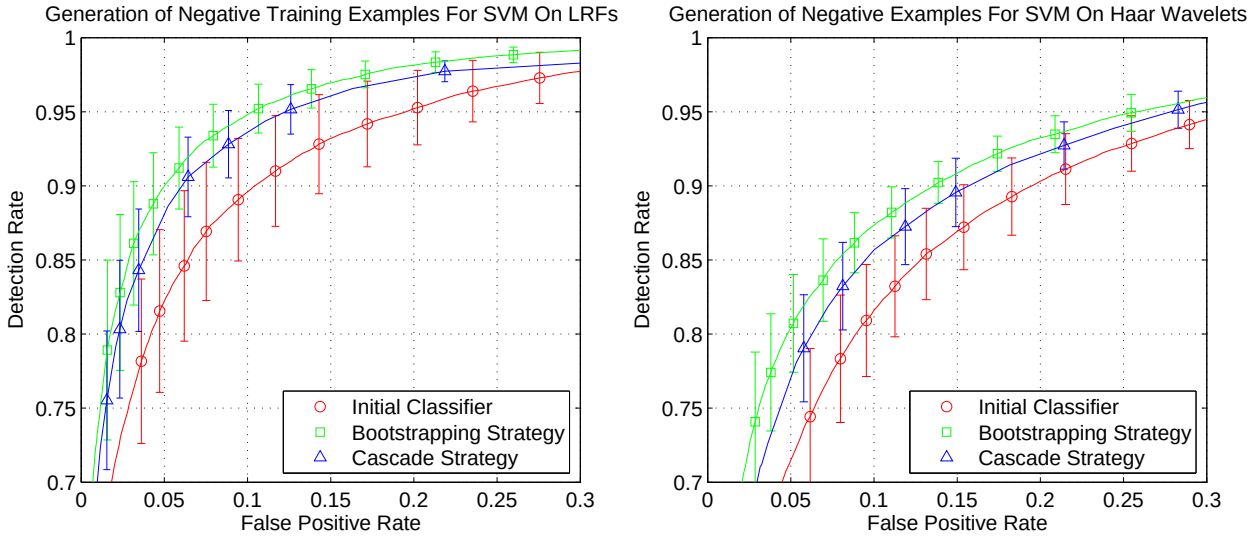


(C)

Training Sample Size	Number of LRF Branches N_B
1600	16
3200	16
6400	24
12800	48

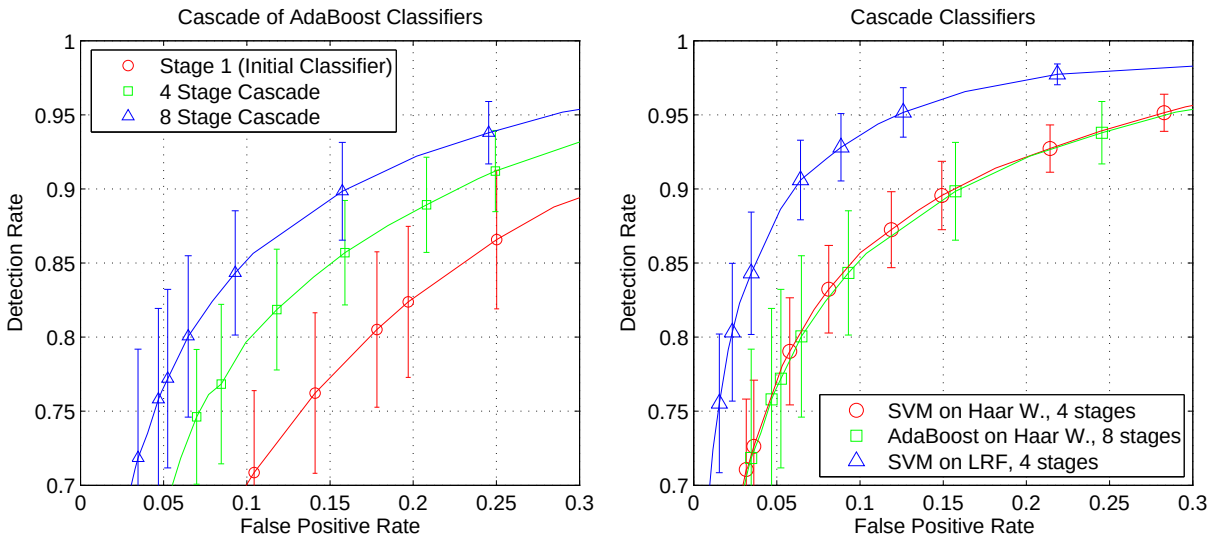
(D)

Figure 3.6: Performance gain by increasing training sample sizes for (A) quadratic SVM on local receptive fields (LRF), (B) quadratic SVM on Haar wavelet features, and (C) neural network with local receptive fields. Table (D) shows what numbers of LRF branches N_B have been found optimal for each training sample size (for both, NN and SVM on LRFs). Remaining parameters are unchanged.



(A) (B)

Figure 3.7: A comparison of the bootstrapping versus the cascade strategy for (A) quadratic SVM on LRFs, and (B) quadratic SVM on Haar wavelet features.



(A) (B)

Figure 3.8: (A) ROC performance of the AdaBoost cascade by Viola and Jones [99]. (B) Comparison of the three cascade results.

3.5.3 Summary

The experimental examination of different combinations of feature extraction and classification techniques has revealed superiority of local features over global ones. Among the latter, adaptive features (local receptive fields) outperformed non-adaptive ones (Haar wavelets). Regarding classification methods, SVMs outperformed the other classifiers tested, except for the AdaBoost cascade approach, which achieved comparable performance at much lower computational costs.

What choice to make for a practical real-time application depends on the number of ROIs to process. While for pure sliding window-based applications, the AdaBoost cascade will be the method of choice, local receptive fields in combination with a neural network or even an SVM are attractive for post-processing the output of a faster detector.

The greatest performance gain was achieved by increasing the training sample size. Here, the automatic generation of non-pedestrian examples resulted in a performance gain that, after few iterations, ran into saturation. Not so for the addition of target examples at the quantities considered. The obvious consequence is to diligently continue collecting more training (target) samples, but this is time consuming. Thus, techniques for extending and designing the training set using interactive learning techniques seem an especially worthwhile direction of further research.

4 Integration of Shape and Texture

Texture-based classification methods have been shown to provide effective discrimination between patterns of the target (i.e. pedestrian) class and the non-target class. Yet challenges remain due to the high complexity of appearance variation of both, target and non-target patterns, in conjunction with a high dimensionality. The goal of this chapter is to examine how texture classification benefits from explicit prior knowledge about object appearance variation.

The appearance variance of pedestrians arises from three different sources: Pedestrian foreground texture varies with clothing and illumination, shape variation is induced by body articulation and different viewpoints, and there is a variable background. No prior knowledge about the foreground and background texture distributions applicable to pattern classification is available (see Sections 2.2 and 2.3). The utilization of these cues is left to a generic pattern classifier that learns a discrimination function directly from a training set.

The variability of pedestrian shapes, however, is subject to physiological and application-specific constraints. These constraints allow to build explicit representations of pedestrian shape as reported in Section 2.2. Matching the shape representation to an input image then provides shape knowledge that can be used to reduce the intra-class appearance variability, and, hence, to simplify the classification problem. This is done in two steps: First, clusters of distinct body pose, defined by viewpoint and articulation, are identified and one specialized texture classifier is generated for each pose cluster. Online, a probabilistic assignment of (unknown) input patterns to the pose-specific experts is derived from the shape matching results. The better an image fits to the shape of a particular pose cluster, the higher the cluster membership probability. This resembles a pose-specific *mixture-of-experts* architecture that combines pose-specific texture classifiers (the local experts) with shape matching for expert selection.

In a second step, a parametric representation of the shapes within each cluster is used to perform shape normalization by warping input images to the reference shape of the associated pose cluster. As a result, shape variability is eliminated from the texture patterns, which is supposed to simplify the classification problem.

The resulting architecture implements a mixture model of shape and texture by integrating a generative, explicit shape model into discriminative texture classification. An overview is shown in Figure 4.1. This model is particularly attractive for application within a multi-cue recognition system that combines shape matching for object localization and texture classification for object discrimination, so that the shape-based gating function comes at no extra cost. Two such examples are presented in Chapters 5 and 6.

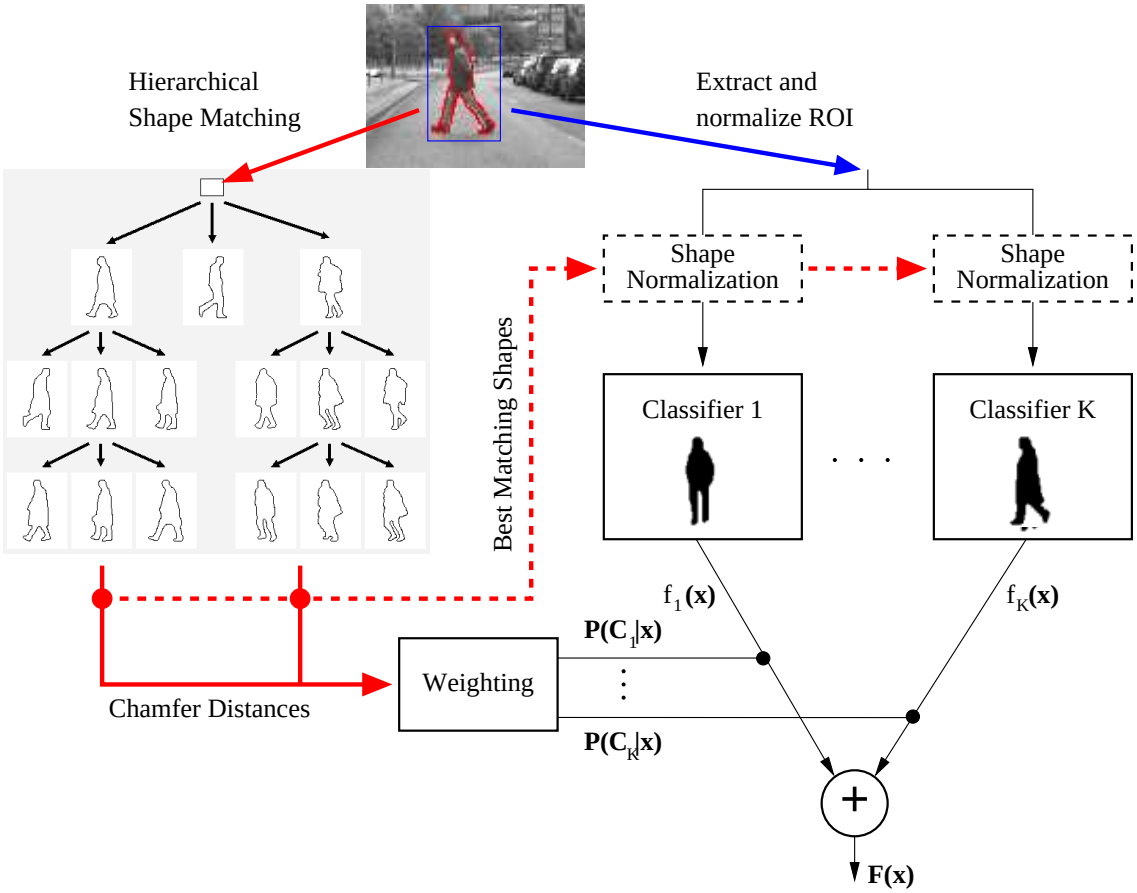


Figure 4.1: Overview of the proposed pose-specific mixture-of-experts architecture integrating shape and texture: Shape matching is applied to an input image to determine shape information (left). For each pose cluster, the best matching shape instance along with a measure of the matching quality is determined. In an optional pre-processing step, the matching shape instances are used to perform shape normalization, i.e. to warp the input pattern to a reference shape; separately for each pose cluster. The resulting texture pattern is then processed by a mixture of patterns classifiers, each specialized to one pose cluster. The final result is obtained by a linear combination of the classifiers' outputs, where the mixture coefficients are derived from the shape matching results.

This chapter focuses on utilizing shape matching for texture classification. Below, the underlying shape representations are introduced first before the two integration steps are presented. This chapter is complemented by a thorough experimental analysis of the proposed approach.

4.1 Shape Representation

The purpose of this section is to introduce the two different representations of object shape utilized within this thesis, and to provide brief descriptions of the acquisition and application of these representations. Motivations for our choice were the fact that both representations can be learned from 2D example images, and that efficient matching techniques exist.

In this work, the shape of a target object is described by a list of 2D contour points,

$$C = \{(u_1, v_1), \ldots, (u_n, v_n)\} \,, \tag{4.1}$$

or, equivalently, by the binary image $I(C)$ of contour points. In order to compare two shapes, we make use of a multi-feature distance transform as proposed in [37], with feature types given by discretized edge orientations. Given a shape C and a feature image I, the multi-feature *chamfer* distance is defined as

$$D_{chamfer}(C, I) = \sum_{m \in M} \frac{1}{|C_m|} \sum_{c \in C_m} d_{I_m}(c) \,, \tag{4.2}$$

where M is the set of feature types, the index m denotes the subset of features of type m, and $d_I(c)$ denotes the distance between contour point c and the closest feature in image I. For the latter, we make use of the *chamfer-2-3* metric [7]. A symmetric distance functions of two shapes C_1 and C_2 is then derived as

$$D(C_1, C_2) = \frac{1}{2}\Big\{ D_{chamfer}(C_1, I(C_2)) + D_{chamfer}(C_2, I(C_1)) \Big\} \,. \tag{4.3}$$

Advantages of this choice of a distance function are the gradual measure of shape dissimilarity, and the avoidance of explicit feature point correspondences. See [37] for further details.

The distribution of pedestrian contours is complex due to object deformations, unrestricted viewpoints, and self-occlusion. Of the many approaches developed in the past, two representations of object shape are of particular interest for this work and discussed below. Both are learned from a training set of object shapes, $\mathcal{C} = \{C_1, \ldots, C_N\}$, obtained from manual labeling of contour pixels.

4.1.1 Exemplar-based Shape Representation

The first approach is to directly use the training set as an exemplar-based representation of the shape distribution. Together with the above distance function, this parameter-free representation evades an explicit, error-prone shape registration.

In order to handle large data sets efficiently, Gavrila and Philomin [37] proposed to organize the shape set in a hierarchical tree structure. Offline, this hierarchy is constructed automatically in a bottom-up, level-by-level fashion using clustering. For each level, the intra-cluster variance

$$E = \sum_{k=1}^{K} \min_{P \in \mathcal{C}_k} \max_{C \in \mathcal{C}_k} D_{chamfer}(C, P) \tag{4.4}$$

is minimized by means of *simulated annealing* [52]. At its top level, this hierarchy provides a partitioning of the training set into disjoint clusters, $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_K$, where, ideally, each cluster represents a certain pedestrian body pose. Online, matching the shape hierarchy to a given input pattern involves a tree traversal process which can be done in an efficient coarse-to-fine manner [37].

4.1.2 Multi Point Distribution Model (MPDM)

The second representation is adopted from Gavrila et al.[35] and provides a dynamic, parametric shape model based on *Point Distribution Models* (PDMs) [14]. Opposed to the exemplar-based representation above, this one allows to synthesize new shape examples from the model.

Clustering. For this to work, shapes need to be registered into a common vector space by establishing point correspondences. But two arbitrary 2D contours do not necessarily possess physically corresponding points due to differing viewpoints and/or self-occlusion, so that one global vector space does not adequately describe the shape manifold. Instead, the training set is first partitioned into K pose clusters of common viewpoint and articulation. Automatic techniques are available, e.g. [35, 5], but we decided to do this first step manually to take texture information into account that are not available from the list of contour points; e.g., to distinguish front and back views of pedestrians. As above, let $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_K$ denote the K disjoint pose clusters. An example of such manually defined clusters is shown in Figure 4.5.

Registration. Point correspondences within each pose cluster are then obtained in a semi-automatic process, as described in [35]. Roughly, the procedure for registering two shapes is:

1. Align center points and heights of both shapes by translation and scaling.

2. Determine interest points of extreme curvature on both shapes.

3. Compute the correspondence of minimum average Euclidean distance between the interest points using Dynamic Programming.

This fully automatic algorithm is followed by a manual refinement step to correct obvious registration errors. For each cluster k, the training example with minimum average distance to other cluster elements is taken as the cluster prototype, denoted by C_k^p. A common vector space is then established by the registration of all example shapes of cluster k to that prototype. Its dimension d_k equals twice the number of correspondence points. Let $\mathbf{s} \in \mathbb{R}^{d_k}$ denote a shape vector of correspondence points obtained by registration.

Linear subspace model. In order to obtain a compact representation of the shape variations within each cluster, the *Active Shape* approach of Cootes et al.[14] is adopted. Dimensionality of each local vector space is first reduced using principal component analysis (PCA), where the number of eigenvectors to retain is chosen such that a user-supplied fraction of the total variance is explained. (We use 95% throughout this thesis.) A Mahalanobis threshold is then determined that covers a user-supplied fraction of the training examples. (We use 75%). Examples outside the resulting hyperellipsoid are considered outliers, while a (truncated) normal distribution is assumed within the hyperellipsoid.

Formally, let A_k denote the matrix of selected eigenvectors in its columns, and let Λ_k denote the diagonal matrix of corresponding eigenvalues. A shape $\mathbf{s}$ in cluster k is represented by the pair $(k, \mathbf{b})$, with

$$\mathbf{b} = A_k^T (\mathbf{s} - \bar{\mathbf{s}}) \tag{4.5}$$

being the vector of PCA coefficients; $\bar{\mathbf{s}}$ is the cluster mean[1]. The prior shape distribution is given by

$$p(k, \mathbf{b}) \propto P(k) \begin{cases} N(\mathbf{b}; 0, \Lambda_k) & \text{if } \mathbf{b}^T \Lambda_k^{-1} \mathbf{b} \leq M^2 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where M is the Mahalanobis threshold.

MPDM Matching. Matching the MPDM shape representation to an input image involves to first select the shape cluster k, and then to adapt the coefficient $\mathbf{b}$ to fit the input image. If no prior information about the shape cluster is available, shape fitting is performed in parallel for all clusters, and the best match is selected afterwards.

Matching one local subspace model to an input image is done by a snake-like algorithm [14]. Starting with the cluster mean shape, the following two steps are performed iteratively:

[1]Note that the cluster mean might be different from the cluster prototype in this context.

- Project the current shape parameters to the input image and adapt the list of contour points by means of Dynamic Programming to minimize the chamfer distance (4.2) between the projected shape and the input image.
- Project the adapted list of contour points to the local subspace and apply the Mahalanobis threshold M.

This procedure either terminates after a fixed number of iterations, or when no further reduction of the chamfer distance between the projected shape and the input image can be achieved.

4.2 Pose-Specific Mixture of Experts

Mixture-of-experts architectures [46] pursue a divide-and-conquer strategy. The classification problem is divided into a number of simpler subproblems, and each is handled by a specialized classifier, called the *local expert*. Compared to a single global classifier, a performance gain is attained if each local expert outperforms the global classifier within its local subregion. The association of input patterns to the local experts is done by a gating function which is either learned jointly with the experts [46, 49] or obtained separately from a clustering of the training data set [40, 55].

Though intuitively promising, dividing the training set into smaller subsets might also have unfavorable consequences to the classification performance due to over-fitting, especially in high-dimensional input spaces where training sets naturally become sparse. In general, dividing the input space increases the variance of classification error. The solution proposed in the literature [46, 49] is to use "soft" splits of the data, such as modeling probabilistic cluster associations, thus allowing subregions to overlap. This alleviates the effects of reduced training sample sizes of the local experts, and results in a weighted linear combination of the experts' outputs, which is known to reduce the variance of classification error [94, 29].

Regarding the pedestrian recognition application, natural clusters of distinct appearance are given by viewpoint (e.g., frontal or back views vs. side views) and articulation (e.g., feet closed vs. knees closed), which generate distinctive object silhouettes. This motivates our pose-specific mixture-of-experts approach with a shape-based gating function and texture-based local experts. By relying on different image cues, the gating function effectively introduces additional knowledge (shape) to texture classification.

The mixture architecture also exhibits a strong relation to the concept of classifier fusion. Additive combinations of classifiers ("sum rule") have been shown to outperform single classifiers, given that the component classifiers are sufficiently diverse [73, 53, 95, 56]. Whereas in previous works on classifier fusion, diversity is created by using different classification models, different initializations (in conjunction with neural networks), or random splits of the training data, mixture-of-experts architectures inherently ensure diversity by creating distinct training sets.

4.2.1 Shape-Based Gating Function

Both shape representations introduced in Subsections 4.1.1 and 4.1.2 above are based on a partitioning of the training set. Instead of generating a clustering anew, the shape clusters are re-used so that a pose clustering of the texture patterns can be derived from shape matching. One could simply use the partitioning of the training set and train a separate classifier for each subset. However, the variation of camera viewpoint and human articulation is continuous, so that ambiguities in cluster associations arise. Furthermore, errors in shape matching occur due to missing edge features or background clutter. Both problems are handled by modeling cluster associations probabilistically, i.e., by determining the posterior probability that a texture pattern $\mathbf{x}$ belongs to a certain pose cluster, given the shape matching results.

Let $\mathcal{C}_j$, $j = 1, \ldots, K$, denote the pose clusters given by the shape model. Matching the shape model to texture pattern $\mathbf{x}$ involves to compute, for each shape cluster $\mathcal{C}_j$, the best matching shape instance $\mathbf{t}_j^*(\mathbf{x})$ and residual chamfer distance $d_j(\mathbf{x})$:

$$d_j(\mathbf{x}) = \min_{\mathbf{t} \in \mathcal{C}_j} D_{chamfer}(\mathbf{t}, I_{\mathbf{x}}) \,, \tag{4.7}$$

where $I_{\mathbf{x}}$ denotes the feature image obtained from $\mathbf{x}$. We represent the cluster-conditional density by an exponential function,

$$p(d_j(\mathbf{x}) \,|\, \mathcal{C}_j) \approx \alpha_j e^{-\alpha_j d_j(\mathbf{x})} \,, \tag{4.8}$$

and model the posterior $P(\mathcal{C}_j \,|\, \mathbf{x})$ given the cluster distances as observations. Assuming equal prior probabilities, the latter is given by Bayes law, [2]

$$P(\mathcal{C}_j \,|\, \mathbf{x}) \approx \frac{\alpha_j e^{-\alpha_j d_j(\mathbf{x})}}{\sum_{k=1}^{K} \alpha_k e^{-\alpha_k d_k(\mathbf{x})}} \stackrel{def}{=} w_j(\mathbf{x}) \,. \tag{4.9}$$

Let $w_j(\mathbf{x})$ denote the cluster posterior approximations which are subsequently used as the pose expert weights. Parameters α_j are determined on the training set by maximizing the likelihood

$$\mathcal{L}(\alpha_j \,|\, \mathcal{C}_j) = \prod_{\mathbf{x} \in \mathcal{C}_j} p(d_j(\mathbf{x}) \,|\, \mathcal{C}_j) \,. \tag{4.10}$$

Substituting Eq. (4.8) and solving for α_j yields

$$\frac{1}{\alpha_j} = \frac{1}{|\mathcal{C}_j|} \sum_{\mathbf{x} \in \mathcal{C}_j} d_j(\mathbf{x}) \,. \tag{4.11}$$

[2] Strictly, this is an approximation of the posterior since the observation d_j depends on the state hypothesis $\mathcal{C}_j$.

4.2.2 Expert Training and Combination

Each pose expert is aimed to discriminate texture patterns of the target class from those of the non-target class, given it belongs to the particular pose cluster of that expert. More specifically, we wish expert f_j to approximate the cluster-conditional posterior

$$f_j(\mathbf{x}) \approx P(\mathcal{O} \,|\, \mathcal{C}_j, \mathbf{x}) \,, \tag{4.12}$$

where $\mathcal{O}$ denotes the object class. This is achieved by training expert f_j on the modified distribution w_j of the training set, either by using a weighted training set or by resampling the training set. Then, the final target class posterior is the weighted average of the pose experts, i.e.,

$$P(\mathcal{O} \,|\, \mathbf{x}) = \sum_{j=1}^{K} P(\mathcal{C}_j \,|\, \mathbf{x}) P(\mathcal{O} \,|\, \mathcal{C}_j, \mathbf{x}) \tag{4.13}$$

$$\approx \sum_{j=1}^{K} w_j(\mathbf{x}) f_j(\mathbf{x}) \stackrel{def}{=} F(\mathbf{x}) \,. \tag{4.14}$$

Figure 4.1 illustrates this architecture.

4.2.3 Implementation Details

A few design considerations have critical impact to the classification outcome, and are discussed below.

Number of clusters. The definition of the pose clusters used here is given by the shape clustering of the shape model. However, shape clustering methods, manual or automatic, tend to generate a large number of clusters with low intra-class variance, suitable for representation by simple models such as the linear PDM model described in Section 4.1.2. But this choice may be inappropriate for generating texture classifiers, as too small training sample sizes remaining in each cluster degrade classification performance. Hence, the number of clusters K used for generating the shape models is manually set to a small number, and the optimum is found empirically. For the experiments below, K is varied within the range 3 to 12.

Expert complexity. Intuitively, one could assume that simpler classifiers can be used for the pose experts compared to a single classifier approach due to the reduced complexity of each pose cluster, but this assumption is misleading. Previous studies of the bias-variance trade-off, e.g. [94, 29], have pointed out that classifier combination via averaging requires component classifiers to have low bias and low correlations, since error variance is reduced by their combination. Consequently, the pose experts should be highly specialized to their pose cluster (low error bias but high variance), which can be

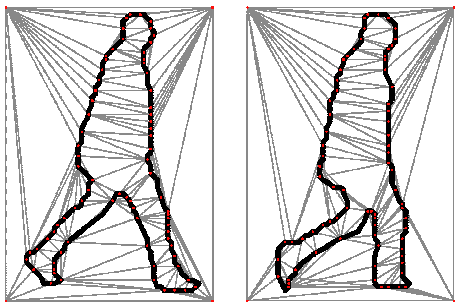


Figure 4.2: Illustration of shape normalization [22]: Correspondence points (red dots) are determined along the contours of the source (left) and target shape (right). The correspondence points of the source shape are triangulated using the Delaunay algorithm. A transformation of the triangle is then established by shifting the correspondence points to their new location.

achieved by increasing the complexity of the component classifiers, i.e., by increasing the number of free parameters to adjust during the training process. We account for this by initially choosing each expert's complexity equal to that of the optimal single classifier, followed by experimental tuning.

Weight approximation. Determination of the weights $\mathbf{w}(\mathbf{x})$ is rather time-consuming as it requires the computation of the chamfer distance of a previously unseen texture pattern to all shape templates (Eqs. (4.7) and (4.9)). If online processing employs hierarchical chamfer matching as described in [37], only the best matching shape template is given as output. Hence, we speed up online processing by pre-computing a fixed weight vector $\hat{\mathbf{w}}^{(i)}$ for each shape cluster $\mathcal{C}_i$. Online, cluster index i is given by the best matching shape template. The best approximation of the true weight vector of Eq. (4.9) is given by its expectation over the set $\mathcal{D}_i$ of training examples that have the best matching shape template $\mathbf{t}^*$ in cluster $\mathcal{C}_i$:

$$P(\mathcal{C}_j \,|\, \mathbf{t}^* \in \mathcal{C}_i) \approx \mathcal{E}_{\mathcal{D}_i}[w_j(\mathbf{x})] \stackrel{def}{=} \hat{w}_j^{(i)} \,, \tag{4.15}$$

where

$$\mathcal{D}_i = \{\mathbf{x} \in \mathcal{D} | i = \arg \min_{i'=1\ldots K} d_{i'}(\mathbf{x})\} \,. \tag{4.16}$$

4.3 Shape Normalization

Matching the shape mixture models to an input pattern yields two kinds of information, the pose cluster association and the local contour deformation within each pose cluster. While cluster associations were used in the mixture-of-experts architecture above, object contour information is to be utilized now for shape-normalizing the foreground region and for masking out background pixels.

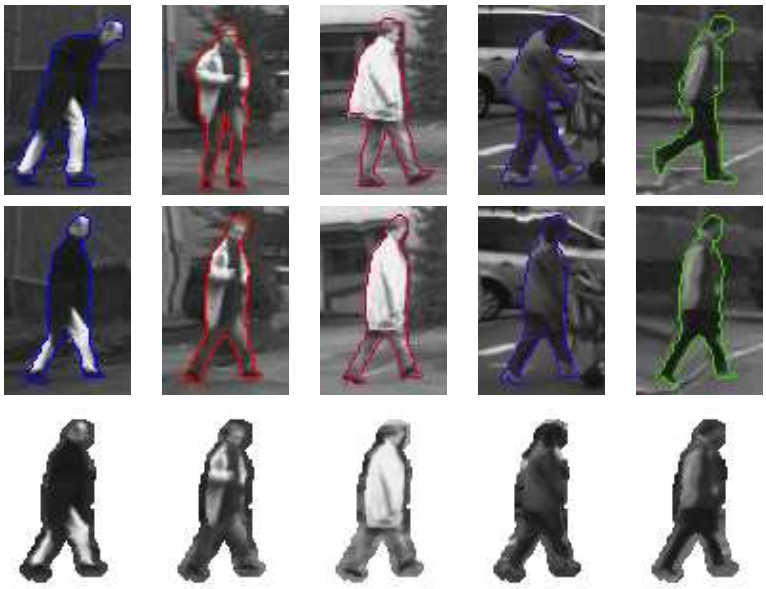


Figure 4.3: Examples of shape normalization and background masking. Top row shows a few example of the training set that are all (manually) assigned to the same pose cluster ("Right"/"Knees Apart"). Warps to the cluster reference shape are shown in the second row, the third row shows the same examples with background pixels removed (excluding a margin of 2 pixels around the object contour). Contour labels are superimposed on the examples images for visualization purposes.

Contour point correspondences, as available in the MPDM shape model, provide an explicit representation of local contour deformation. By using these correspondence points as support points for image warping, all input patterns (target and non-target) can be warped to a common reference shape. Shape variability is thus eliminated from the input space, i.e., patterns are shape-normalized, which effectively reduces appearance complexity to be handled by the texture classifiers and potentially leads to improved classification performance.

Various image warping techniques are available from the literature. Here, we employ a piecewise affine warp based on Delaunay triangulation [2] and bilinear interpolation. See Figure 4.2 for an illustration, and Figure 4.3 for examples. Details of the algorithm are given in [22]. Shape normalization is done locally within each pose cluster, since contour point correspondences are not defined between pose clusters by our shape models.

Secondly, the object contour obtained from shape matching separates foreground and background pixels. In most applications, background clutter increases appearance complexity without providing additional information about the object class. Removing background pixels from the input patterns is, hence, supposed to be beneficial to classification performance. In conjunction with shape normalization, the foreground mask and thus the input dimension is fixed, so that background masking is applicable with any texture classification method. For the experiments below, a small margin (e.g., 2 pixels) is added to the foreground region in order to preserve edge information along the object contour.

If probabilistic cluster associations are involved, then shape normalization and background masking are performed K times for each input pattern, once for each pose cluster. Each warping process transforms the best matching shape of one pose cluster to the corresponding cluster reference shape. An illustration of the resulting architecture is given in Figure 4.1. The effect of shape normalization and background masking is evaluated experimentally in the next section.

An interesting alternative to shape normalization was proposed by Enzweiler [22]. Instead of eliminating shape variation from the input patterns, the given prior knowledge about shape variation is represented by a virtual training set in that virtual samples are generated from the combined shape-texture model. This works by sampling shape and texture independently from the (existing) training set, and by warping the selected texture pattern to the new shape. A selective sampling method based on *Active Learning* [12] guides the sample generation process towards the more informative examples. Though this approach requires more complex local experts compared to shape normalization, it is potentially more robust since shape matching errors are avoided and texture warping is only applied to the training set where accurate contour labels are available.

4.4 Experiments

This section evaluates the benefit of the mixture-of-experts approach in comparison to a single texture classifier, and experimentally verifies the assumptions made above.

Figure 4.4: Example results of automatic shape clustering. Cluster prototype shapes of the 5 pose clusters are shown on the left, and examples of best-matching texture patterns ("hard partitioning") are shown to the right. Although most cluster associations are correct, errors frequently occur either from shape matching failures (e.g., rightmost examples in rows 3 and 5), or from association ambiguities where examples fall in-between two clusters (e.g., 4th and 5th example in row 3).

Following the results of the previous chapter, a neural network with local receptive fields was selected as the pattern classifier for all experiments in this chapter. The number of LRF branches N_B was set to 24 unless otherwise stated.

4.4.1 Data Sets

Experiments are conducted on a large real-world pedestrian data set. Video images were recorded at various times and locations in order to capture a great variety of pedestrian appearances. Pedestrians were required to be fully visible or have only minor occlusion such as one arm or foot. The data were split into a training set and a fully disjoint test set. In the training set, pedestrian contours were manually labeled, and the positions of certain body parts were marked to obtain point correspondences. Pedestrians in the

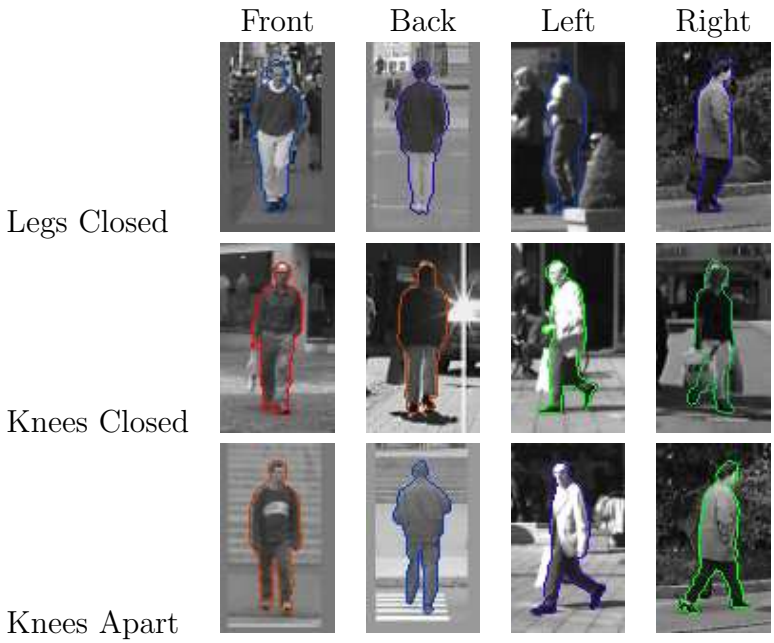


Figure 4.5: For manual clustering, 12 pose clusters of distinct viewing direction and leg position were defined, and each pedestrian training example is manually assigned to one of these clusters. Merging front and back views yields 9 clusters, further merging leg articulations "Knees Closed" and "Knees Apart" leads to 6 cluster. In addition, 3 different pose clusters are obtained by only considering leg articulation. Contour labels are superimposed on the examples images for visualization purposes.

test set are only labeled by their bounding box position.

Examples have been mirrored and shifted by a few pixels in both directions to increase the number of pedestrian examples, and to account for small localization errors. In order to obtain meaningful non-pedestrian patterns from sets of images without pedestrians, shape template matching was applied at random positions, and only matches below a threshold of 2.5 pixels were extracted. See Table 4.1 for statistics of the resulting training and test data sets.

4.4.2 Pose Clustering

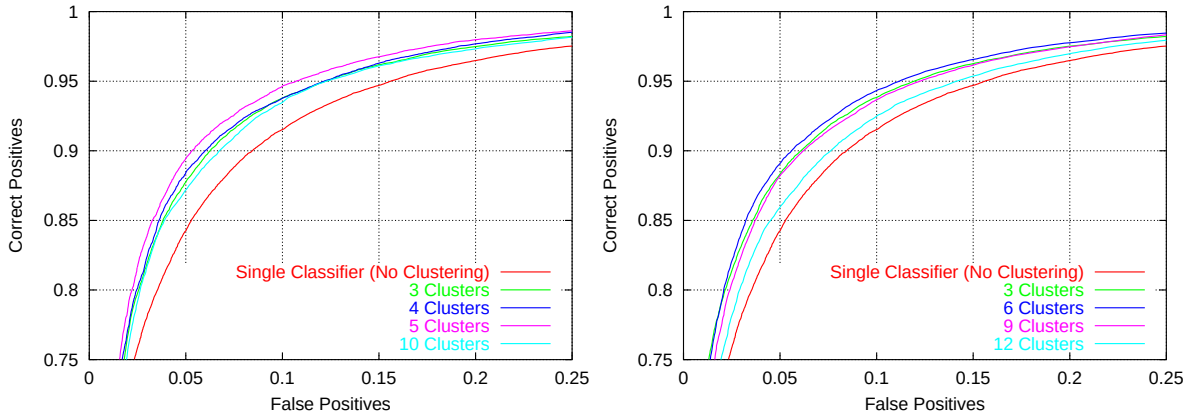
The first processing step involves the generation of the pose clusters. Two approaches, automatic shape clustering as employed for the exemplar-based shape representation and manual clustering as proposed for the MPDM model, are pursued and compared experimentally w.r.t. their benefit to texture classification performance.

Automatic shape clustering by means of simulated annealing (see Section 4.1.1) operates directly on the contour labels of the training set. The number of clusters K was

Table 4.1: Training and test data set statistics.

	Training Sets	Test Sets
Pedestrian Labels	6,522	3,375
Pedestrian Examples	61,640	58,362
Non-Pedestrian Examples	45,853	48,455

"Pedestrian Labels" denotes the number of pedestrians manually labeled, whereas "Pedestrian Examples" denotes the number of pedestrian examples in each data set derived from the pedestrian labels by mirroring and shifting.



(A) (B)

Figure 4.6: ROC results of different pose clusterings. (A) Automatic shape clustering with K=3, 4, 5, or 10 clusters. (B) Manual clustering with K=3, 6, 9, or 12 clusters. Performance of a single classifier approach is given for comparison.

varied within the values of $\{3, 4, 5, 10\}$. Example clustering results are shown in Figure 4.4. For the manual clustering approach, 12 pose clusters have been defined as shown in Figure 4.5. The number of clusters was varied by merging pose clusters, yielding 9, 6, or 3 pose clusters.

ROC results obtained with the proposed pose-specific mixture-of-experts approach using probabilistic cluster assignments are shown in Figure 4.6. The ROC performance of a single classifier approach is given for comparison. Remarkably, all mixture-of-experts variants improve upon the single classifier approach. Best results are obtained with $K = 5$ and $K = 6$ for automatic and manual clustering, respectively. The perhaps surprising result is that both clustering variants perform almost identically. Performance variations for different numbers of clusters are similar, and the best ROC results of each variant nearly coincide. This effect is examined in further experiments below.

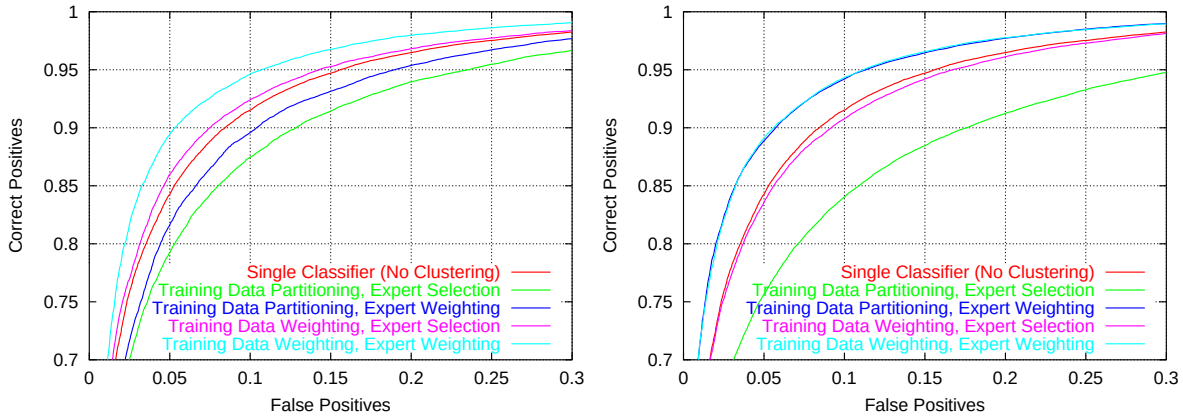


Figure 4.7: Cluster Association Experiments: Experimental comparison of probabilistic clustering vs. hard cluster assignments applied individually to training data and test data. (A) Automatic shape clustering approach, K=5. (B) Manual clustering approach, K=6.

4.4.3 Cluster Association

The assumption of the necessity of probabilistic cluster assignments is to be validated by replacing the cluster membership weights by a hard partitioning. For the pedestrian training set, a hard partitioning is given by the pose clustering, i.e., the training set of expert k consists of the texture patterns corresponding to shape cluster $\mathcal{C}_k$. For the non-pedestrian training data, and for all test data, samples are assigned to the cluster of maximum posterior probability.

Figure 4.7 shows ROC results for both pose clustering variants (automatic and manual). "Expert Selection" refers to the hard partitioning of the test set, "Expert Weighting" denotes probabilistic assignments. Except in the case of manual training data clustering, all hard partitioning variants show degraded performance compared to their probabilistic counterparts. Partitioning of training data even deteriorates performance below that of a single classifier approach (green and blue curve vs. red curve in subfigure A). However, no performance degradation is observed if (nearly) error-free cluster assignments are obtained manually (blue vs. cyan curve in subfigure B). Obviously, probabilistic cluster assignments effectively compensate for errors in shape matching or shape clustering, the latter being particularly interfering in conjunction with reduced training sample sizes. Another positive effect of the probabilistic combination of the experts is a performance gain by a reduction of error variance, i.e., errors of one expert can be compensated for by the others, as will be shown below.

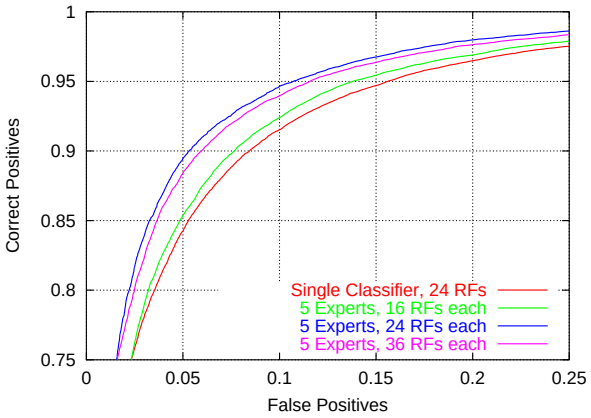
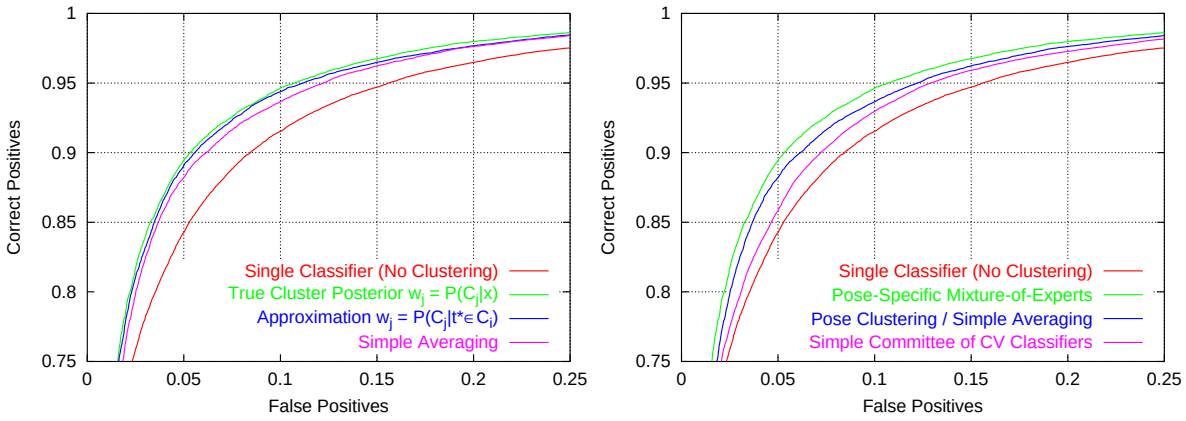


Figure 4.8: Variation of expert complexity and comparison to best performing single classifier.



(A) (B)

Figure 4.9: Experiments with clustering techniques and cluster association. (A) Approximation of expert weighting vs. true weights and simple averaging. (B) Pose clustering (blue and green) vs. a committee of cross-validation experts, each trained on 4/5 of the original training set (magenta). Experiments conducted with 5 experts and automatic shape clustering.

4.4.4 Validation of Design Choices

An important issue in creating a mixture-of-experts architecture involves the complexity of the local experts. For the neural networks with local receptive fields used here, classifier complexity is mainly defined by the number of receptive field branches. Figure 4.8 shows ROC performance for three different numbers used in the experts in comparison to the best performing single classifier (24 receptive field branches). Adaptation of expert complexity to the reduced appearance variance of each pose cluster, i.e., reducing expert complexity, clearly degrades performance. Instead, optimal mixture performance is achieved with experts of relatively high complexity that result in low error bias but high error variance, the latter being reduced by the averaging expert combination.

The effect of approximating the expert weighting is analyzed in Figure 4.9(A). Surprisingly, replacing the true weighting of Eq. (4.9) with the approximation of Eq. (4.15) results in no performance difference. Even simple averaging of the expert results performs only slightly inferior to the mixture approach.

Finally, the concept of pose-specific experts is validated experimentally by a comparison to a simple committee of classifiers [73] which was trained on random subsets of the training data and uses simple averaging to combine the individual results. We chose to use 5 experts, each trained on 4/5 of the original training data in a cross-validation manner. ROC results in Figure 4.9(B) show that this simple committee does not reach the performance obtained by pose clustering, even if a simple averaging of the experts is used for the latter. However, a slight performance gain over the single classifier is achieved.

4.4.5 Shape Normalization

Shape normalization requires accurate point correspondences between the shapes of a pose cluster. Experiments are therefore based on the MPDM shape model that was built from a manual clustering of the training set and manually labeled body part positions. The number of pose clusters was set to 12.

Figure 4.10 shows ROC results with and without background masking, and a comparison to a mixture-of-experts architecture without shape normalization. No performance gain could be achieved without background masking. Presumably, inaccuracies in shape normalization due to errors in shape matching spoiled the benefit of a reduced shape variability. Masking out the background region proves beneficial to the ROC performance and outperforms the corresponding mixture-of-experts architecture without shape normalization by about 10–20%. However, this performance gain was paid by the additional processing cost for shape normalization of about 10ms per example and pose cluster on a 3.2GHz Pentium PC, compared to about 2ms consumed by the NN/LRF classifier.

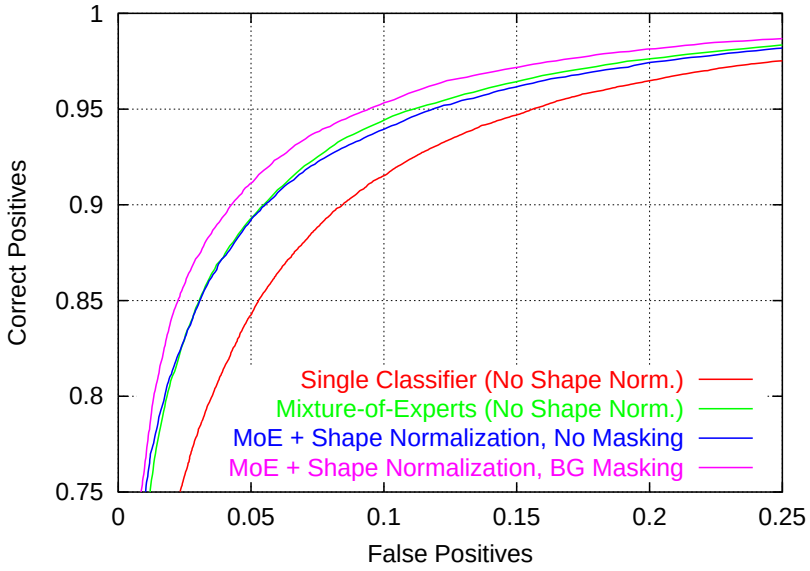


Figure 4.10: ROC results of shape normalization experiments, compared to mixture and single classifier approaches without shape normalization. A manual partitioning of the training examples into K=12 clusters was used.

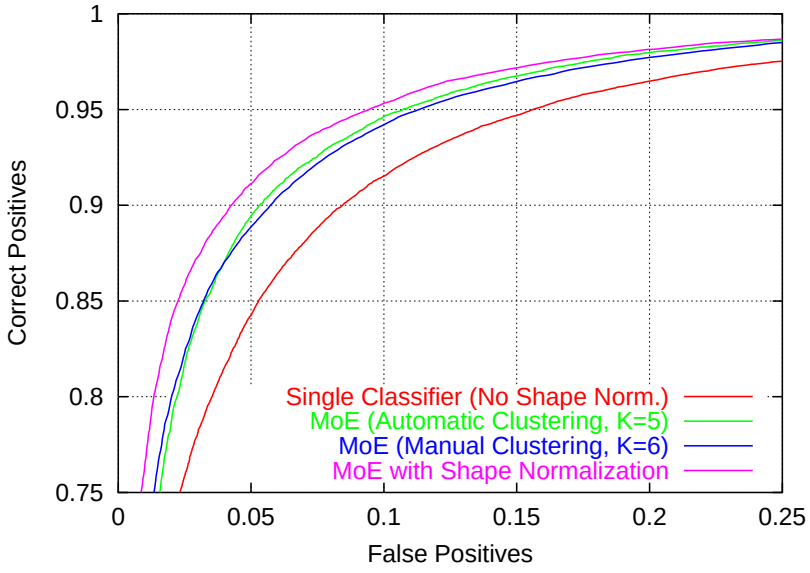


Figure 4.11: Summary of shape-texture integration results. The shape normalization variant involves manual pose clustering with $K = 12$ clusters.

4.4.6 Results

Figure 4.11 summarizes the results of this Chapter obtained by integrating shape information into texture classification. ROC results of three mixture-of-experts approaches are shown in comparison to that of a single global classifier: mixture-of-experts with manual and automatic pose clustering, and mixtures-of-experts in combination with shape normalization (manual clustering, $K = 12$). All three approaches significantly improve upon the single global classifier, with a slight advantage of the shape normalization approach. For fixed detection rates, the number of false positives was reduced by about 30–40% by the mixture-of-experts approaches, and by a further 10–20% when using shape normalization.

Combining the above results, we notice two main aspects of the performance gain attained from shape information. The first one is the creation of diverse experts via pose clustering that underlies our mixture-of-experts approaches. In contrast, the exact choice of the combination rule appears to be less important, see Figure 4.9. The second one is the usage of contour information for masking out background pixels.

The drawback is increased processing cost. In addition to replacing a single classifier by K experts, each of about the same processing costs as the single classifier, shape matching and (optionally) shape normalization must be performed for each input pattern. An application of the proposed approach is, hence, most appealing for detection systems that precede texture classification by a shape matching step for object localization. Two such examples are described in the following two chapters.

5 Cascade Optimization

Many object recognition applications require computationally efficient algorithms, either because real-time processing is required or because of limited computing resources. Such demands are typically accomplished by cascade-like architectures that aim to prune out irrelevant image regions early in the processing chain, and to apply powerful but computationally expensive classifiers to a small number of regions of interest (ROIs) only. While early approaches have consisted of a simple 2-stage cascade of ROI generation and ROI classification, more complex systems with a higher number of cascade stages have been developed recently. Those cascade systems may either be homogenous such as the popular cascade of AdaBoost classifiers by Viola and Jones [99], or may consist of complementary system modules such as the multi-cue system proposed by Gavrila and Munder [36].

Assume that the individual modules have already been generated separately, e.g., by learning a classification rule on a training set or by hand-crafting a detection algorithm. We here turn to the problem on how to adjust the parameters of the system modules so as to optimize the overall system performance. Module parameters typically involve thresholds applied to some decision function that control the percentage of ROIs passed to the next cascade stage, but other parameters might affect system performance as well.

Three performance measures are considered for system optimization. Beside the inevitable trade-off between detection rate and false positive rate, processing time is of interest as well. Ignoring the latter could lead to an optimization outcome where all work is shifted to the single most powerful but computationally most expensive module, so that the resulting system would be impractically slow. By integrating a processing time constraint into the optimization process, efficient pruning of the search space by the early (usually fast) cascade stages is ensured.

We distinguish two problem variants. First, a classification problem is considered where a cascade classifier assigns an input pattern to either the target or the the non-target class. This decision is compared to the true class label. Secondly, we consider the somewhat broader problem of object detection which involves classification and localization. A cascade detector is fed with an input image and yields a set of predicted target object positions, e.g., by shifting a classifier over the input image. Performance evaluation then involves matching the set of system detections to the set of true objects positions (the "ground truth").

After formalizing the optimization problem, two parameter optimization approaches are presented below and reasons for their combined use are given. In addition, an earlier approach based on the assumption of module independence is shortly reviewed. The

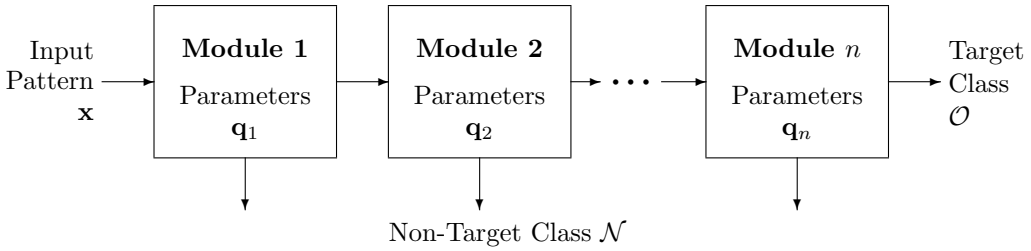


Figure 5.1: A general cascade architecture. Vectors $\mathbf{q}_i$ may include thresholds and other module parameters.

methods are first evaluated using synthetic data sets, before we present a complex real-world cascade system and apply our optimization techniques.

5.1 Problem Formulation

Given a cascade system as shown in Figure 5.1, let n denote the number of modules under consideration, $\mathbf{q}_i$ a parameter vector of module i chosen from the set $\mathcal{Q}_i$ of admissible parameter settings. Let $\mathbf{q}_{1:i} = (\mathbf{q}_1, \ldots, \mathbf{q}_i)$ and $\mathcal{Q}_{1:i} = \mathcal{Q}_1 \times \ldots \times \mathcal{Q}_i$ respectively denote the concatenated parameter vector and parameter set of the first i cascade modules. Finally, let $F_{1:i}(\mathbf{q}_{1:i})$, $H_{1:i}(\mathbf{q}_{1:i})$, and $C_{1:i}(\mathbf{q}_{1:i})$ denote the three performance metrics false positive rate, detection (or hit) rate, and processing cost, respectively, obtained when the first i system modules are considered with parameter setting $\mathbf{q}_{1:i}$. The performance of the overall system is specified by $i = n$.

The optimization objective now is to find the parameter setting $\mathbf{q}^\star_{1:n} \in \mathcal{Q}_{1:n}$ that optimizes system performance. In order for the optimization problem to become tractable, the multi-objective problem is first turned into a single-objective problem by selecting one performance measure as the optimization objective and fixating the other two as optimization constraints. With regard to our prospective application, we choose to minimize the false positive rate given a user-defined minimum detection rate $H^\star$ and maximum processing cost $C^\star$. Formally, we seek to find $\mathbf{q}^\star_{1:n}$ such that

$$F_{1:n}(\mathbf{q}^\star_{1:n}) = \min_{\mathbf{q}_{1:n} \in \mathcal{Q}_{1:n}} F_{1:n}(\mathbf{q}_{1:n}) \;\text{ subject to }\; \begin{cases} H_{1:n}(\mathbf{q}_{1:n}) \geq H^\star\,, \\ C_{1:n}(\mathbf{q}_{1:n}) \leq C^\star\,. \end{cases} \tag{5.1}$$

Equations of the three performance measures differ for the two problem variants, as follows.

Cascade Classification Problem

A cascade classifier involves a sequence of decision functions $d_i : \mathbb{R}^k \times \mathcal{Q}_i \to \{0, 1\}$ that each assign an input pattern $\mathbf{x} \in \mathbb{R}^k$ to either the target class $\mathcal{O}$ if $d_i(\mathbf{x}; \mathbf{q}_i) = 1$, or

to the non-target class $\mathcal{N}$ otherwise, subject to the parameter setting $\mathbf{q}_i$. The overall classification result is class $\mathcal{O}$ if all cascade stages agree to that decision, and class $\mathcal{N}$ otherwise:

$$\text{Assign } \mathbf{x} \text{ to class } \left\{ \begin{array}{c} \mathcal{O} \\ \mathcal{N} \end{array} \right\} \text{ if } D_{1:n}(\mathbf{x}; \mathbf{q}_{1:n}) = \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\}, \tag{5.2}$$

where $D_{1:n}(\mathbf{x}; \mathbf{q}_{1:n}) = \prod_{i=1}^{n} d_i(\mathbf{x}; \mathbf{q}_i)$.

The false positive rate and the detection rate for the cascade classification problem[1] are then given by

$$F^{cl}_{1:n}(\mathbf{q}_{1:n}) = \Pr[\, D_{1:n}(\mathbf{x}; \mathbf{q}_{1:n}) = 1 \,|\, \mathcal{N}\,], \tag{5.3}$$

$$H^{cl}_{1:n}(\mathbf{q}_{1:n}) = \Pr[\, D_{1:n}(\mathbf{x}; \mathbf{q}_{1:n}) = 1 \,|\, \mathcal{O}\,]. \tag{5.4}$$

For modeling the processing time constraint, we assume that the processing cost for one evaluation of decision function $d_i(\mathbf{x}; \mathbf{q}_i)$ depends on the parameter setting $\mathbf{q}_i$ but is independent of input pattern $\mathbf{x}$. Let $c_i(\mathbf{q}_i)$ denote this processing cost. Decision function d_i is only evaluated if the outcome of the preceding cascade stages $D_{1:i-1}(\mathbf{x}; \mathbf{q}_{1:i-1}) = \prod_{i=1}^{i-1} d_i(\mathbf{x}; \mathbf{q}_i)$ equals 1. Hence, the expected processing cost of cascade stage i is given by

$$\begin{aligned} C^{cl}_i(\mathbf{q}_{1:i}) &= \mathrm{E}[\, D_{1:i-1}(\mathbf{x}; \mathbf{q}_{1:i-1})\, c_i(\mathbf{q}_i)\,] \\ &= \Pr[\, D_{1:i-1}(\mathbf{x}; \mathbf{q}_{1:i-1}) = 1\,]\, c_i(\mathbf{q}_i)\,. \end{aligned} \tag{5.5}$$

Summation yields the processing cost of the entire cascade classifier:

$$C^{cl}_{1:n}(\mathbf{q}_{1:n}) = c_1(\mathbf{q}_1) + \sum_{i=2}^{n} C^{cl}_i(\mathbf{q}_{1:i})\,. \tag{5.6}$$

In practice, these performance measures will be replaced by their empirical estimates obtained on a training sample $\mathcal{S}$,

$$F^{cl}_{1:n}(\mathbf{q}_{1:n}) = \frac{1}{|\mathcal{S}_\mathcal{N}|} \sum_{\mathbf{x} \in \mathcal{S}_\mathcal{N}} D_{1:n}(\mathbf{x}; \mathbf{q}_{1:n})\,, \tag{5.7}$$

$$H^{cl}_{1:n}(\mathbf{q}_{1:n}) = \frac{1}{|\mathcal{S}_\mathcal{O}|} \sum_{\mathbf{x} \in \mathcal{S}_\mathcal{O}} D_{1:n}(\mathbf{x}; \mathbf{q}_{1:n})\,, \tag{5.8}$$

$$C^{cl}_{1:n}(\mathbf{q}_{1:n}) = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \Big(c_1(\mathbf{q}_1) + \sum_{i=2}^{n} D_{1:i-1}(\mathbf{x}; \mathbf{q}_{1:i-1}) c_i(\mathbf{q}_i) \Big)\,, \tag{5.9}$$

where $\mathcal{S}_\mathcal{O}$ and $\mathcal{S}_\mathcal{N}$ denote the training subsets of target and non-target examples, respectively, and $|\mathcal{S}|$ is the cardinality of the finite set $\mathcal{S}$.

[1] Performance measures of the cascade classification problem are indicated by superscript *cl*, and those of the cascade detection problem introduced in the next paragraph by *det*. The superscript is dropped in descriptions that apply to both of the two problem variants.

Cascade Detection Problem

A cascade detector aims to detect and localize a set of target objects in video images. Let $\mathcal{G}$ denote the set of true objects appearing in these images, and let $\mathcal{Z}(\mathbf{q})$ denote the set of object detections emitted by the cascade detector for parameter setting $\mathbf{q}$. A user-defined indicator function M specifies whether entries from ground truth and system output match, i.e., whether a system output object $\mathbf{z} \in \mathcal{Z}(\mathbf{q})$ is a valid detection of the ground truth object $\mathbf{g} \in \mathcal{G}$.

$$M(\mathbf{g}, \mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{g} \text{ and } \mathbf{z} \text{ match,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

The false positive rate is then given by the number of system output objects without a matching ground truth object:[2]

$$F^{det}_{1:n}(\mathbf{q}_{1:n}) = \left| \left\{ \mathbf{z} \in \mathcal{Z}(\mathbf{q}_{1:n}) \,:\, \sum_{\mathbf{g} \in \mathcal{G}} M(\mathbf{g}, \mathbf{z}) = 0 \right\} \right| . \tag{5.11}$$

The subscript $1:n$ specifies entities that refer to the full cascade of modules $1, \ldots, n$, as above. Analogously, the detection rate specifies the number of ground truth objects that have a matching system object:

$$H^{det}_{1:n}(\mathbf{q}_{1:n}) = \left| \left\{ \mathbf{g} \in \mathcal{G} \,:\, \sum_{\mathbf{z} \in \mathcal{Z}(\mathbf{q}_{1:n})} M(\mathbf{g}, \mathbf{z}) > 0 \right\} \right| . \tag{5.12}$$

As above, the processing time of each cascade module is modeled as a linear function of the number of input objects, and accumulated to yield the overall processing cost:

$$C^{det}_{1:n}(\mathbf{q}_{1:n}) = \left| \mathcal{Z}_0 \right| c_1(\mathbf{q}_1) \;+\; \sum_{i=2}^{n} \left| \mathcal{Z}_{1:i-1}(\mathbf{q}_{1:i-1}) \right| c_i(\mathbf{q}_i) \,, \tag{5.13}$$

where $\mathcal{Z}_0$ denotes the set of ROIs to be processed by the first cascade stage.

Remarks

Many (cascade) detectors are based on a sliding window approach, i.e., a (cascade) classifier is shifted over the input images. In that case, the only difference between the two problem formulations is the definition of the detection rate. The detection problem allows n-to-m associations between ground truth and system objects, whereas the classification problem does not. However, our formulation of the detection problem is more general in that it applies to sophisticated coarse-to-fine search strategies as well.

Unless otherwise stated, the optimization approaches considered below apply to both problem formulations, by "plugging-in" the respective performance measure equations into the optimization algorithm.

[2] The false positive *rate* is commonly given per frame. This factor is constant w.r.t. to the system parameters and therefore neglected here. The same applies to the detection rate and processing time.

5.2 Sequential ROC Optimization

The trade-off between detection rate and false positive rate is typically depicted in ROC space. Varying parameter settings over a discrete, finite set $\mathcal{Q}$ leads to a cloud of ROC points, as illustrated in Figure 5.2. After filtering this set of ROC points to meet additional optimization constraints, optimal ROC points are given by its frontier or *Pareto optimal* [9] subset. Each ROC point along the frontier represents an optimal false positive rate and corresponding parameter setting for a particular choice of the desired detection rate. However, an exhaustive search over all parameter settings $\mathcal{Q}$ is practically infeasible even for moderate numbers of parameters, discrete settings, and training sample sizes.

The idea of our sequential ROC optimization method is to tackle the high cardinality of the parameter space by successively integrating the cascade modules into the optimization process. Given an optimal frontier subset of ROC points for a cascade of modules $1 \dots i-1$, we include module i and seek for the optimal subset of ROC points for a cascade of module $1 \dots i$. This search can be simplified if each of the new optimal ROC points of interest is also optimal if only cascade modules $1 \dots i-1$ are considered, i.e., if none of the previously discarded non-optimal parameter settings for modules $1 \dots i-1$ can lead to an optimal ROC point for modules $1 \dots i$. Then, the search space can be restricted to the preceding optimal parameter set and a variation of the parameters of the currently considered module i.

The final optimization result is then given by searching the overall optimal ROC subset for the point of minimum false positive rate that meets the detection rate constraint. The processing time constraint is incorporated analogously by extending the ROC space to a three-dimensional objective space.

In the subsections below, the sequential optimization approach is formalized and implementation details are given, experimental evaluations are given in Sections 5.5 and 5.6.

5.2.1 Sequential Optimization Algorithm

Consider a discrete, finite set of parameter settings $\mathcal{Q}$. Each setting $\mathbf{q} \in \mathcal{Q}$ leads to a point $(F(\mathbf{q}), H(\mathbf{q}))$ in ROC space plus associated processing cost $C(\mathbf{q})$. We aim to find the subset $\mathcal{Q}^\star \subset \mathcal{Q}$ of parameter settings that yield the dominating or *Pareto optimal* [9] ROC points along the frontier, i.e., those that are not dominated by another ROC point in terms of all three performance measures. Formally, this subset is defined as

$$\mathcal{Q}^\star \stackrel{def}{=} \{\mathbf{q} \in \mathcal{Q} \,|\, \forall_{\mathbf{q}' \in \mathcal{Q}} F(\mathbf{q}') \geq F(\mathbf{q}) \vee H(\mathbf{q}') \leq H(\mathbf{q}) \vee C(\mathbf{q}') \geq C(\mathbf{q})\} \,. \tag{5.14}$$

We say that $\mathcal{Q}^\star$ is the *optimal subset* of $\mathcal{Q}$, and all elements of $\mathcal{Q}^\star$ are called *optimal*.

In order to restrict the parameter search space $\mathcal{Q}$, we need to make the following recursive assumption: Each parameter vector $\mathbf{q}_{1:i} \in \mathcal{Q}^\star_{1:i}$ optimal for a cascade of modules

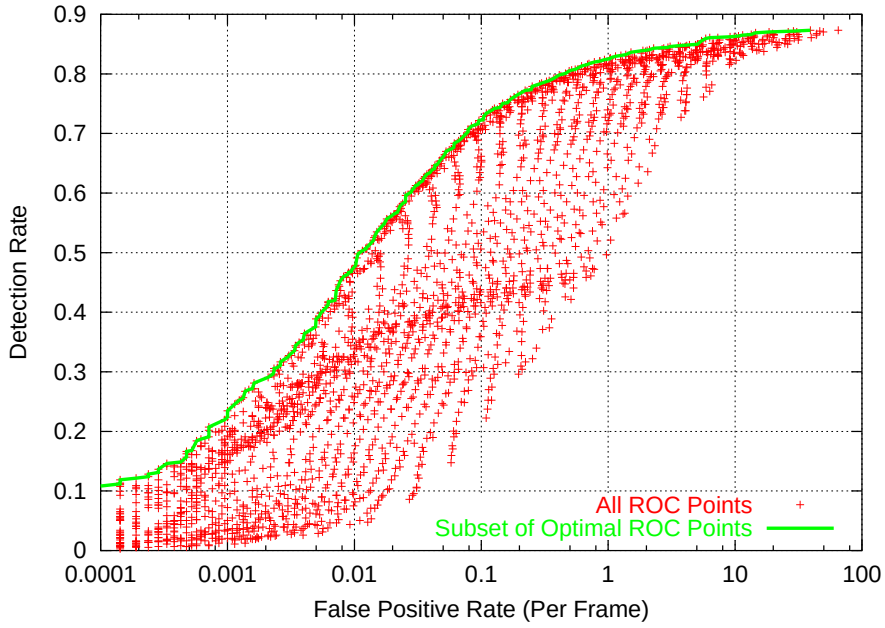


Figure 5.2: Illustration of an optimal subset of ROC points: A cloud of ROC points has been computed, one for each possible parameter setting, as shown by red the crosses. The subset of optimal ROC points along the frontier is depicted by the green curve.

Given the sets of feasible parameter settings $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$, the minimum detection rate $H^\star$, and the maximum processing cost $C^\star$.
Initialize $\mathcal{Q}'_{1:1} = \mathcal{Q}_1$.
For $i = 1, \ldots, n$:

- For each parameter setting $\mathbf{q}_{1:i} \in \mathcal{Q}'_{1:i}$, compute the false positive rate $F_{1:i}(\mathbf{q}_{1:i})$, the detection rate $H_{1:i}(\mathbf{q}_{1:i})$, and the processing cost $C_{1:i}(\mathbf{q}_{1:i})$.
- Determine the subset of optimal parameter vectors:

$$\mathcal{Q}^\star_{1:i} = \{\mathbf{q} \in \mathcal{Q}'_{1:i} \,|\, \forall_{\mathbf{q}' \in \mathcal{Q}'_{1:i}} F_{1:i}(\mathbf{q}') \geq F_{1:i}(\mathbf{q}) \vee H_{1:i}(\mathbf{q}') \leq H_{1:i}(\mathbf{q}) \vee C_{1:i}(\mathbf{q}') \geq C_{1:i}(\mathbf{q})\}\,.$$

- Let $\mathcal{Q}'_{1:i+1} = \mathcal{Q}^\star_{1:i} \times \mathcal{Q}_{i+1}$.
- Continue with $i \leftarrow i + 1$

Determine the final optimization result $\mathbf{q}^\star_{1:n}$ that satisfies

$$F_{1:n}(\mathbf{q}^\star_{1:n}) = \min_{\mathbf{q}_{1:n} \in \mathcal{Q}^\star_{1:n}} F_{1:n}(\mathbf{q}_{1:n}) \;\text{ subject to }\; \begin{cases} H_{1:n}(\mathbf{q}_{1:n}) \geq H^\star \\ C_{1:n}(\mathbf{q}_{1:n}) \leq C^\star \end{cases}$$

by a search over the optimal subset $\mathcal{Q}^\star_{1:n}$.

Figure 5.3: Sequential optimization procedure

$1, \ldots, i$ is also optimal if only modules $1, \ldots, i-1$ are considered. That is,

$$\mathbf{q}_{1:i} = (\mathbf{q}_{1:i-1}, \mathbf{q}_i) \in \mathcal{Q}^\star_{1:i} \;\Rightarrow\; \mathbf{q}_{1:i-1} \in \mathcal{Q}^\star_{1:i-1}\,, \tag{5.15}$$

for $i = 2, \ldots, n$.

Consequently, the overall optimal subset $\mathcal{Q}^\star_{1:n}$ can be found by successively computing $\mathcal{Q}^\star_{1:1}$, $\mathcal{Q}^\star_{1:2}$, $\ldots$, $\mathcal{Q}^\star_{1:n}$, where in each step, the parameter search space $\mathcal{Q}'_{1:i}$ is restricted to contain the preceding solution:

$$\begin{aligned} \mathcal{Q}'_{1:i} &= \mathcal{Q}^\star_{1:i-1} \times \mathcal{Q}_i\,, \\ \mathcal{Q}^\star_{1:i} &= \{\mathbf{q} \in \mathcal{Q}'_{1:i} \,|\, \forall_{\mathbf{q}' \in \mathcal{Q}'_{1:i}} F_{1:i}(\mathbf{q}') \geq F_{1:i}(\mathbf{q}) \vee H_{1:i}(\mathbf{q}') \leq H_{1:i}(\mathbf{q}) \vee C_{1:i}(\mathbf{q}') \geq C_{1:i}(\mathbf{q})\}\,. \end{aligned} \tag{5.16}$$

The resulting sequential optimization algorithm is listed in Figure 5.3.

5.2.2 Discussion

It is easy to show that the sequentiality assumption in Eq. (5.15) is met for a cascade classifier of independent cascade modules if the processing cost constraint is dropped.

Module independence is given if the output $d_i(\mathbf{x}; \mathbf{q}_i)$ of module i does not depend on the output of the other modules so that

$$\begin{aligned} F^{cl}_{1:n}(\mathbf{q}_{1:n}) &= \Pr[\, D_{1:n}(\mathbf{x}; \mathbf{q}_{1:n}) = 1 \,|\, \mathcal{N}] \\ &= \prod_{i=1}^{n} \Pr[\, d_i(\mathbf{x}; \mathbf{q}_i) = 1 \,|\, \mathcal{N}] \stackrel{def}{=} \prod_{i=1}^{n} f_i(\mathbf{q}_i) \,, \end{aligned} \tag{5.17}$$

and accordingly

$$H^{cl}_{1:n}(\mathbf{q}_{1:n}) = \prod_{i=1}^{n} \Pr[\, d_i(\mathbf{x}; \mathbf{q}_i) = 1 \,|\, \mathcal{O}\,] \stackrel{def}{=} \prod_{i=1}^{n} h_i(\mathbf{q}_i) \,. \tag{5.18}$$

Then,

$$\begin{aligned} &\mathbf{q}_{1:i} = (\mathbf{q}_{1:i-1}, \mathbf{q}_i) \in \mathcal{Q}^{\star}_{1:i} \\ &\Rightarrow \forall_{\mathbf{q}'_{1:i-1} \in \mathcal{Q}_{1:i-1}} : F^{cl}_{1:i}(\mathbf{q}'_{1:i-1}, \mathbf{q}_i) \geq F^{cl}_{1:i}(\mathbf{q}_{1:i-1}, \mathbf{q}_i) \vee H^{cl}_{1:i}(\mathbf{q}'_{1:i-1}, \mathbf{q}_i) \leq H^{cl}_{1:i}(\mathbf{q}_{1:i-1}, \mathbf{q}_i) \\ &\Rightarrow \forall_{\mathbf{q}'_{1:i-1} \in \mathcal{Q}_{1:i-1}} : F^{cl}_{1:i-1}(\mathbf{q}'_{1:i-1}) \geq F^{cl}_{1:i-1}(\mathbf{q}_{1:i-1}) \vee H^{cl}_{1:i-1}(\mathbf{q}'_{1:i-1}) \leq H^{cl}_{1:i-1}(\mathbf{q}_{1:i-1}) \\ &\Rightarrow \mathbf{q}_{1:i-1} \in \mathcal{Q}^{\star}_{1:i-1} \,. \end{aligned}$$

In other words, the sequential optimization procedure of Eq. (5.16) applied to a cascade classification problem without a processing cost constraint is provably optimal if the cascade nodes are independent. For many practical applications, cascade nodes are correlated so that this procedure is no longer guaranteed to find the global optimum. However, there is experimental evidence that often, the solution such found is close to optimal. Examples are given in Sections 5.5 and 5.6. An optimization approach that directly builds upon the independence assumption is discussed in Section 5.4.

The sequential optimization approach provides a major speed-up compared to a brute-force search over the set of all possible parameter settings. If each of the n modules has K discrete combinations of parameter values, $K = |\mathcal{Q}_i|$, then a brute-force search is of complexity $O(K^n)$. If we assume that the size of the optimal subsets $\mathcal{Q}^{\star}_{1:i}$ is of the same order $O(K)$, then the computational complexity of the sequential optimization procedure is only $O((n-1)K^2)$. Since typically, $K \gg n$, a speed-up is achieved for $n \geq 3$. Furthermore, the complexity is linear in n, so that this optimization approach is applicable to large-scale cascade systems consisting of a large number of modules.

The sequential optimization procedure is particularly suitable for optimizing threshold parameters, i.e., if $\mathbf{q}_i$ represents a vector of thresholds to be applied to the output of module i. In that case, performance measures for varying values of $\mathbf{q}_i$ (but for a fixed setting $\mathbf{q}_{1:i-1}$) can be calculated by a single evaluation of the cascade system by means of logging the output values of module i and applying the thresholds $\mathbf{q}_i$ offline. This further speeds up the optimization procedure.

Module parameters are typically continuous, not discrete, though often, reasonable upper and lower bounds can be given. The sequential optimization procedure can then

only be applied after a discretization of the parameter ranges, which leads to a trade-off between the resolution of the discretization and the required computing resources. A discussion of this issue is skipped here. Instead, we propose a two-step approach of first applying the sequential optimization to a coarse discretization of the parameter ranges, and then applying the generic optimization proposed in Section 5.3.

5.3 Generic Optimization Techniques

Constrained non-linear optimization techniques are a large active field of research, and many different approaches were proposed in the past. Yet the problem at hand is challenging due to a number of unfavorable properties:

- Both the objective function and the constraint functions are non-convex and non-smooth.
- Since the probability density functions in Eqs. (5.3)-(5.6) are unknown, empirical estimations based on a finite sample set are used instead. These estimates are piecewise constant "step functions".
- For the same reason, derivatives of the objective function and the constraint functions are not available, and numerical approximations are difficult because of the finite sample size effects.
- Evaluations of the objective function and the constraint functions are relatively costly because it takes a run of the full cascade system over a potentially large data set.

For these reasons, simple gradient-descent methods fail on the given problem. Recently, however, a constrained non-linear optimization technique named "CONDOR"[3] [3] was proposed that fits into the problem profile. A detailed description is given in [3]; below, a rough sketch of the optimization method is given that motivates its use for the problem at hand.

The CONDOR optimization technique belongs to the iterative gradient-descent family of optimization methods. Let $\mathbf{x}_k$ denote the current point at iteration k (in our case, the best parameter setting $\mathbf{q}_{1:n}$ found so far), the starting point $\mathbf{x}_1$ being provided by the user. For each iteration, the following steps are performed.

1. Construct a local quadratic model of the objective function F around the current point $\mathbf{x}_k$ from a few function evaluations:

$$F(\mathbf{x}_k + \delta) \approx Q_k(\delta) = F(\mathbf{x}_k) + \mathbf{g}_k^T \delta + \frac{1}{2}\delta^T B_k \delta \,, \tag{5.19}$$

[3]CONDOR stands for "COnstrained, Non-linear, Direct, parallel Optimization using trust Region method for high-computing load function"

where $\mathbf{g}_k$ and B_k denote the current approximations of the gradient and the Hessian of the objective function, respectively. The sample points are drawn from region of radius ρ_k around the current point. An initial radius is given by the user, which is then successively narrowed down (step 3). The number of function evaluations is minimized by reusing previous sampling. Following a heuristic by Powell [76], the validity of the current local model is assessed. New sampling points are only generated if this assessment is negative.

2. Compute an update $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta_k$ that goes to the minimum of the current local model, i.e., that minimizes $Q_k(\delta_k)$. Since Q_k represents a *local* model of F, the update step δ_k is restricted to that local area called the *trust region* [13]. The current trust region is defined by the dynamically adapted maximum radius Δ_k and minimum radius $\rho_k/2$, so that the objective of this step is to find δ_k such that

$$Q_k(\delta_k) = \min_{\delta} Q_k(\delta) \;\text{ subject to }\; \frac{1}{2}\rho_k < \|\delta_k\|_2 < \Delta_k \,, \tag{5.20}$$

and subject to the user-defined constraint functions (in our case, minimum detection rate and maximum processing cost). The solution to this subproblem is found by Sequential Quadratic Programming [69].

3. Update the sampling radius ρ_k and the trust region radius Δ_k. The update heuristic for Δ_k is based on a "degree of agreement" τ_k between the objective function F and its local model Q_k:

$$\tau_k = \frac{F(\mathbf{x}_k) - F(\mathbf{x}_k + \delta_k)}{Q_k(0) - Q_k(\delta_k)} \,.$$

At a high level of agreement (e.g., $\tau_k > 0.9$), Δ_k is increased; if the agreement is poor (e.g., $\tau_k < 0.01$), Δ_k is decreased. In the latter case, the update δ_k is dropped and $\mathbf{x}_{k+1} = \mathbf{x}_k$.

The adaptation heuristic for ρ is based on the update step δ_k. If the step becomes small, $\|\delta_k\|_2 < \rho_k$, then ρ is decreased. If ρ reaches the user-defined minimum ρ_{end}, iterations are terminated.

This algorithm is, under some weak assumptions, guaranteed to find a local minimum, but not necessarily the global one. Two parameters mainly guide the algorithm towards the global minimum: the starting point $\mathbf{x}_1$, and the initial sampling radius ρ_{start}. For our application, the result of the sequential optimization procedure, obtained at a coarse discretization of the parameter range, is used as the starting point.

The most appealing feature of the CONDOR algorithm is the computation of the local model in step 1. The need for an explicit formulation of the gradient of the objective function is eliminated as the gradient and the Hessian are approximated from a few sampling points drawn from an adapted radius ρ, while at the same time, the number

of necessary function evaluations is minimized. The termination radius ρ_{end} is chosen large enough to avoid finite data set (step function) problems.

A MATLAB implementation [4] of the CONDOR algorithm is used for the experiments below.

5.4 Independent Optimization

For a cascade classification problem with independent nodes, Luo [59] presented an optimization approach that directly builds upon the independence property. Reformulate problem (5.1) by considering the logarithm of F and H:

$$\min_{\mathbf{q}_{1:n} \in \mathcal{Q}_{1:n}} \log F_{1:n}(\mathbf{q}_{1:n}) \;\text{ subject to }\; \log H_{1:n}(\mathbf{q}_{1:n}) \geq \log H^\star \,. \tag{5.21}$$

The processing time constraint is not taken into account here. The constrained problem (5.21) is tackled by taking its Lagrange dual function

$$g(\lambda) = \min_{\mathbf{q}_{1:n}} \log F_{1:n}(\mathbf{q}_{1:n}) - \lambda H_{1:n}(\mathbf{q}_{1:n}) + \lambda H^\star \,. \tag{5.22}$$

Substituting the independence properties of Eqs. (5.17) and (5.18) into the above yields

$$g(\lambda) = \sum_{i=1}^{n} g_i(\lambda) + \lambda \log H^\star \,, \;\text{ with }\; g_i(\lambda) = \min_{\mathbf{q}_i} \log f_i(\mathbf{q}_i) - \lambda \log h_i(\mathbf{q}_i) \,. \tag{5.23}$$

The elements g_i are independent and can be computed separately by minimization with respect to $\mathbf{q}_i$. Given that the minimum exists at point $\mathbf{q}_i = \mathbf{q}_i^\star$, the derivate must be zero,

$$\frac{\partial}{\partial \mathbf{q}_i} \left(\log f_i(\mathbf{q}_i^\star) - \lambda \log h_i(\mathbf{q}_i^\star) \right) = 0 \,,$$

which yields the optimality condition

$$\lambda = \frac{\frac{\partial}{\partial \mathbf{q}_i} \log f_i(\mathbf{q}_i^\star)}{\frac{\partial}{\partial \mathbf{q}_i} \log h_i(\mathbf{q}_i^\star)} \,. \tag{5.24}$$

Eq. (5.24) represents a necessary condition for the optimality of $\mathbf{q}_{1:n}^\star$. For a given λ, $\mathbf{q}_{1:n}^\star$ can be found by separately computing optimal parameters $\mathbf{q}_1^\star, \ldots, \mathbf{q}_n^\star$ using Eq. (5.24). This transforms the original, multi-dimensional problem in Eq. (5.1) into the one-dimensional one of finding λ such that F is minimized with respect to the $H^\star$ constraint. The solution to the latter problem is left open here; constrained line-search techniques may apply. For the experiments below, we vary λ within a suitable, discrete, finite subset of $(0, \infty)$, which yields a system ROC curve. The final result is given by the ROC point with minimum false positive rate F that still meets the detection rate constraint $H^\star$.

5.5 Cascade Classification Experiments

This section provides an experimental analysis of the efficacy of each of the three cascade optimization techniques described in Sections 5.2 to 5.4 on cascade classification problems. The first batch of experiments operates on synthetic data that simulate the output of three component classifiers. Relative performance of the classifiers and their correlation can be adjusted exactly, and their effect to the optimization results can be analyzed. Then, we evaluate how the synthetic results can be carried over to a "real" example.

5.5.1 Synthetic Data Sets

We simulate a cascade of three classifiers. For an input pattern, the output g_i of each classifier is supposed to consist of some nominal target value y perturbed by random noise e_i, i.e., $g_i = y + e_i$, for $i = 1, \ldots, 3$. Let $y = 1$ for the object class, and $y = 0$ for the non-object class. We model the error term e_i to be zero-mean Gaussian noise, and allow for a correlation between the classifiers:

$$\mathbf{e} = (e_1, e_2, e_3)^T \sim N(0, \Sigma)\,,$$

where the covariance matrix is furthermore allowed to differ for the object and non-object class.[4] For each experiment, a data set of 25,000 examples per class is drawn from that model.

The parameters q_i to be optimized are the thresholds on the output functions g_i, one for each classifier. No processing time constraint is considered here. All three optimization techniques described above are applied. For the sequential optimization technique, 90 discrete values within the range $[-4, 5]$ are considered for each threshold. The independent optimization technique requires the derivatives of the false positive rate and the detection rate, which are obtained numerically from individual ROC curves. The value λ was varied in the range $[e^{-1}, e^{2.5}]$. Both methods yield a full ROC curve, as plotted in Figure 5.4 (green and magenta curve, respectively). The result of sequential optimization is used as the starting point for the generic optimization method CONDOR, which was run for a number of different minimum detection rates (blue points). For comparison, an exhaustive search over the parameters was performed using the same discretization as for the sequential optimization method (red curve). This curve acts as "ground truth", although the results may not be optimal due to the discretization.

[4] In Figure 5.4, the covariance matrix chosen is given as variances σ_i and correlation coefficients ρ_{ij}, i.e.

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix} \begin{pmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{pmatrix},$$

for the target class $\mathcal{O}$ and the non-target class $\mathcal{N}$.

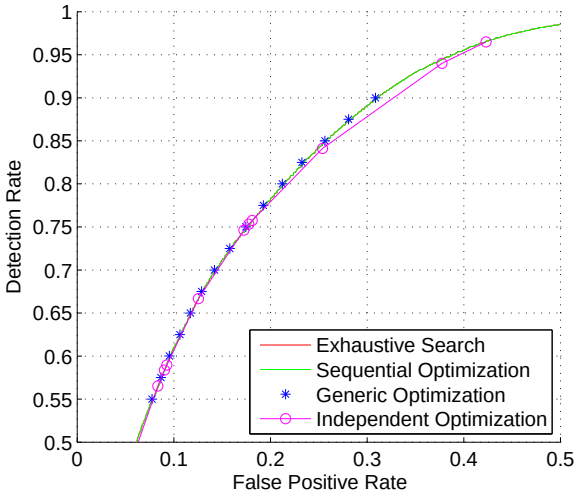
(A) Independent Errors

$\mathcal{O}$: σ_1=1.2, σ_2=1.0, σ_3=0.5

ρ_{12}=ρ_{13}=ρ_{23}=0

$\mathcal{N}$: σ_1=1.0, σ_2=1.2, σ_3=1.5

ρ_{12}=ρ_{13}=ρ_{23}=0



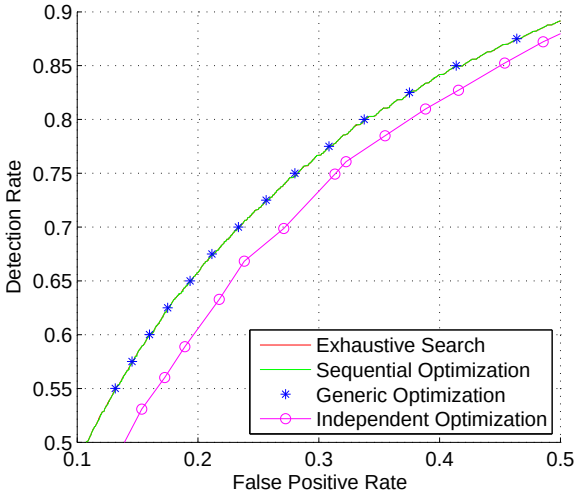
(B) Equal Positive Correlations

$\mathcal{O}$: σ_1=1.2, σ_2=1.0, σ_3=0.8

ρ_{12}=ρ_{13}=ρ_{23}=0.7

$\mathcal{N}$: σ_1=1.2, σ_2=1.0, σ_3=0.8

ρ_{12}=ρ_{13}=ρ_{23}=0.7



(C) Random Positive Correlations

$\mathcal{O}$: σ_1=1.0, σ_2=1.0, σ_3=0.5

ρ_{12}=0, ρ_{13}=0, ρ_{23}=0.9

$\mathcal{N}$: σ_1=1.0, σ_2=1.0, σ_3=1.5

ρ_{12}=0, ρ_{13}=0, ρ_{23}=0.9

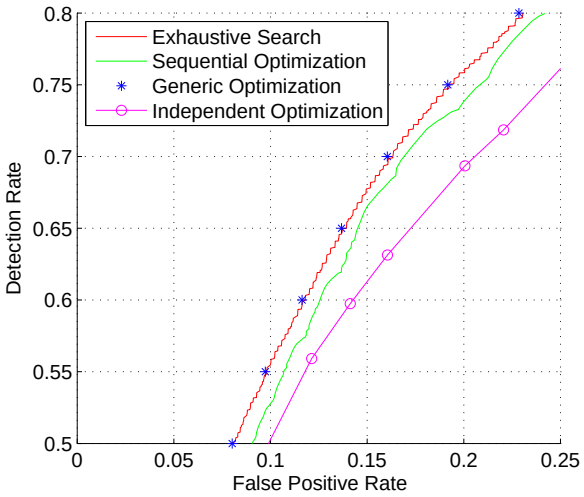


Figure 5.4: Results of cascade optimization on synthetic data sets. See text for details. Note that in subfigures (A) and (B), the red and green curves coincide, which makes the red one invisible.

The first experiment shown in Figure 5.4A assumes independent cascade nodes. Here, both the sequential optimization technique and the independent optimization approach are supposed to yield optimal results, which is verified by the experiment. All four results are identical; remaining differences are due to the discretization.

The next two experiments examine the more common case of positively correlated classifiers. Figure 5.4B shows results for a moderate equal error correlation. The independent optimization method fails here, since the prerequisite of this method is violated. However, the sequential optimization approach still yields correct results, i.e., the ROC curve coincides with that of an exhaustive search. Consequently, the generic optimization, which uses the sequential outcome as a starting point, leads to the same result.

The third experiment is especially designed to trap the sequential optimization approach in a local minimum. The first two classifiers are made independent with equal error variance, so that optimal parameters for a cascade of these two nodes only have $q_1 = q_2$. The third classifier is made independent of the first one, but to replace classifier two by a lower error variance and high correlation ($\rho_{23} = 0.9$). Since the second classifier is redundant, overall optimal result are given by $q_2 \to -\infty$. Results are shown in Figure 5.4C. Although the ROC curve of the sequential optimization method deviates from the optimal one (that of the exhaustive search), it still significantly outperforms the independent optimization approach. Starting from the non-optimal sequential results, the generic optimization approach by the CONDOR method is able to find optimal parameters settings. Differences in the results of the generic optimization and the exhaustive search are caused by the parameter discretization.

5.5.2 Real Example

The second batch of experiments analyzes the optimization techniques on a "real" cascade classifier. We selected the cascade of SVM classifiers on LRF (local receptive fields) features as generated in Section 3.5.2. The first two (of three) training data sets were used for the training of the SVM classifiers, the remaining one serves as the optimization training set. Parameters subject to optimization are the thresholds on the SVM output, one for each cascade node.

Figure 5.5A shows ROC results for each of the optimization techniques as obtained on the optimization training set. The ROC results obtained from an exhaustive search over a very coarse discretization of the parameter range are given for comparison. The sequential optimization method performs slightly better than the exhaustive search. We conclude that the results are optimal, and that the differences are caused by a more fine-grained discretization used for sequential optimization. Consequently, no further performance gain could be achieved by a subsequent CONDOR optimization step. The independent optimization technique does not yield correct results since the classifiers show some correlation, see Figure 5.6.

The parameter settings obtained on the optimization training set were subsequently applied to an independent test set. ROC curves shown in Figure 5.5B confirm that the

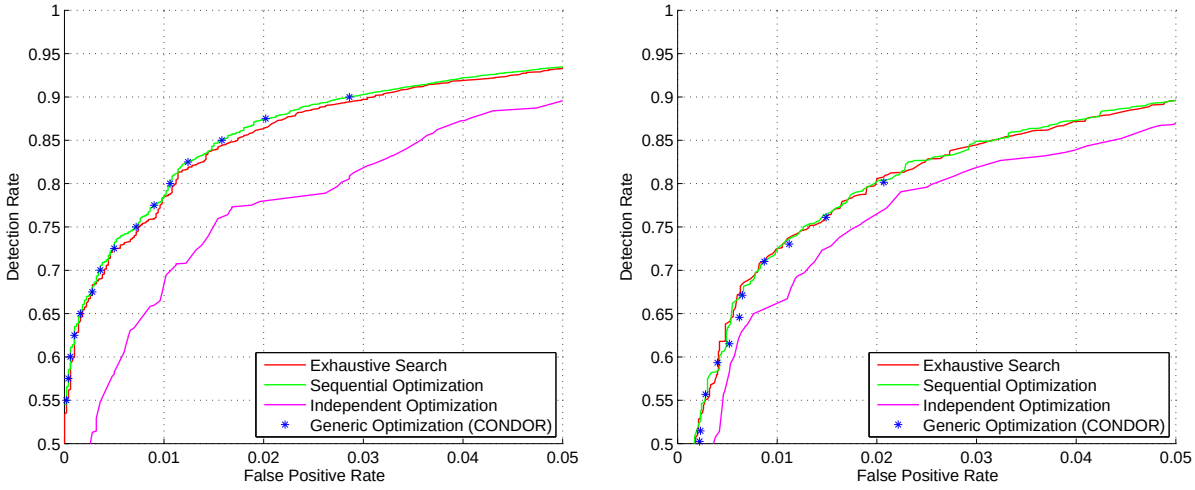


Figure 5.5: Parameter optimization applied to the cascade classifier obtained in Section 3.5.2. The 4-stage SVM on LRFs trained on the first training data set was selected. Optimization was performed on the validation set (A), and resulting thresholds were applied to the test set (B).

$$\mathcal{O}: \begin{pmatrix} 1 & 0.627 & 0.526 & 0.577 \\ & 1 & 0.582 & 0.686 \\ & & 1 & 0.548 \\ & & & 1 \end{pmatrix} \qquad \mathcal{N}: \begin{pmatrix} 1 & 0.450 & 0.151 & 0.029 \\ & 1 & 0.253 & 0.424 \\ & & 1 & 0.263 \\ & & & 1 \end{pmatrix}$$

Figure 5.6: Correlation matrix between the outputs of the four SVM nodes of the cascade classifier used in Figure 5.5, obtained separately for the target class (left) and the non-target class (right). The SVM classifiers show a considerable amount of (positive) correlation, especially on examples of the target class. The reason is the generation of the training data sets. A new set of non-target training examples was created for each classifier, while the same set of target training examples was shared among the SVM classifiers.

optimization outcomes generalize; qualitative results on the training and test sets are the same.

Concluding from the results of both experiments, we observe that the independent optimization approach is sensitive to its prerequisite. Results are degraded as soon as the independence condition is violated. The sequential optimization approach shows a certain amount of robustness to correlations between the cascade nodes. Although optimal results cannot be guaranteed, the outcome is often close to optimal. In cases where the sequential optimization approach is non-optimal, a subsequent application of a generic constrained optimization technique such as CONDOR was observed to lead to the correct results. We therefore propose the combined application of both optimization techniques if computationally viable.

5.6 Pedestrian Recognition Application

We now turn to a real-world application for the detection and tracking of pedestrians from a moving vehicle, called the PROTECTOR system. This application is challenging from a machine vision perspective, since it combines the difficulties of a complex object appearance variability and changeable cluttered background with real-time constraints. The PROTECTOR system involves a cascade of modules, each utilizing complementary visual criteria to successively narrow down the image search space. Four detection modules are considered: sparse stereo-based ROI generation, shape-based detection, texture-based classification, and dense stereo-based object verification. These are complemented by a tracking module.

5.6.1 PROTECTOR System Modules

This section gives short descriptions of the system modules, and lists the parameters considered for optimization. See Figure 5.7. We then apply the cascade optimization techniques to the PROTECTOR system and provide extensive experimental results.

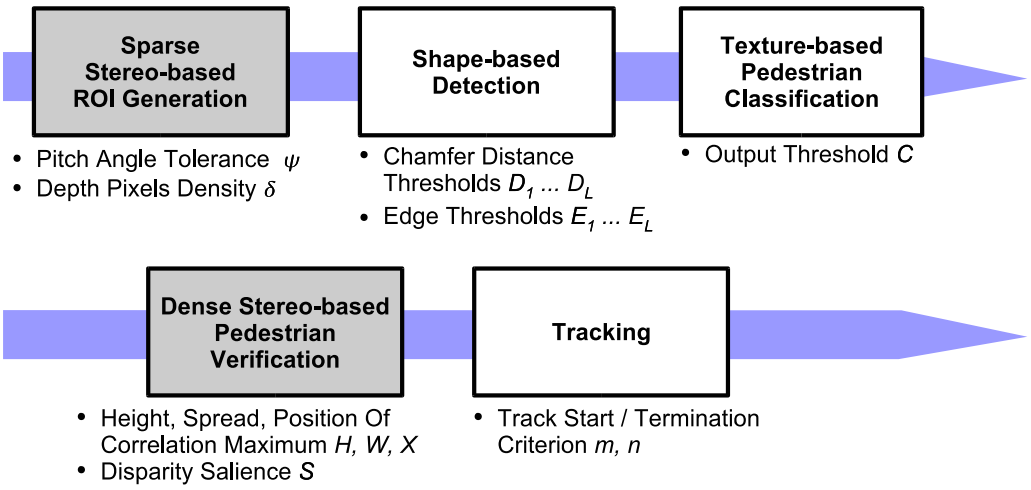


Figure 5.7: Overview of the PROTECTOR system modules. Modules shaded grey depend on stereo imaging. Parameters selected for optimization are listed under the corresponding module.

Sparse Stereo-based ROI Generation

Processing starts with the computation of a disparity map using stereo. First step is the rectification of the left and right camera image using an optimized implementation of Bouguet's method [8]. This improves the epipolar alignment and reduces the effects

of lens distortion away from the optical center. In order to allow real-time processing, we use a feature-based, multi-resolution stereo algorithm developed by [25]; alternate choices would have been possible. The outcome is a relatively sparse disparity map that is subsequently multiplexed into a number of discrete depth ranges.

The resulting binary images are scanned with windows related to minimum and maximum extents of pedestrians, taking into account the ground plane location at a particular depth range and appropriate pitch angle tolerances. The locations where the number of depth features exceeds a percentage of the window area are added to the ROI list for the subsequent shape detection module. The two parameters, pitch angle tolerance ψ and feature density threshold δ control the amount of ROIs passed onto the next stage; they are subject to optimization in the next Section.

Shape-based Detection

This module follows a template matching approach to shape-based pedestrian detection, as described in [37], that builds upon the exemplar-based, hierarchical shape representation introduced in Section 4.1.1. Pedestrians are represented by a set of training shapes which ideally cover the set of object appearances due to transformations (i.e. scale) and intra-class variance (i.e. different pedestrians, different poses). Offline, a template hierarchy is constructed automatically in a bottom-up, level-by-level fashion using clustering. Previous experiments have shown that a three-level hierarchy is suitable for capturing the pedestrian shape distribution, with number of templates nodes decreasing an order of magnitude at successive levels towards the root [37].

Online, matching involves a depth-first traversal of the template tree structure, starting at the root. Each node corresponds to matching a (prototype) template with the image at particular interest locations (i.e. at various template translations). For the locations where the (template size-normalized) chamfer distance measure between template and image is below a user-supplied distance threshold D_l, one computes new interest locations for the children nodes (generated by sampling the local neighborhood on a finer grid of image locations) and adds the children nodes to the list of nodes to be processed. For locations where the distance measure is above this threshold, search does not propagate to the sub-tree; it is this pruning capability that brings large efficiency gains. Following [37], a single distance threshold D_l applies for each level of the hierarchy. An additional parameter E_l governs the edge density that is extracted from the original image at that level, which is the basis for the underlying distance map. The resulting six parameters $D_1, \ldots, D_3$ and $E_1, \ldots, E_3$ are subject to optimization in Section 5.6.3.

Texture-based Pedestrian Classification

Whereas the preceding module uses shape contours to refine and filter the position and pose of candidate pedestrians, the current pattern classification module utilizes the richer set of intensity features to make the distinction pedestrian versus non-pedestrian.

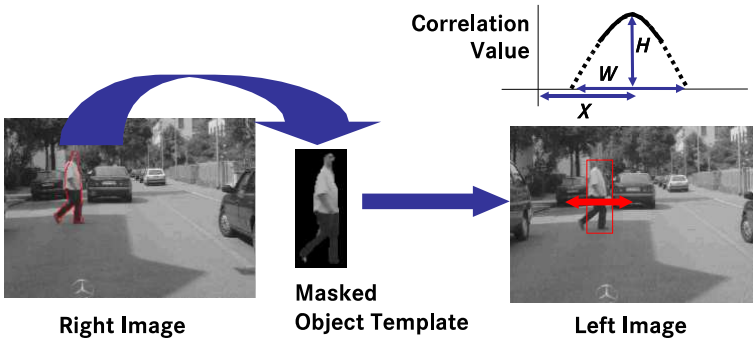


Figure 5.8: Stereo Verification: The shape template masks out background pixels for a dense cross-correlation between both stereo images within a certain disparity search range. A threshold is enforced on both height and spread of the resulting correlation function.

From the many combinations of feature extraction and classification methods evaluated in Section 3, we selected the neural network with local receptive fields as a reasonable compromise between classification performance and processing speed.

Furthermore, we employ the mixture-of-experts architecture introduced in Section 4.2. K pedestrian pose clusters are given by the top-level of the shape template hierarchy constructed above, and one component classifier was generated for each pose cluster. Online, a weighted average of the outputs of each component classifier is computed, where the mixture coefficients are derived from the shape matching results. For efficiency reasons, we use the weight approximation of Eq. (4.15).

ROIs for which the classification result exceeds the user-supplied threshold C are passed to the next system module; the threshold is determined automatically by the cascade optimization step. The number of clusters K was set to 5, which was found optimal in previous experiments.

Dense Stereo-based Pedestrian Verification

The aim at this stage is to filter out false detections which contain an appreciable amount of background. For this, the pedestrian shape template which generated the candidate solution is applied as a mask for a dense cross-correlation with the other stereo image. See Figure 5.8. Cross-correlation is performed within the particular disparity range as determined by the estimated pedestrian depth (i.e. flat world assumption). A second-order polynomial is fitted on the correlation values obtained over this one-dimensional search range. A detection is accepted only if three conditions are met: the maximum of the polynomial is above a threshold H, the normalized "spread" of the polynomial (a measure of localization confidence) is below a threshold W, and the deviation of the position of the correlation maximum is below a threshold X.

A second verification step compares the average disparity within the pedestrian mask

Figure 5.9: Stereo Verification: Examples of removal of background-corrupted detections (accepted solutions shown red, rejected green).

with that in a local neighborhood. The disparity difference is compared to a user-supplied threshold S. For objects that stand out from a more distant background, this disparity difference is positive, while negative values or values near zero indicate false detections caused by background clutter. All four parameters are subject to the optimization step below.

For some intuition in the above, see Figure 5.9. The candidate solutions shown in color may have passed the stereo, shape and texture module in the cascade. Based on the ground plane constraint, however, they are estimated at 10-15m in front of the camera. When cross-correlating with the other stereo image with the corresponding disparities, matching will not produce a high score, since the contained background pixels match best at markedly lower disparity values (i.e. larger distances).

Tracking

Tracking allows us to overcome gaps in detection, to suppress spurious measurements and to obtain trajectory information. Mainly because of computational cost and because shape variations are handled by the detection modules, our tracker is simplified to involve 2.5-D bounding box position (x, y), extent (w, h), and depth (z), as well as their derivatives. The depth of a bounding box is determined from stereo vision using the shape template to mask out background pixels. We use a straightforward α-β tracker to estimate the object state parameters.

To deal with non-trivial data associations (i.e. single-measurement multiple-track assignments or vice versa) in an optimal fashion, we use the classical Hungarian method [54]. It operates on a cost matrix, which is built from the similarity between the prediction of the tracks and the associated measurements. As similarity measure, we use a weighted linear combination of Euclidean distance between object centroids and pairwise shape dissimilarity (the chamfer distance).

A new track is started whenever a new object appears in m successive frames and no active track fits to it. It ends, if the object corresponding to an active track has not been detected in n successive frames. The two associated parameters, m and n are determined automatically by the optimization step.

	Run1	*Run2*
Duration	27 min	24 min
Total images	21053	17390
Images containing pedestrians	1021	855
Pedestrian instances: all / risky	733 / 112	694 / 89
Pedestrian trajectories: all / risky	45 / 17	50 / 10

Table 5.1: Statistics of the two video sequences recorded in urban environment. Pedestrians directly in front of the car (maximum lateral offset from the vehicle medial axis of 1.5m) are considered "risky".

5.6.2 Test Methodology and Data Sets

In order to obtain meaningful results, experiments with the PROTECTOR system are conducted on very large data sets taken in urban traffic environment. Two video sequences (*Run1* and *Run2*) were recorded on the same route through suburbia and inner city of Aachen, Germany. On the route, ten pedestrian "actors" awaited the system, either standing or crossing at various walking speeds, according to a pre-defined choreography (for both runs the same). In addition, there were the "normal" pedestrians which happened to be on the road. The vehicle driver was requested to maintain 30 km/h, traffic conditions permitting. Statistics for both sequences are shown in Table 5.1. Sequence *Run1* was used to perform the parameter optimization experiments below, system evaluation as described in the next Section was done on *Run2*. None of these sequences was used for the training process of a system module (as shape or texture pattern examples).

A matching criterion is required to define whether entities from ground truth and system output match. Considering application-specific tolerances and inaccuracies in measuring ground truth positions, we specify localization tolerances as a percentage of distance, 10% in lateral and 30% in longitudinal direction. A sensor coverage area was defined at 10m to 25m in front of the car, and 4m to each side. The pedestrian recognition capability is only required within this area, and considered optional outside in the sense that the system is not rewarded/penalized for correct/false/missing detections.

5.6.3 Parameter Optimization Experiments

Parameters of the PROTECTOR system subject to optimization are listed in Figure 5.7. Threshold parameters that apply to the same ROIs are grouped to speed up computation. For one such parameter group, only a single run over the training images is necessary per prior parameter setting to record the module outputs on which the thresholds are subsequently applied. These groups are

- parameters ψ and δ of the ROI generation module,

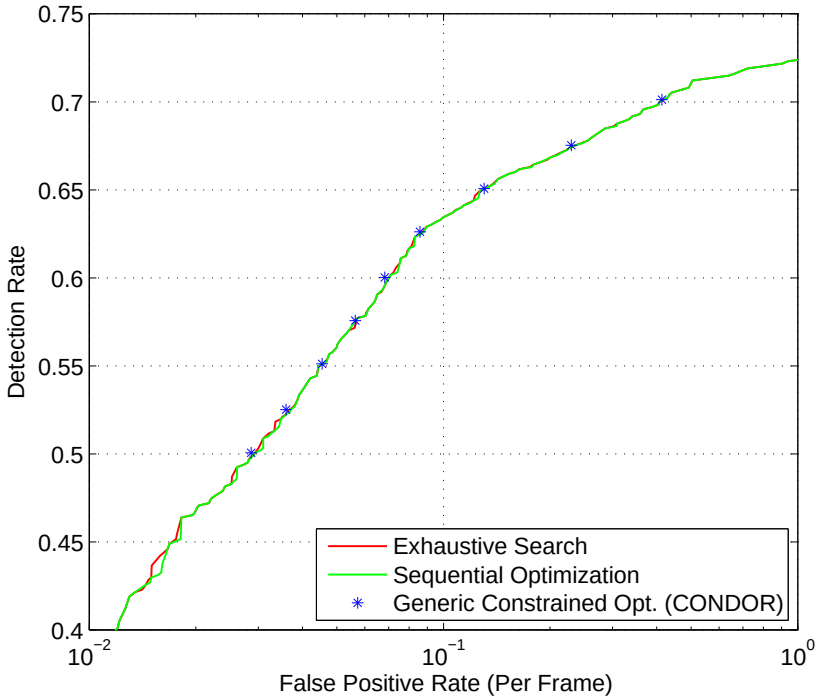


Figure 5.10: Comparison of optimization techniques applied to a subsystem of the PROTECTOR system. Three parameters from three system modules were selected for optimization, see text. ROC curves of the sequential optimization methods and exhaustive search coincide.

- the final chamfer distance threshold D_3, the texture classification output threshold C, and all parameters of the dense stereo-based verification module, and
- parameters m and n of the tracking module.

Before turning to the optimization of the full PROTECTOR system, we first evaluate the efficacy of the different cascade optimization techniques on a "real-world" detection problem. For reasons of computational complexity, this is done on subsystems only.

The first experiment involves to compare the sequential and the generic optimization technique with approximate ground truth obtained by an exhaustive search with a coarse discretization of the parameter ranges. We restricted optimization to only three parameters of the same group and their respective system modules: the chamfer distance threshold D_3 on the leaf level of the shape-based detection module, the output threshold C of the texture classifier, and the correlation threshold H of the dense stereo-based verification module.

ROC results shown in Figure 5.10 conform to those of the classification experiments of Section 5.5.2. ROC curves of the sequential optimization technique and the exhaus-

tive search coincide, i.e., the sequential optimization technique performs optimally. A subsequent generic optimization step yields a negligible reduction of the false positive rate. Resulting parameter settings differ from those found by exhaustive search only by the amount of discretization.

Differences between the optimization techniques appear with regard to their computational complexity. 100 discrete values per parameter were considered for the techniques that require discretization, so that a search over 10^6 different parameter settings was necessary for the exhaustive search. For the sequential optimization, 10^4 different settings were tried in the first step (first two parameters) with a result of 285 ROC points, so that the second step involved a search over 285×100 parameter settings, hence a total of only 38,500. In the CONDOR method, the average number of evaluations of the objective function per ROC point was 22, which confirms the effectiveness of the local function approximation approach used, plus a higher number of evaluations of the constraint function.

In a second experiment, the comparison to an exhaustive search is dropped and the two optimization techniques are applied to all PROTECTOR detection modules, i.e., all but the final tracking module. Optimization was performed without a processing time constraint, and by restricting the average processing time per frame to a maximum of 100ms. For computational reasons, six discrete parameter settings were chosen for parameters $\psi, \delta, E_1, E_2, E_3, D_1, D_2$. Since the remaining parameters belong to the same group, only six (computationally costly) passes over the training images were necessary. For these remaining parameters, 100 discrete values were chosen for the sequential optimization method. Results are shown in Figure 5.11. The maximum performance gain achieved by the subsequent generic optimization step (CONDOR) was 12%. Although this is non-negligible, it indicates that the sequential optimization technique yields useful results in real-world applications.

Finally, we applied the sequential optimization technique to the full PROTECTOR system, in order to obtain a overall optimized system. Figure 5.12 shows the successive improvements of the ROC curve obtained after each optimization step. A subsequent generic optimization step by the CONDOR method has been dropped for computational reasons.

5.6.4 System Evaluation

We considered two variants of the PROTECTOR system for evaluation, one optimized without a processing time constraint, and one with a maximum processing time of 100ms (per frame). The desired detection rate was set to 60%. The system such obtained was tested on the second video sequence, results are shown in Table 5.2.

Further insight into the system is gained by considering performance on the trajectory level in addition to a frame-by-frame evaluation. Trajectories from ground truth and system output are compared by requiring that a certain percentage of their entries match. The exact choice is application-dependent, we distinguish two types of

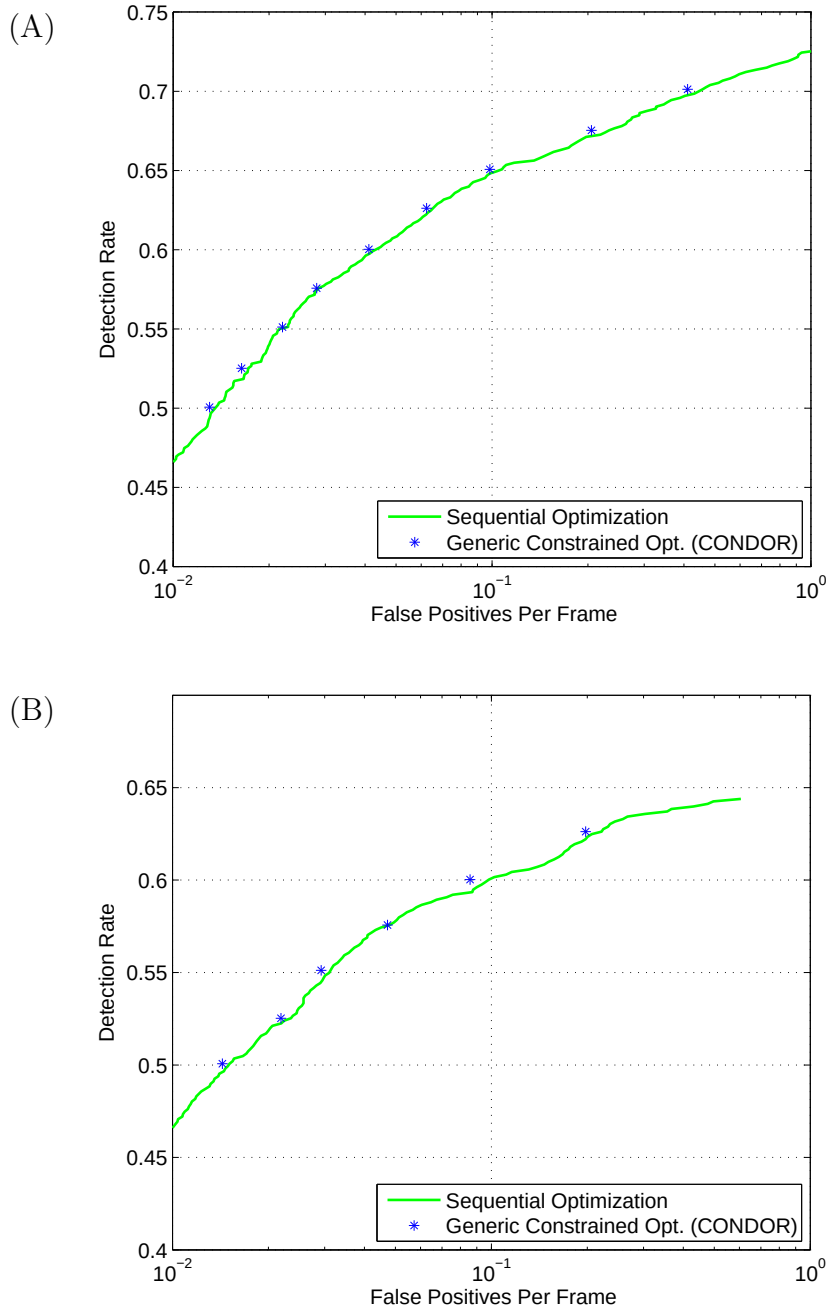
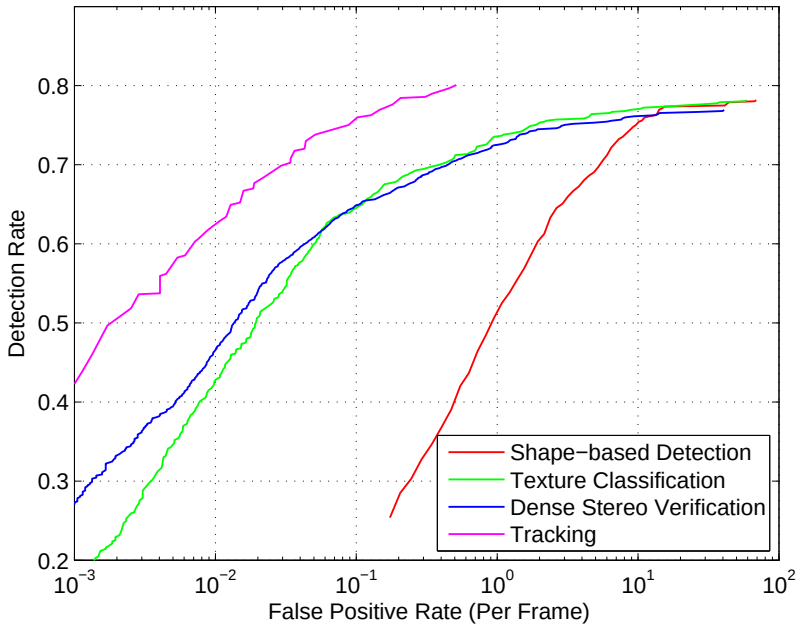


Figure 5.11: Optimization techniques applied to the PROTECTOR detection modules (i.e., all but the final tracking module). (A) Without a processing time constraint. (B) Processing time constraint $C^{\star} = 100$ms.

(A)



(B)

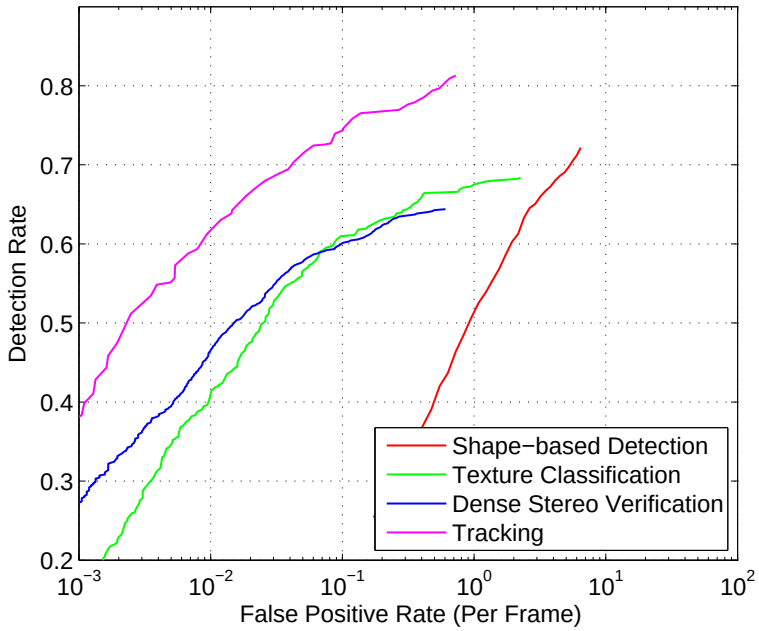


Figure 5.12: Parameter optimization of the full PROTECTOR system using the sequential optimization method. ROC results obtained after each system module are shown. (A) Optimization without a processing time constraint. (B) Processing time constraint $C^{\star} = 100$ms.

Table 5.2: System evaluation results obtained on the test sequence *Run2*: "F" denotes frame-level performance, "A"/"B" denote class-A/class-B trajectory performance, respectively. False positive rates (FPR) are given per 10^3 frames for frame-level performance, and per driving minute for trajectory performance.

	No Proc. Time Constraint			Max Proc. Time 100ms		
	F	A	B	F	A	B
Detection Rate (all)	61.0%	62.0%	78.0%	58.8%	64.0%	78.0%
FPR (per 10^3fr / min) (all)	23	3.6	3.5	27	5.2	5.1
Detection Rate (risky)	80.9%	90.0%	100%	75.3%	80.0%	100%
FPR (per 10^3fr / min) (risky)	0.69	0.33	0.33	1.0	0.38	0.38
Avg. Proc. Time Per Frame	162ms			101ms		

trajectories: "class-A" trajectories that have at least 50% of their entries matched, and "class-B" trajectories that have at least one entry matched. Thus, all class-A trajectories are also class-B trajectories, but the former pose stronger detection demands that might be necessary in some applications.

A separate evaluation was made for pedestrians directly in front of the car, i.e., which are in particular risk, by restricting the sensor coverage area to a maximum lateral offset from the vehicle medial axis of 1.5m (instead of 4m). Performance significantly increases when only those "risky" pedestrians are considered. For the (generous) class-B performance measure, an ideal detection rate of 100% is achieved.

Comparing both variants, we notice that enforcing the processing time constraint increases the false positive rate by 15% - 45%, but achieves a speed-up of almost 40%. The reason for this is that the processing time constraint enforces very strict parameters for the first two system modules (ROI generation and shape-based detection) to reduce the number of ROIs, while rather relaxed parameter settings are chosen for the subsequent modules in order to reach the desired detection rate.

6 Bayesian Detection and Tracking

While object detection and tracking frameworks have been extensively studied in the literature in isolation, this chapter aims to gain robustness by a tight integration of detection and tracking, and by the use of multiple visual cues. So far, only the recognition of static pedestrian patterns has been considered in this work, while temporal aspects were neglected. But the appearance alteration rate of pedestrians in successive video frames is limited by physical constraints. These are exploited by making decisions about the object class based on multiple successive observations, obtained using a sound Bayesian framework for object tracking, instead of a single one. A second source of robustness is the use of multiple complementary visual cues, where temporary failures of one cue may be compensated by the others.

A spatio-temporal object representation is created that combines mixture models of shape and texture, both learned from training data. The associated observation density function integrates the three visual cues *shape*, *texture*, and *depth*. Object shape is used, since it is distinctive yet sidesteps appearance variations due to texture, and because efficient matching techniques exist [37]. We make use of the spatio-temporal shape model proposed by Giebel et al.[38], which consists of the MPDM representation (cf. Section 4.1.2) for modeling 2D object silhouettes, and models temporal shape changes by means of a Markov transition matrix. This shape model is generative in that it allows to synthesize shape hypotheses from the transition prior. It furthermore provides accurate segmentation of an object's foreground region by matching the shape hypothesis to image data using an active contour algorithm.

The object model is complemented by a texture component that represents the variation of shape-normalized object (foreground) pixel intensities. Although pedestrian appearance variation is significantly reduced by shape normalization, the distribution of pedestrian texture is still very complex due to the great diversity of clothing and lighting conditions. Although generative texture models have been developed in the past (e.g., linear models based on PCA), we refrain from this approach. Instead, we make use of a generic texture classifier to find the decision boundary between object and non-object texture patterns, that has demonstrated its discrimination efficacy in the experiments of the previous chapters. Within a Bayesian tracking framework, this discrimination capability allows inference not only about object configuration (i.e., position, shape, etc.), but also about the object class (target object vs. background clutter), thus enabling the tracker to recognize false initializations and object disappearance.

We furthermore model 3D object kinematics (position and velocity), which allows to incorporate available real-world knowledge, and for which we obtain direct observations

by means of stereo imaging (*depth cue*).

The multi-cue object model is applied within a Bayesian framework for pedestrian detection and tracking based on particle filtering. Due to the cluttered environment of our prospective application, and due to the use of highly non-linear observations, we are faced with a non-Gaussian, multi-modal posterior probability density function for which a particle-based representation is a natural choice. An independently operating object detection module provides object hypotheses from single image frames, which are used to initialize new tracks and which serve as an additional source of information in the proposal distribution of existing tracks.

This work differs from a previous multi-cue pedestrian tracking framework [38] in that the object representation combines shape and texture models, and in the state vector consisting of object class and configuration, which integrates object detection and tracking in the particle filtering framework. Furthermore, the texture observation is based on a pattern classifier, and the assumption of independent shape and texture observations is dropped.

6.1 Particle Filtering Framework

This section gives a brief derivation of the particle filtering framework for our particular state space. We make use of a standard SIR (sequential importance resampling [1]) particle filter that has been successfully employed in similar applications (e.g., [45]). A critical design choice involves the handling of the variable number of objects (pedestrians) that occur in the field of view. From a theoretical point of view, the straightforward solution is to construct a joint state space of variable dimension and to infer the number of objects in parallel with the object configurations. However, the computational burden of such an approach is high unless sophisticated techniques (such as [45, 51]) are being used. An alternative approach is to run multiple trackers in parallel, one for each target object, guided by some supervisor process for track initialization and termination. In this work, we follow the second approach.

Consequently, the state $\mathcal{X}$ of each tracker either represents the presence of a target object, denoted by $\mathcal{O}$, with object configuration $\mathbf{x}$, or the absence of an object, denoted by $\mathcal{N}$. The latter state arises from false initializations or the disappearance of objects from the field of view.

Filtering denotes the recursive computation of the posterior probability $p(\mathcal{X}_t \,|\, \mathcal{Z}_{1:t})$ of state $\mathcal{X}_t$ at time index t given the series of image observations $\mathcal{Z}_{1:t} = (\mathcal{Z}_1, \ldots, \mathcal{Z}_t)$. Following a Bayesian approach, inference of the object state is based on the pdf (probability density function) of temporal state transition and on the state-conditional observation density.

The transition pdf of object configuration $p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})$ is defined by our object representation (see below). Here, we need to specify the transition of object class $(\mathcal{O}_t, \mathcal{N}_t)$, i.e., the appearance and disappearance of objects. An illustration is given in Figure 6.1.

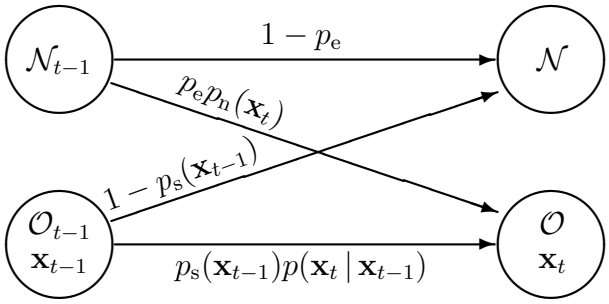


Figure 6.1: Transition PDF

- $p_s(\mathbf{x}_{t-1}) = p(\mathcal{O}_t \mid \mathcal{O}_{t-1}, \mathbf{x}_{t-1})$ represents the probability that an object remains within the detection area given its previous position ("Stay"),
- $p_e = p(\mathcal{O}_t \mid \mathcal{N}_{t-1})$ denotes the probability of the event that a new object enters the detection area, and
- $p_n(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathcal{O}_t, \mathcal{N}_{t-1})$ describes where new objects enter the detection area.

These functions are application-specific and need to be provided the user. The desired posterior is then obtained using Bayes rule as (cf. Eq. (4) in [1])

$$p(\mathcal{X}_t \mid \mathcal{Z}_{1:t}) \;\propto\; p(\mathcal{Z}_t \mid \mathcal{X}_t)\, p(\mathcal{X}_t \mid \mathcal{Z}_{1:t-1})\,, \tag{6.1}$$

where the transition prior $p(\mathcal{X}_t \mid \mathcal{Z}_{1:t-1})$ is given by the Chapman-Kolmogorov equation (cf. Eq. (3) in [1])

$$\begin{aligned} p(\mathcal{O}_t, \mathbf{x}_t \mid \mathcal{Z}_{1:t-1}) &= p_n(\mathbf{x}_t)\, p_e\, p(\mathcal{N}_{t-1} \mid \mathcal{Z}_{1:t-1}) \\ &\quad + \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p_s(\mathbf{x}_{t-1})\, p(\mathcal{O}_{t-1}, \mathbf{x}_{t-1} \mid \mathcal{Z}_{1:t-1})\, \mathrm{d}\mathbf{x}_{t-1}\,, \qquad (6.2) \\ p(\mathcal{N}_t \mid \mathcal{Z}_{1:t-1}) &= (1 - p_e)\, p(\mathcal{N}_{t-1} \mid \mathcal{Z}_{1:t-1}) \\ &\quad + \int (1 - p_s(\mathbf{x}_{t-1}))\; p(\mathcal{O}_{t-1}, \mathbf{x}_{t-1} \mid \mathcal{Z}_{1:t-1})\, \mathrm{d}\mathbf{x}_{t-1}\,. \qquad (6.3) \end{aligned}$$

In particle filtering, the posterior is approximated by a set of weighted samples or particles. For representing our hybrid state space, we dedicate one special particle with index 0 and weight $w_t^{(0)}$ to the case $\mathcal{N}_t$, while the remaining N_s particles $\{(\mathbf{x}_t^{(i)}, w_t^{(i)}) : i = 1, \ldots, N_s\}$ represent $(\mathcal{O}_t, \mathbf{x}_t)$. Formally,

$$\begin{aligned} p(\mathcal{N}_t \mid \mathcal{Z}_{1:t}) &\approx w_t^{(0)} \\ p(\mathcal{O}_t, \mathbf{x}_t \mid \mathcal{Z}_{1:t}) &\approx \textstyle\sum_{i=1}^{N_s} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})\,. \end{aligned} \tag{6.4}$$

At each time step t, a new particle set is drawn from a proposal distribution q_t. Here, we draw exactly one particle of state $\mathcal{N}_t$, and N_s particles of state $\mathcal{O}_t$ with object

configuration $\mathbf{x}_t$ sampled from the proposal $q_t(\mathbf{x}_t)$, i.e.

$$\begin{array}{rcccl} q_t(\mathcal{N}_t) & = & \frac{1}{N_s+1} & \propto & \frac{1}{N_s} \\ q_t(\mathcal{O}_t, \mathbf{x}_t) & = & \frac{N_s}{N_s+1} q_t(\mathbf{x}_t) & \propto & q_t(\mathbf{x}_t)\,. \end{array} \tag{6.5}$$

Particles are then weighted to represent the posterior,

$$w_t^{(i)} \;=\; \begin{cases} \dfrac{p(\mathcal{N}_t \,|\, \mathcal{Z}_{1:t})}{q_t(\mathcal{N}_t)} & \propto \; p(\mathcal{Z}_t \,|\, \mathcal{N}_t)\, p(\mathcal{N}_t \,|\, \mathcal{Z}_{1:t-1})\, N_s & \text{for } i = 0\,, \\ \dfrac{p(\mathcal{O}_t, \mathbf{x}_t^{(i)} \,|\, \mathcal{Z}_{1:t})}{q_t(\mathcal{O}_t, \mathbf{x}_t^{(i)})} & \propto \; \dfrac{p(\mathcal{Z}_t \,|\, \mathcal{O}_t, \mathbf{x}_t^{(i)})\, p(\mathcal{O}_t, \mathbf{x}_t^{(i)} \,|\, \mathcal{Z}_{1:t-1})}{q_t(\mathbf{x}_t)} & \text{for } i = 1, \ldots, N_s\,, \end{cases} \tag{6.6}$$

where the transition priors given in Eqs. (6.2) and (6.3) are now approximated by the particle set:

$$p(\mathcal{O}_t, \mathbf{x}_t \,|\, \mathcal{Z}_{1:t-1}) \;\approx p_{\mathrm{n}}(\mathbf{x}_t)\, p_{\mathrm{e}}\, w_{t-1}^{(0)} \;+\; \sum_{j=1}^{N_s} p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}^{(j)})\, p_{\mathrm{s}}(\mathbf{x}_{t-1}^{(j)})\, w_{t-1}^{(j)} \tag{6.7}$$

$$p(\mathcal{N}_t \,|\, \mathcal{Z}_{1:t-1}) \;\approx (1 - p_{\mathrm{e}})\, w_{t-1}^{(0)} \;+\; \sum_{j=1}^{N_s} (1 - p_{\mathrm{s}}(\mathbf{x}_{t-1}^{(j)}))\, w_{t-1}^{(j)} \tag{6.8}$$

Proportionalities in the above equations are resolved by normalizing the particle weights to sum to one.

The choice of a good proposal density is a crucial design step in the implementation of a particle filter [1]. The most convenient and most frequent choice is to use the transition prior $p(\mathcal{X}_t \,|\, \mathcal{Z}_{1:t-1})$ as approximated in Eqs. (6.7) and (6.8), because this greatly simplifies the particle weight computation in Eq. (6.6). But this choice is not necessarily optimal, as it may lead to many "wasted" particles with negligible weight, in particular in cases of noisy state prediction (widespread transition prior) and peaked observation densities. It is hence desirable to incorporate the current measurements into the proposal density in order to have particles generated close to the posterior distribution [44, 1]. Since sampling from $p(\mathcal{X}_t \,|\, \mathcal{Z}_t)$ is computationally expensive, the output of the independent target detector is used as an approximation, and the proposal density is then designed as a mixture of both sources of information; details are given in Section 6.3.

6.2 Multi-Cue Object Representation

This section details the proposed multi-cue object representation, their temporal transition, and their observation in video images. Three different visual cues are considered: shape, texture, and depth. Object *shape* is represented by its 2D contour, *texture* denotes the pixel intensity pattern within the object's contour after shape normalization, and *depth* refers to direct 3D measurements obtained from stereo imaging.

All three visual cues are represented in the object state $\mathbf{x}_t = (\mathbf{u}_t, k_t, \mathbf{b}_t, \mathbf{v}_t)$, which consists of object position and velocity in 3D, $\mathbf{u}_t$, pose cluster index k_t, shape model parameters $\mathbf{b}_t$, and texture $\mathbf{v}_t$, at time index t. For modeling the transition pdf (probability density function), we assume independence of the position component and use the decomposition

$$\begin{aligned} p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) &= p(\mathbf{u}_t, k_t, \mathbf{b}_t, \mathbf{v}_t \,|\, \mathbf{u}_{t-1}, k_{t-1}, \mathbf{b}_{t-1}, \mathbf{v}_{t-1}) \\ &= p(\mathbf{u}_t \,|\, \mathbf{u}_{t-1})\; p(k_t, \mathbf{b}_t \,|\, k_{t-1}, \mathbf{b}_{t-1})\; p(\mathbf{v}_t \,|\, k_t, k_{t-1}, \mathbf{b}_t, \mathbf{b}_{t-1}, \mathbf{v}_{t-1})\,. \end{aligned} \tag{6.9}$$

Details of each component transition pdf are given below along with the description of the respective visual cue.

6.2.1 Shape Cue

For shape representation, we make use of the Multi Point Distribution Model (MPDM) described in Section 4.1.2, a mixture of locally linear subspace models learned from training data. Each mixture component represents a distinct pose cluster. Shapes are represented by a pair $(k, \mathbf{b})$, where k is the pose cluster index and coefficient vector $\mathbf{b}$ describes the local variation within cluster k. The prior shape distribution is given by Eq. (4.6).

Shape Transition. The temporal transition of an object's shape is decomposed into pose cluster switching and shape changes within each cluster. The former is handled by a discrete first-order Markov process, where entry $T_{i,j}$ of the transition matrix describes the probability of switching from cluster $k_{t-1} = i$ to $k_t = j$. A Gaussian random walk is assumed for shape changes within the same cluster ($k_{t-1} = k_t$), while the shape prior is used in the case of a cluster switch. More precisely, if $(k_t, \mathbf{b}_t)$ is the shape state at time t, then

$$p(k_t, \mathbf{b}_t \,|\, k_{t-1}, \mathbf{b}_{t-1}) = T_{k_{t-1},k_t} \cdot \begin{cases} g_{k_t}(\mathbf{b}_t \,|\, \mathbf{b}_{t-1}) & \text{if } k_t = k_{t-1} \\ p(\mathbf{b}_t \,|\, k_t) & \text{if } k_t \neq k_{t-1}\,, \end{cases} \tag{6.10}$$

where g_{k_t} is a Gaussian random walk and $p(\mathbf{b}_t \,|\, k_t)$ is the normal shape prior, both subject to the Mahalanobis threshold M prescribed above.

Shape Observation. The chamfer distance (4.2) is used to measure the similarity between a shape instantiation and an observed input image. Shape instantiation involves to project the shape representation $(k, \mathbf{b})$ and 3D object position $\mathbf{u}$ to the image plane to obtain the list of contour points C. As image features, the position and direction of edge pixels found in input image I are used as proposed in [30]. The resulting shape observation

$$z_{shape}(I, \mathbf{x}) = D_{chamfer}(C, I) \tag{6.11}$$

is incorporated into the joint observation density function in subsection 6.2.4.

6.2.2 Texture Cue

The texture cue represents the variation of the intensity pattern across the image region of target objects. Much like in the *Active Appearance Models* by Cootes et al.[15], appearance variations that arise from differing shapes are eliminated by normalizing each object image for shape. See Section 4.3 for details. Let $V_I(\mathbf{u}, k, \mathbf{b})$ denote the texture vector such obtained from image I at position $\mathbf{u}$ (projected to the image plane) with shape parameters $(k, \mathbf{b})$.

Texture Observation. We make use of a generic pattern classifier to find the decision boundary between object and non-object texture patterns. In chapter 3, we found a neural network with local receptive fields [100] particularly suitable for the task of pedestrian classification. One such neural network h_k is trained for each pose cluster $k = 1, \ldots, K$. Texture observation, given an input image and an hypothesized object configuration $\mathbf{x} = (\mathbf{u}, k, \mathbf{b}, \mathbf{v})$, then involves to feed the shape-normalized image patch $V_I(\mathbf{u}, k, \mathbf{b})$ into the neural network corresponding to pose cluster k to yield the texture observation

$$z_{texture}(I, \mathbf{x}) = h_k(V_I(\mathbf{u}, k, \mathbf{b})). \tag{6.12}$$

This observation value is integrated into the joint observation density function below.

Texture Transition. The shape-normalized texture pattern of a pedestrian is assumed to remain constant over time, plus some unknown random noise. The transition pdf is therefore modeled by cross-correlating the two consecutive texture state vectors $\mathbf{v}_{t-1}$ and $\mathbf{v}_t$. If there is no pose cluster switch, i.e., $k_{t-1} = k_t$, then the shape-normalized texture vectors $\mathbf{v}_{t-1}$ and $\mathbf{v}_t$ have pixel-wise correspondence and we define

$$p(\mathbf{v}_t \,|\, \mathbf{v}_{t-1}, k_{t-1} = k_t) \propto \exp\big(- \alpha_1 ZNCC(\mathbf{v}_{t-1}, \mathbf{v}_t) - \alpha_0 \big) . \tag{6.13}$$

ZNCC denotes the zero-mean normalized cross-correlation given by

$$ZNCC(\mathbf{a}, \mathbf{b}) = \frac{(\mathbf{a} - \bar{a}\mathbf{1}) \cdot (\mathbf{b} - \bar{b}\mathbf{1})}{\sqrt{(\mathbf{a} - \bar{a}\mathbf{1})^2 (\mathbf{b} - \bar{b}\mathbf{1})^2}} , \tag{6.14}$$

where $\bar{a}$ is the mean of vector $\mathbf{a}$. In the case of a pose cluster switch, $k_{t-1} \neq k_t$, we observe that texture transformations occur mainly in horizontal image direction, while the vertical intensity distribution remains approximately constant. This is exploited by matching the vertical profile of the two texture patterns given by a projection to the image y axis and resampling to some fixed length. We thus have

$$p(\mathbf{v}_t \,|\, \mathbf{v}_{t-1}, k_{t-1} \neq k_t) \propto \exp\big(- \beta_1 ZNCC(H_{k_{t-1}}(\mathbf{v}_{t-1}), H_{k_t}(\mathbf{v}_t)) - \beta_0 \big) , \tag{6.15}$$

where H is the vertical profile operator. The pdf parameters $\alpha_0, \ldots, \beta_1$ are learned from training data.

6.2.3 Depth Cue

The *depth* cue represents the 3D position and velocity of target objects, the former observed by means of stereo imaging. Modeling object position in 3D space rather than 2D image space simplifies the dynamical model (we assume constant velocities), and allows to incorporate scene constraints such as the assumptions of pedestrians standing with at least one foot on the ground. Thus, $\mathbf{u} = (u_x, u_y, u_z, u_{vx}, u_{vy}, u_{vz})$.

Depth Observation We make use of a feature-based, multi-resolution stereo algorithm developed by Franke [25]; alternative choices would have been possible. The outcome is a relatively sparse depth map that provides depth estimations along vertical image edges. Depth measurements are assumed normally distributed around the true depth, so we compute the mean difference as the depth observation value

$$z_{depth}(I, \mathbf{x}) = \overline{Z_I(\mathbf{u}, k, \mathbf{b})} - u_z \,, \tag{6.16}$$

where $\overline{Z_I(\mathbf{u}, k, \mathbf{b})}$ denotes the mean of depth measurements within the image region given by $(\mathbf{u}, k, \mathbf{b})$ projected to the image plane. $z_{depth}(I, \mathbf{x})$ is integrated into the joint observation density function below.

Dynamics The dynamics of the position and velocity component $\mathbf{u}$ of the state vector is modeled as a first-order auto-regressive process by

$$\mathbf{u}_t = \begin{pmatrix} I_3 & I_3 \Delta t \\ 0 & I_3 \end{pmatrix} \mathbf{u}_{t-1} \; + \; \mathbf{e}_u \Delta t \,, \quad \mathbf{e}_u \sim N(0, \Sigma_u) \,, \tag{6.17}$$

where Σ_u is the user-defined process noise, Δt the time interval, and I_3 denotes the 3×3 identity matrix.

6.2.4 Cue Integration

Having introduced the individual cues above, we now turn to their integration to model the (joint) density function $p(\mathcal{Z}_t \,|\, \mathcal{X}_t)$ of observing image features $\mathcal{Z}_t$ given the true state $\mathcal{X}_t$, by using the above cues $\mathbf{z} = (z_{shape}, z_{texture}, z_{depth})$. The state of interest $\mathcal{X}_t$ to be inferred from the video sequences is the presence and positions of pedestrians. Recall that we assume that either no target object is present at time t, or exactly one at position $\mathbf{x}_t$, which is denoted by $\mathcal{X}_t = \mathcal{N}_t$ and $\mathcal{X}_t = (\mathcal{O}_t, \mathbf{x}_t)$, respectively. (Multiple objects are handled by multiple trackers, one for each object, see Section 6.3.)

Roughly, input image I_t is expected to contain only non-object features in the case $\mathcal{X}_t = \mathcal{N}_t$, whereas in the state $\mathcal{X}_t = (\mathcal{O}_t, \mathbf{x}_t)$, object features are expected within the image region given by $\mathbf{x}_t$, and non-object features elsewhere. This decomposition is realized by making a simplifying assumption: Projecting object state $\mathbf{x}_t$ to the image plane subdivides the image into a foreground and a background region. Although not strictly

true, we assume that features extracted from these regions are statistically independent, and that these features obey a common foreground (FG) or a common background (BG) distribution, respectively. With this assumption at hand, we follow the reasoning of Sidenbladh and Black [84] and other authors [45, 81] to get

$$p(\mathcal{Z}_t \,|\, \mathcal{X}_t) \;\propto\; \begin{cases} \dfrac{p(\mathbf{z}(I_t, \mathbf{x}_t) \,|\, \mathrm{FG})}{p(\mathbf{z}(I_t, \mathbf{x}_t) \,|\, \mathrm{BG})} & \text{for } \mathcal{X}_t = (\mathcal{O}_t, \mathbf{x}_t) \\ 1 & \text{for } \mathcal{X}_t = \mathcal{N}_t \,, \end{cases} \tag{6.18}$$

with the proportionality factor being $p(\mathcal{Z}_t \,|\, \mathcal{N}_t)$.

In order to find a parametric model of the above likelihood ratio, we notice that:

- The distribution of Chamfer distances z_{shape} is approximated by an exponential distribution [74, 93].
- The neural network output $z_{texture}$ is approximately normally distributed about the class means.
- Both cues, shape and (shape-normalized) texture, represent complementary image features, so dependencies are relatively weak.

This motivates the use of a quadratic function to approximate the log of the likelihood ratio

$$\log \frac{p(\mathbf{z} \,|\, \mathrm{FG})}{p(\mathbf{z} \,|\, \mathrm{BG})} \approx \mathbf{z}^T R \mathbf{z} + \mathbf{r}^T \mathbf{z} + r_0 \,, \tag{6.19}$$

which covers multivariate normal distributions and univariate exponential distributions as special cases. Non-zero off-diagonal elements of matrix R represent statistical dependencies between the respective cues.

In our application, we further assume independence of the depth cue, i.e. that the deviation of observed depth values from the true depth in Eq. (6.16) is independent of object shape and texture, by setting the respective entries in R to zero. The foreground depth distribution is assumed normal with zero mean and a variance parameter determined from training data, while the background distribution is assumed uniform within the observation area. The remaining parameters are jointly learned by means of a least squares fit of the model (6.19) to the joint histogram observed on training data.

6.3 Application System

After having introduced the tracking framework and our object representation, we now describe how these are employed in our pedestrian detection and tracking system, and give details of application-specific components.

6.3.1 Target Object Detector

An independently operating target object detector is used to provide track initializations, and to guide the sampling process. Here, we make use of the computationally efficient PROTECTOR system as described in Section 5.6, with the tracking module removed.

The output of the target detector is a list of 3D positions of potential pedestrians. Deviations from the true positions are assumed to obey a normal distribution, with parameters learned from a training set. In order to use the list of detections in the proposal density of a particle filter, detections are associated to (possibly multiple) existing trackers, by means of a maximum distance to the mean track position. For each tracker, a detector density $g_t(\mathbf{u}_t)$ is built as a mixture of Gaussians from the list of associated detections. (g_t is left unspecified if this list is empty.) Detections not associated to any track are used to initialize new trackers, see below.

6.3.2 Proposal Density

The proposal density $q_t(\mathbf{x}_t)$ of our particle filter needs to possess two properties: First, both sampling from the proposal and evaluation of the density needs to be computationally efficient. Second, two sources of information, the transition prior and the detector density are to be incorporated into the proposal density by means of a mixture density. Given the decomposition of the transition prior in Eq. (6.9), sampling is done incrementally for each of the state vector components position, shape, and texture:

- Pdfs of the position component $\mathbf{u}_t$ are given as a Gaussian or mixture of Gaussians (Eq. (6.17) and definition of g_t above), for which efficient sampling and evaluation is possible. Therefore,

$$q_t(\mathbf{u}_t) = \rho g_t(\mathbf{u}_t) \; + \; (1-\rho) p(\mathbf{u}_t \,|\, \mathcal{O}_t, \mathcal{Z}_{1:t-1}) \,, \tag{6.20}$$

  where the mixing coefficient ρ is set to 0 if g_t is undefined (i.e. no associated detections), otherwise, we choose $\rho = 0.5$. The component transition prior in the second term,

$$p(\mathbf{u}_t \,|\, \mathcal{O}_t, \mathcal{Z}_{1:t-1}) = \frac{p(\mathcal{O}_t, \mathbf{u}_t \,|\, \mathcal{Z}_{1:t-1})}{1 - p(\mathcal{N}_t \,|\, \mathcal{Z}_{1:t-1})}$$

  is obtained from Eqs. (6.7) and (6.8) by replacing $\mathbf{x}_t$ with $\mathbf{u}_t$.

- The transition pdf of the shape component (6.10) is composed of a discrete first-order Markov process and a Gaussian random walk and allows efficient sampling and evaluation, so we let

$$q_t(k_t, \mathbf{b}_t) = p(k_t, \mathbf{b}_t \,|\, \mathcal{O}_t, \mathcal{Z}_{1:t-1}) \,, \tag{6.21}$$

  where the RHS is computed analogously to the position transition prior above.

- The transition pdf of the texture component (6.13),(6.15) based on cross-correlation is easily evaluated, but does not permit direct sampling. Instead, the texture component of particles is always sampled directly from the input image given the particle's position and shape:

$$\mathbf{v}_t \,|\, \mathbf{u}_t, k_t, \mathbf{b}_t = V_{I_t}(\mathbf{u}_t, k, \mathbf{b}_t)\,. \tag{6.22}$$

The joint proposal density is then composed of the above parts as

$$q_t\big(\mathbf{x}_t = (\mathbf{u}_t, k_t, \mathbf{b}_t, \mathbf{v}_t)\big) = \begin{cases} q_t(\mathbf{u}_t) q_t(k_t, \mathbf{b}_t) & \text{if } \mathbf{v}_t = V_{I_t}(\mathbf{u}_t, k_t, \mathbf{b}_t)\,, \\ 0 & \text{otherwise.} \end{cases} \tag{6.23}$$

In order to explore the high-dimensional state space with only a limited number of particles, we employ a particle optimization step as proposed in [41]. After each new sample is drawn from the proposal density, an active contour algorithm is used to refine the shape and to obtain an accurate segmentation of the foreground region, which is crucial for subsequent texture observations.

6.3.3 Track Initialization and Termination

New tracks are initialized if a detection made by the target object detector is not associated to an existing track. In order to suppress spurious detections, tracks start *hidden*, i.e., their output is suppressed. A track becomes *visible* if the probability of not tracking a target object, $p(\mathcal{N}_t \,|\, \mathcal{Z}_{1:t})$, falls below a threshold $\theta_{visible}$, while we switch back to *hidden* if $p(\mathcal{N}_t \,|\, \mathcal{Z}_{1:t}) > \theta_{hidden}$. Tracks are terminated if $p(\mathcal{N}_t \,|\, \mathcal{Z}_{1:t})$ exceeds the user-defined threshold θ_{term}. For the experiments below, we chose $\theta_{visible} = 0.5$, $\theta_{hidden} = 0.7$, and $\theta_{term} = 0.9$.

To avoid that multiple trackers "jump" onto the same target, the track with higher non-target posterior is discarded if the mean object positions of two tracks coincide (subject to some user-defined tolerances).

6.4 Experiments

6.4.1 Model Generation

The shape-texture mixture model generated in Chapter 4.4 is reused here. Generation of the shape model is based on a training set of 6,522 pedestrian examples. Manually, contour pixels were labeled, and the training set was clustered into 12 pose clusters as shown in Figure 4.5. The pose-specific texture classifiers employed here are the background-masked neural networks with local receptive fields studied in Section 4.4.5, trained on the same training set plus a set of 7,000 video images without pedestrians, after shape normalization was applied. Notice that, opposed to the mixture-of-experts

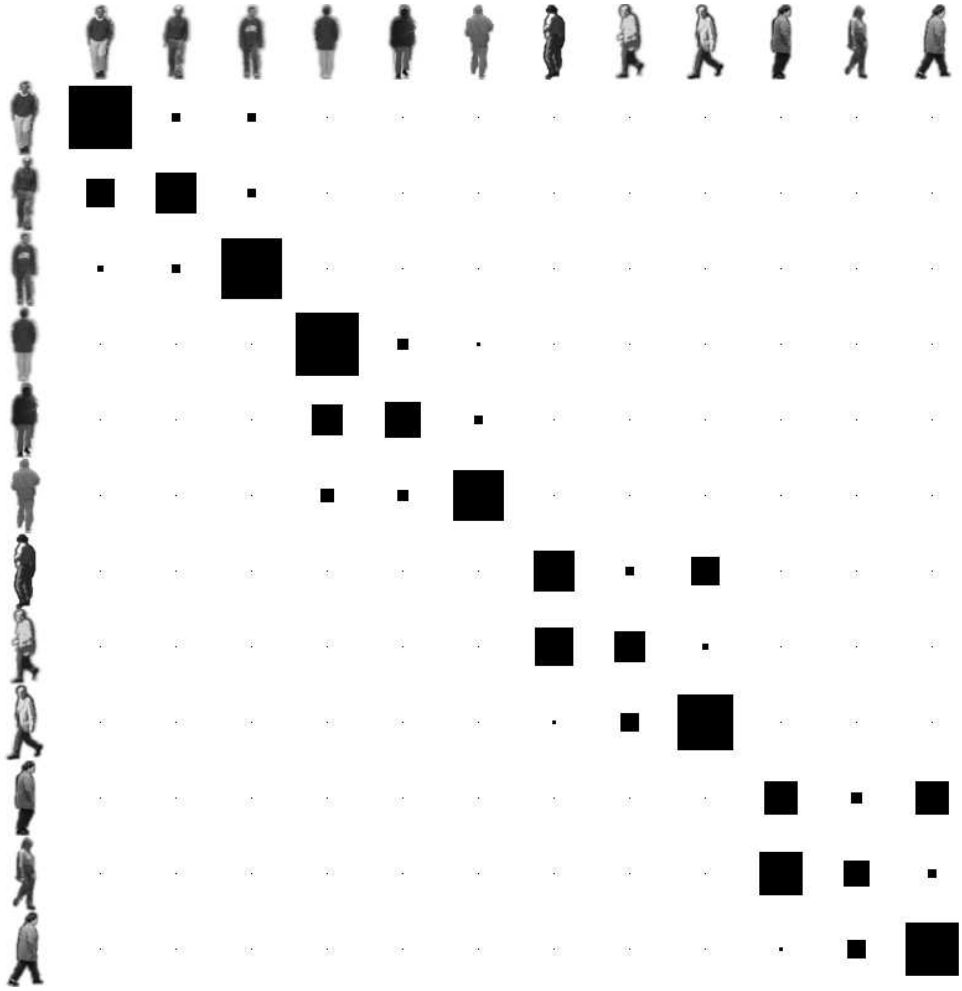


Figure 6.2: Visualization of the pose cluster transition matrix. The size of the squares represents the transition probability from a pose cluster of row i to a pose cluster of column j. Example images from each cluster are given for visualization purposes.

Figure 6.3: Example of a synthesized trajectory from the spatio-temporal shape model. Starting from the mean shape of cluster 11 (leftmost example), random samples were drawn from the shape transition pdf (6.10). Greyscale changes represent shape cluster switches.

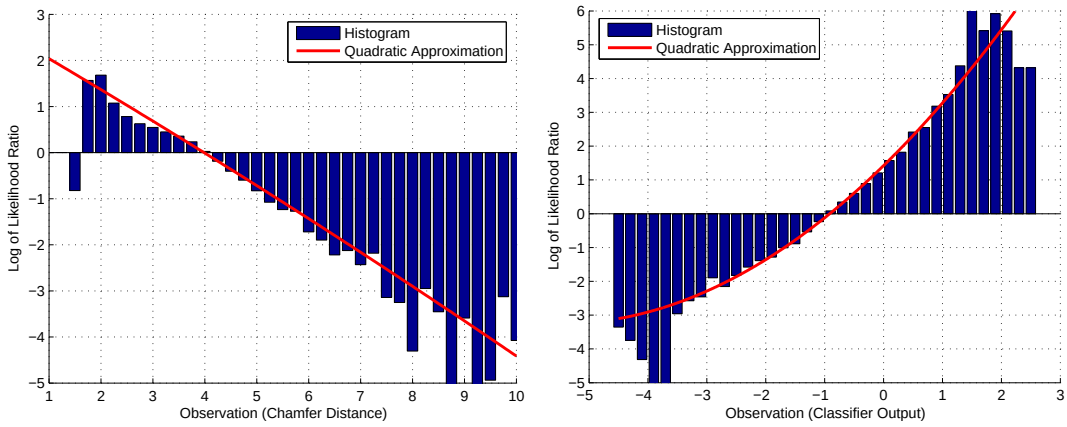


Figure 6.4: Quadratic approximation of the log of the likelihood ratio, shown for the marginal distributions of z_{shape} (left), and $z_{texture}$ (right). The ragged borders of both histograms plots arise from sparsely populated histogram bins, whereas the more densely populated regions around the graph centers are well approximated by the quadratic function.

architecture in Chapter 4, no mixture weights are to be learned here since the cluster selection is given by the hypothesized object state.

The pose cluster transition matrix has been obtained from the same pedestrian training set, see Figure 6.2 for an illustration. This matrix exhibits a clear block-diagonal structure where each block corresponds to a certain viewing direction. That is because changes of leg articulation are more likely to occur than changes in the viewing direction, i.e., pedestrians tend to walk straight on. Figure 6.3 illustrates the resulting shape transition pdf.

The next step involves to learn the joint observation density function that integrates the three visual cues shape, texture, and depth. The latter is assumed to follow an independent Gaussian distribution with parameters obtained from previous experiments. Parameters of the log likelihood ratio (6.19) for the shape and texture cue are learned from an additional training set independent from the above, in order to avoid distortions due to classifier over-fitting. Figure 6.4 shows a verification of the quadratic approximation by a comparison to the log likelihood ratio obtained from histogramming. Disregarding the ragged borders of the histogram results that arise from sparse data, the quadratic approximation is shown to fit the data very well. Whereas Figure 6.4 shows to separate results for shape and texture for visualization purposes, a joint approximation is used in the application system below.

Table 6.1: Experimental results comparing the proposed multi-cue detection and tracking approach to the performance of the detection module alone and to that of the PROTECTOR system of Section 5.6. See text for details.

	Detector Only			PROTECTOR			This Approach		
	F	A	B	F	A	B	F	A	B
Detection Rate	49.3%	55.1%	83.7%	61.0%	62.0%	78.0%	61.4%	65.3%	77.6%
Tracking Rate	(63.3%)	(70.0%)	(100%)	76.1%	80.0%	95.0%	77.6%	85.0%	92.5%
FPR (per 10^3fr, min)	14.2			23.2	3.6	3.5	16.1	2.0	2.0

Columns "F" show frame-level performance, "A" and "B" denote class-A and class-B trajectory performance, respectively. "FPR" denotes the number of false positives and is given per 10^3 frames for frame-level performance, and per driving minute for trajectory-level performance. "Tracking Rate" denotes the rate of object detection after the first track initialization. Numbers for the detector only are given for reference; see text for details.

6.4.2 System Evaluation

We tested the resulting detection and tracking system on the same 24min video sequences that was used in Section 5.6.4, applying the same evaluation criteria. Performance evaluation results are listed in Table 6.1 (column "This Approach"), in comparison to that of the detector module alone and that of the PROTECTOR system (Section 5.6). Slightly different parameter setting were used for the target detector in this chapter and the PROTECTOR system. For the latter, the ROC point at 60% detection rate was chosen, whereas here, we adjusted the detector towards fewer false positives.

At an approximately equal detection rate, the Bayesian multi-cue approach proposed here has a significantly reduced false positive rate at both, the frame-level and the trajectory level (by about 30% and 44%, respectively). This result confirms the efficacy of the integrated detection and tracking approach, where a Bayesian decision about the object class is based on multiple successive observations. In opposite, the α-β tracker of the PROTECTOR system merely concatenates single-frame results. Compared to the detector module alone, the slight increase in the frame-level false positive rate (14.2 vs. 16.1 per 10^3 frames) seems to be contradictory to these results. This increase is caused by a few false positive tracks that are erroneously continued over a couple of frames. However, notice that a far greater increase is necessary for the PROTECTOR system to achieve the same detection rate.

A second evaluation examines the tracking capabilities of both systems by only considering those ground truth pedestrians that follow the first track initialization made by the detector module. That is, along each ground truth trajectory, the pedestrian instances prior to the first matching detection are ignored, and ground truth trajectories without any matching detection are not considered at all. See row "Tracking Rate" in Table

Figure 6.5: Typical example of false positives of the detector process (red boxes in top subimages) that lead to false tracks in the PROTECTOR system (middle subimages). The multi-cue tracker correctly identifies these initialization "non-pedestrian" (black contours in bottom subimages).

6.1. Numbers for the detector represent the baseline performance obtained without any tracking capabilities. By definition, its class-B tracking rate is 100%. The proposed multi-cue tracker achieves a class-B tracking rate of 92.5%, i.e., 92.5% of the at least once detected true trajectories lead to a correct track, while 7.5% are erroneously discarded. The α-β tracker of the PROTECTOR system achieves a slightly better 95.0% here, but falls short on the class-A criterion (80% vs. 85%). This results corresponds to the observation that, once initialized, the multi-cue tracker is able to correctly keep track of a pedestrian, whereas the α-β tracker often loses track and needs reinitialization by the detector process.

Figures 6.5 and 6.6 show results of a few example frames of the test sequence. Top subimages show the output of the detector module (red boxes), middle subimages the output of the PROTECTOR system after α-β tracking. The maximum a-posteriori output (i.e., the particle with maximum weight) of the proposed multi-cue tracking is given in the bottom subimages. Red contours denote *visible* tracks which are classified as tracking a target object, while black contours denote *hidden* tracks that are more likely not tracking a target object. In Figure 6.6, the detector module only provides sporadic detections of the two pedestrians, so that the α-β tracker loses track. In contrast, the multi-cue tracking correctly detects and tracks the two pedestrians after a few initialization frames. Figure 6.5 shows an example of false positives of the detector process that leads to a false track initialization. In contrast to the α-β tracker, the multi-cue tracker correctly classifies this track as "non-pedestrian".

With 500 particles used per track, overall processing time was about 15s per frame

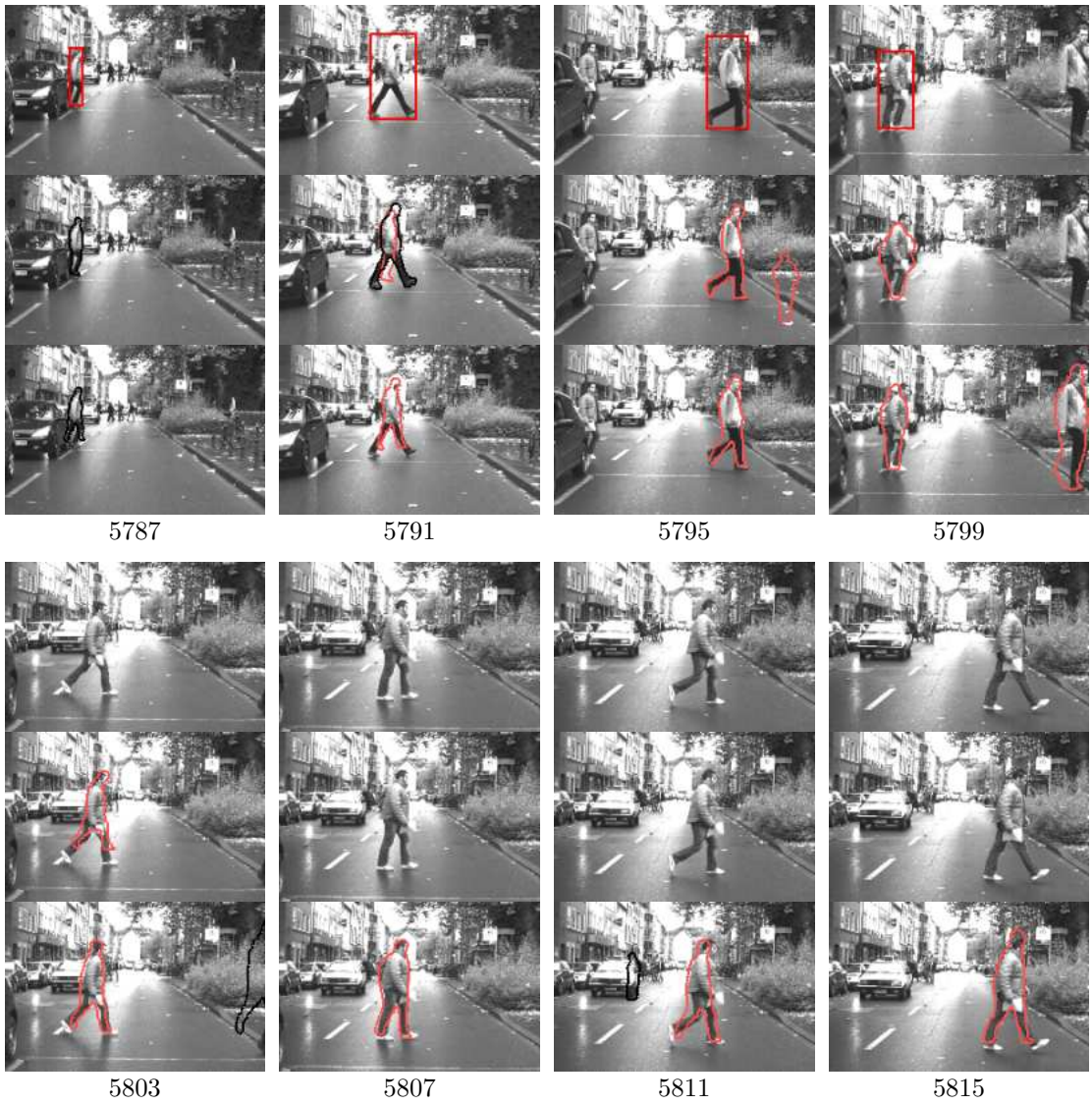


Figure 6.6: Example results obtained on the test sequence. In each row, the top subimages show results of the detector module alone, middle subimages show the PROTECTOR system output, and bottom subimages show results of the multi-cue tracker. Tracks classified as *visible* are shown by red contours, i.e. they denote the actual system output, while *hidden* tracks are shown in black. Frame numbers are given below each image.

and track on a 3.2GHz Pentium IV PC. Shape normalization, i.e. the warping of the texture patterns to the cluster mean shape, turned out to be the bottleneck of the current implementation that consumes the large majority of processing time.

7 Conclusion

7.1 Summary

This thesis investigated what combinations of visual cues, feature extraction techniques, and decision making rules perform best for detecting and tracking deformable objects in complex environments.

Texture classification. In the first part of this thesis, an in-depth experimental study on pedestrian classification was conducted. Multiple combinations of feature extraction and pattern classification were examined with respect to their ROC performance and efficiency on a large data set with ground truth. Local features were found superior to global features, here represented by PCA coefficients. Among the former, adaptive features (local receptive fields) outperformed non-adaptive ones (Haar wavelets). SVMs generally performed best regardless of the feature type, except for Haar wavelet features were an AdaBoost cascade approach achieved comparable performance at much lower computational costs.

The greatest performance gain was, however, achieved by (manually) increasing the training sample size. Since the acquisition of target training examples by manual labeling is expensive, two methods for the automatic generation of non-target examples were examined. Although a significant performance gain could be achieved, these methods ran into saturation after a few iterations.

Shape-texture integration. The integration of the two visual cues shape and texture was investigated, aimed at determining how texture classification benefits from explicit prior shape knowledge.

In a first step, a mixture-of-experts approach was pursued. One specialized texture classifier, the local expert, was learned per pose cluster, and their outcomes were combined by a weighted average with weights derived from shape matching. The use of shape information during both, the training and the recall stage, was shown to outperform related classifier combination methods without explicit shape knowledge (random data splits and simple averaging). By deriving probabilistic cluster assignments, errors in shape matching and automatic shape clustering could be compensated for. From the two clustering approaches tested, a convenient automatic shape clustering based on the chamfer distance was as good as a tedious manual clustering. Experiments on a large

pedestrian dataset revealed an overall reduction of the false positive rate of 30–40% compared to a single texture classifier without shape knowledge, at equal detection rates. Computation costs, though, increase proportionally with the number of clusters.

In a second step, remaining intra-cluster shape variability was eliminated via shape normalization. Based on a parametric (linear) model of contour point variation, all input images were warped to a common cluster prototype shape prior to texture classification. Furthermore, the resulting pixel-wise correspondence among the texture patches allowed to mask out background pixels. This has led to an additional performance gain of 10–20% in the experiments. However, the additional processing cost for image warping and the susceptibility to shape matching errors impairs practical applications of this technique.

Cascade optimization. If computational efficiency is an issue, cascade architectures consisting of a sequence of successively more complex system modules become attractive. While there is a large body of literature on how to generate individual system modules based on different visual cues, feature extraction, or classification methods, surprisingly few approaches exist for the optimization of a generic complex cascade system. Three optimization techniques were described in this thesis, including a novel sequential optimization method, which adjust module parameters to optimize for the three performance measures detection rate, false positive rate, and processing cost. Although none of these methods guarantees an optimal outcome, experimental results on synthetic and real-world data sets have shown that (close-to) optimal parameter settings can be obtained by a combination of the sequential optimization method with a subsequent iterative post-processing by a generic optimization technique, given the latter is computationally feasible.

Bayesian detection and tracking. The last part of this thesis dealt with object detection based on a sequence of observations instead of single static images. Building upon previous work in [38], a joint object representation and associated observation density function was described that integrates the three visual cues shape, texture, and depth. Shape was represented by a set of linear subspace models called Multi Point Distribution Models (MPDM). The texture cue was composed of a static representation by means of a texture classifier, and a dynamic texture component observed via cross-correlation. Direct depth observations were given by stereo imaging. This combination of a generative shape model for exact localization and a discriminative texture classifier allowed a particle filter-based tracker to make simultaneous inference of object class and configuration. Large-scale experiments showed a performance gain of 30–44% (in terms of reduction of the false positive rate at equal detection rate) compared to the non-Bayesian cascade system. However, this gain was paid with a large increase of the processing cost by about a factor of 100.

The proposed methods were applied to the problem of recognizing pedestrians in urban environment from within a moving vehicle. Large-scale experiments involving thousands of video images proved that the achieved pedestrian detection performance is on the leading edge (cf. Tables 2.1 and 6.1).

7.2 Future Work

There are several ways to extend the current work. One obvious consequence from the texture classification study (Chapter 3) is the need for more (target) training examples. One could diligently continue labeling images, but this is time consuming and expensive. Techniques that generate virtual target examples from an existing training set seem appealing. For this, a generative texture model of the target class is required, which, in general, is difficult to obtain. But opposed to those texture models that are to be used for model matching and parameter estimation, the texture model required here does not need to be comprehensive. It suffices to model only those components of the texture variation for which prior knowledge or explicit models exist, e.g., global illumination changes and local shape variation, while a representation of the remaining variation components, such as textile patterns of the clothes, is left to the original training set, to be learned by the texture classifier. A successful early example of such a method can be found in [22], further advancement and integration with the pose-specific mixture-of-experts techniques of this thesis seems to be a worthwhile direction of future research.

In terms of classification methods, a combination of the best-performing features, local receptive fields, and the computationally efficient AdaBoost classifier could be investigated, trying to achieve the same good classification performance at lower computational cost.

Regarding cascade systems, future work could investigate probabilistic / Bayesian techniques for combining module outputs, as previously developed for homogeneous cascade systems [103, 32, 88]. The modules in the current system emit binary outputs: A ROI is either classified as containing a target object and passed to the next cascade stage, or otherwise discarded. All information about the confidence in that decision is lost, whereas on the other hand, the performance gain of the Bayesian tracking approach (Chapter 6) is partly achieved by the joint probabilistic object model. Inferring a class posterior probability from all available information, i.e., by integrating the results of the previous system modules, supposedly leads to a more accurate decision. The computational efficiency of a cascade architecture is maintained by thresholding the posterior values after each cascade stage, and the proposed cascade optimization techniques are then applied to these thresholds.

The proposed Bayesian tracking system mainly suffers from slow processing speed. A tight coupling of particle sampling with the hierarchical shape matching employed in the independent object detection module, aimed at guiding shape sampling towards the matched shapes, could be investigated.

Considering the pedestrian protection application, can this system be employed in a production vehicle? What performance is necessary certainly depends on the exact application scenario. If used within some visibility enhancement system that highlights pedestrians for drivers' notice, one or two false highlightings may be acceptable within a one hour drive, whereas the same frequency of false emergency brakings is not. Given the progress made within in this thesis together with expected further advances by the directions described above, vision-based pedestrian detection systems may soon reach a performance level viable for commercial applications.

Bibliography

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002. (page 10, 76, 77, 78)

[2] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software*, 22(4):469–483, 1996. (page 39)

[3] F. Vanden Berghen. *CONDOR: a constrained, non-linear, derivative-free parallel optimizer for continuous, high computing load, noisy objective functions*. PhD thesis, IRIDIA, Faculté des Science Appliquées, Université Libre de Bruxelles, 2004. (page 11, 57)

[4] F. Vanden Berghen. CONDOR Optimizer. http://www.applied-mathematics.net/, 2006. (page 59)

[5] M. Bergtholdt, D. Cremers, and C. Schnörr. Variational segmentation with shape priors. In N. Paragios, Y. Chen, and O. Faugeras, editors, *Mathematical Models in Computer Vision: The Handbook*. Springer, 2005. (page 6, 32)

[6] M. Bertozzi, A. Broggi, A. Fascioli, A. Tibaldi, R. Chapuis, and F. Chausse. Pedestrian localization and tracking system with kalman filtering. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 584–589, Parma, Italy, 2004. (page 9)

[7] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986. (page 21, 31)

[8] J.-Y. Bouguet. Camera calibration toolbox for Matlab. In *http://www.vision.caltech.edu/bouguetj/calib_doc/*, 2000. (page 65)

[9] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. (page 53)

[10] A. Broggi, A. Fascoli, I. Fedriga, A. Tibaldi, and M. Del Rose. Stereo-based preprocessing for human shape localization in unstructured environments. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 410–415, Ohio, U.S.A., 2003. (page 6)

[11] C.-C. Chang and C.-J. Lin. LIBSVM – A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/, 2004. (page 18)

[12] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994. (page 39)

[13] A.R. Conn, N.I.M. Gould, and P. L. Toint. *Trust-Region Methods*. SIAM/MPS Series on Optimization. SIAM, Philadelphia, 2000. (page 58)

[14] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models - their training and applications. *Computer Vision and Image Understanding*, 61(1):38–59, 1995. (page 6, 12, 32, 33)

[15] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001. (page 7, 80)

[16] T. F. Cootes and C. J. Taylor. Statistical models of appearance for computer vision. Technical report, Imaging Science and Biomedical Engineering, University of Manchester, Manchester, U.K., 2004. (page 7)

[17] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. (page 18)

[18] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 886–893, San Diego, CA, USA, 2005. (page 9)

[19] J. Deutscher, A. Blake, and I. D. Reid. Articulated body motion capture by annealed particle filtering. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 126–133, 2000. (page 10, 12)

[20] R. Duda, P. Hart, and D. Stork. *Pattern Classification, second edition*. John Wiley and Sons, New York, 2001. (page 7)

[21] H. Elzein, S. Lakshmanan, and P. Watta. A motion and shape-based pedestrian detection algorithm. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 500–504, Ohio, U.S.A., 2003. (page 6, 8, 9)

[22] M. Enzweiler. Resampling techniques for pedestrian classification. Master's thesis, University of Ulm, Faculty of Computer Science, 2005. (page 37, 39, 93)

[23] R. Fablet and M. Black. Automatic detection and tracking of human motion with a view-based representation. In *ECCV*, pages 476–491, 2002. (page 7, 12)

[24] L. Fan, K.K. Sung, and T.K. Ng. Pedestrian registration in static images with unconstrained background. *Pattern Recognition*, 36(4):1019–1029, April 2003. (page 7)

[25] U. Franke. Real-time stereo vision for urban traffic scene understanding. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Detroit, USA, 2000. (page 66, 81)

[26] U. Franke and S. Heinrich. Fast obstacle detection for urban traffic situations. *IEEE Transactions on Intelligent Transportation Systems*, 3(3):173–181, 2002. (page 6)

[27] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995. (page 8, 19)

[28] K. Fukushima, S. Miyake, and T. Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, 13:826–834, 1983. (page 17)

[29] G. Fumera and F. Roli. A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):942–956, 2005. (page 34, 36)

[30] D. M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Proc. of the International Conference on Pattern Recognition*, pages 439–444, Brisbane, 1998. (page 79)

[31] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999. (page 6)

[32] D. M. Gavrila. A Bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8), 2007. (page 93)

[33] D. M. Gavrila and J. Giebel. Shape-based pedestrian detection and tracking. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Versailles, France, 2002. (page 9)

[34] D. M. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection: the PROTECTOR+ system. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 13–18, Parma, Italy, 2004. (page -)

[35] D. M. Gavrila, J. Giebel, and H. Neumann. Learning shape models from examples. In *Proc. of the Deutsche Arbeitsgemeinschaft für Mustererkennung*, pages 369–376, Munich, Germany, 2001. (page 6, 32)

[36] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision*, 73(1):41–59, 2007. (page 49)

[37] D. M. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *Proc. of the International Conference on Computer Vision (ICCV'99)*, pages 87–93, Kerkyra, Greece, 1999. (page 6, 21, 31, 32, 37, 66, 75)

[38] J. Giebel, D. M. Gavrila, and C. Schnörr. A Bayesian framework for multi-cue 3d object tracking. In *Proc. of the European Conference on Computer Vision (ECCV'04)*, Prague, Czech Republic, May 11-14 2004. Springer-Verlag. (page 6, 7, 75, 76, 92)

[39] G. Grubb, A. Zelinsky, L. Nilsson, and M. Ribbe. 3d vision sensing for improved pedestrian safety. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 19–24, Parma, Italy, 2004. (page 6, 8, 9)

[40] J. B. Hampshire and A. Waibel. The meta-pi network - building distributed knowledge representations for robust multisource pattern-recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):751–769, 1992. (page 8, 34)

[41] T. Heap and D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. In *Proc. of the International Conference on Computer Vision*, pages 344–349, Bombay, India, 1998. (page 6, 7, 10, 84)

[42] X. Huo and J. Chen. Building a cascade detector and applications in automatic target recognition. *Applied Optics: Information Processing*, 43(2):293–303, 2004. (page 13)

[43] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. (page 10)

[44] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. of the European Conference on Computer Vision*, volume 1, pages 893–908, 1998. (page 10, 11, 78)

[45] M. Isard and J. MacCormick. BraMBLE: a bayesian multiple-blob tracker. In *Proc. of the International Conference on Computer Vision*, volume II, pages 34–41, 2001. (page 7, 11, 12, 76, 82)

[46] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. (page 8, 34)

[47] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000. (page 7, 16, 17, 18, 19)

[48] M. J. Jones and T. Poggio. Multidimensional morphable models. In *Proc. of the International Conference on Computer Vision*, pages 683–688, 1998. (page 7)

[49] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994. (page 8, 34)

[50] H.-G. Kang and D. Kim. Real-time multiple people tracking using competitive condensation. *Pattern Recognition*, 38(7):1045–1058, 2005. (page 11, 12)

[51] Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1805–1819, 2005. (page 11, 76)

[52] S. Kirkpatrick, Jr. C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. (page 32)

[53] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998. (page 34)

[54] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955. (page 68)

[55] L. Kuncheva, J. C. Bezdek, and R. P. W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001. (page 8, 34)

[56] L. I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002. (page 34)

[57] R. Labayrade, D. Aubert, and J.-P Tarel. Real time obstacle detection on non flat road geometry through 'v-disparity' representation. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Versailles, France, 2002. (page 6)

[58] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 878–885, San Diego, CA, USA, 2005. (page 9)

[59] H. Luo. Optimization design of cascaded classifiers. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 480–485, San Diego, CA, USA, 2005. (page 13, 59)

[60] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71, 2000. (page 12)

[61] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proc. of the European Conference on Computer Vision (ECCV'00)*, volume II, pages 3–19, Dublin, Ireland, June 2000. Springer-Verlag. (page 10)

[62] I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, 2004. (page 7)

[63] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Proc. of the European Conference on Computer Vision (ECCV'04)*, volume I, pages 69–82, Prague, Czech Republic, May 11-14 2004. Springer-Verlag. (page 8, 9)

[64] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001. (page 8, 9, 18)

[65] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, 2006. (page -)

[66] S. Munder, C. Schnörr, and D. M. Gavrila. Pedestrian detection and tracking using a mixture of view-based shape-texture models. *Submitted to IEEE Transactions on Intelligent Transportation Systems*, 2007. (page -)

[67] C. Nakajima, M. Pontil, B. Heisele, and T. Poggio. Full-body recognition system. *Pattern Recognition*, 36:1997–2006, 2003. (page 8)

[68] H. Ning, T. Tan, L. Wang, and W. Hu. People tracking based on motion model and motion constraints with automatic initialization. *Pattern Recognition*, 37:1423–1440, 2004. (page 12)

[69] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Verlag, 2nd edition, 2006. (page 58)

[70] K. Okuma, A. Taleghani, N. de Freitas, J. Little, and D. Lowe. A boosted particle filter: Multitarget detection and tracking. In *Proc. of the European Conference on Computer Vision (ECCV'04)*, volume I, pages 28–39, Prague, Czech Republic, May 11-14 2004. Springer-Verlag. (page 11, 12)

[71] Intel open source computer vision library, 2004. http://www.intel.com/research/mrl/research/opencv/. (page 20)

[72] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000. (page 8, 9, 10, 16, 17, 18)

[73] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In *Neural Networks for Speech and Image Processing*, pages 126–142. Chapman-Hall, 1993. (page 34, 45)

[74] V. Philomin, R. Duraiswami, and L. S. Davis. Quasi-random sampling for condensation. In *Proc. of the European Conference on Computer Vision (ECCV'00)*, pages 134–149, Dublin, Ireland, June 2000. Springer-Verlag. (page 82)

[75] R. Polana and R.C. Nelson. Detecting activities. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2–7, New York, 1993. (page 6)

[76] M.J.D. Powell. UOBYQA: Unconstraint optimization by quadratic approximation. Technical Report DAMTP2000/14, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England, 2000. (page 58)

[77] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001. (page 13)

[78] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 271–278, San Diego, CA, USA, 2005. (page 7, 12)

[79] S. Roth, L. Sigal, and M. J. Black. Gibbs likelihoods for bayesian tracking. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 886–893, June 2004. (page 12)

[80] S. Sarkar and S. Chavali. Modeling parameter space behavior of vision systems using bayesian networks. *Computer Vision and Image Understanding*, 79:185–223, 2000. (page 11)

[81] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 56(3):151–177, 2004. (page 82)

[82] A. Shashua, Y. Gdalyaha, and G. Hayun. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Parma, Italy, 2004. (page 8, 9, 10)

[83] H. Shimizu and T. Poggio. Direction estimation of pedestrian from multiple still images. In *Proc. of the IEEE Intelligent Vehicle Symposium*, pages 596–600, Parma, Italy, 2004. (page 8)

[84] H. Sidenbladh and M. J. Black. Learning the statistics of people in images and video. *International Journal of Computer Vision*, 54(1/2/3):183–209, 2003. (page 7, 12, 82)

[85] A. Soto and P. Khosla. Probabilistic adaptive agent based system for dynamic state estimation using multiple visual cues. In *10th Int'l Symposium of Robotics Research (ISRR 2001).*, Lorne, Victoria, Australia, November 9-12 2001. (page 7, 12)

[86] M. Spengler and B. Schiele. Towards robust multi-cue integration for visual tracking. *Machine Vision and Applications*, 14(1):50–58, 2003. (page 7, 10, 12)

[87] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *Proc. of the International Conference on Computer Vision (ICCV'03)*, volume II, pages 1063–1070, Nice, France, October 2003. (page 6)

[88] B. Stenger, A. Thayananthan, P.H.S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical Bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, 2006. (page 6, 93)

[89] J. Sun, J. M. Rehg, and A. Bobick. Automatic cascade training with perturbation bias. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 276–283, 2004. (page 13)

[90] K. K. Sung and T. Poggio. Example based learning for view-based human face detection. Technical Report CBCL-112, MIT Artificial Intelligence Laboratory, January 1995. (page 10, 19)

[91] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata. Pedestrian detection with convolutional neural networks. In *Proc. of the IEEE Intelligent Vehicle Symposium*, 2005. (page 9)

[92] W. Thompson and T.-C. Pong. Detecting moving objects. *International Journal of Computer Vision*, 4:39–57, 1990. (page 6)

[93] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *International Journal of Computer Vision*, 48(1):9–19, 2002. (page 6, 12, 82)

[94] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996. (page 34, 36)

[95] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, pages 127–162. Springer-Verlag, London, 1999. (page 34)

[96] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995. (page 16)

[97] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998. (page 18)

[98] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'01)*, volume 1, pages 511–518, Kauai, Hawaii, 2001. (page 8)

[99] P. A. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005. (page 8, 9, 10, 11, 17, 19, 27, 49)

[100] C. Wöhler and J. Anlauf. An adaptable time-delay neural-network algorithm for image sequence analysis. *IEEE Transactions on Neural Networks*, 10(6):1531–1536, 1999. (page 8, 9, 17, 18, 80)

[101] B. Wu and R. Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 951–958, New York, NY, USA, 2006. (page 7, 12)

[102] Y. Wu and T. Yu. A field model for human detection and tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):753–765, 2006. (page 6, 12)

[103] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *Proc. of the International Conference on Computer Vision (ICCV'03)*, volume 1, page 709, Nice, France, October 2003. (page 93)

[104] J. Zhang, R. Collins, and Y. Liu. Bayesian body localization using mixture of nonlinear shape models. In *Proc. of the International Conference on Computer Vision (ICCV'05)*, pages 725–732, October 2005. (page 7)

[105] L. Zhao and C. Thorpe. Stereo- and neural network-based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(3), 2000. (page 5, 8, 9, 10)

[106] T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1208–1221, September 2004. (page 7, 12)