International Journal of Computer Vision 73(1), 41–59, 2007

DOI: 10.1007/s11263-006-9038-7

Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle

D. M. GAVRILA
Machine Perception, DaimlerChrysler Research and Development, 89081 Ulm, Germany;
Intelligent Systems Lab, Faculty of Science, University of Amsterdam, 1098 SJ Amsterdam, The Netherlands
dariu.gavrila@DaimlerChrysler.com

S. MUNDER
Machine Perception, DaimlerChrysler Research and Development, 89081 Ulm, Germany
stefan.munder@DaimlerChrysler.com

Received June 5, 2005; Revised December 22, 2005; Accepted May 4, 2006

First online version published in July, 2006

Abstract. This paper presents a multi-cue vision system for the real-time detection and tracking of pedestrians from a moving vehicle. The detection component involves a cascade of modules, each utilizing complementary visual criteria to successively narrow down the image search space, balancing robustness and efficiency considerations. Novel is the tight integration of the consecutive modules: (sparse) stereo-based ROI generation, shape-based detection, texture-based classification and (dense) stereo-based verification. For example, shape-based detection activates a weighted combination of texture-based classifiers, each attuned to a particular body pose.

Performance of individual modules and their interaction is analyzed by means of Receiver Operator Characteristics (ROCs). A sequential optimization technique allows the successive combination of individual ROCs, providing optimized system parameter settings in a systematic fashion, avoiding ad-hoc parameter tuning. Application-dependent processing constraints can be incorporated in the optimization procedure.

Results from extensive field tests in difficult urban traffic conditions suggest system performance is at the leading edge.

Keywords: multiple visual cues, pedestrian detection, intelligent vehicles

1. Introduction

Our long term aim is to develop video-based driver assistance systems for the detection of potentially dangerous situations with pedestrians, in order to either warn the driver, or, if no such time remains, initiate appropriate protective measures (e.g. automatic vehicle braking). See Fig. 1. The use of video sensors comes quite natural for this problem; they provide texture information at fine horizontal and vertical resolution, which in turn enables the use of discriminative pattern recognition techniques for distinguishing pedestrians from other static and dynamic objects in the traffic environment. The human visual perception system is perhaps the best example of what performance might be possible with such sensors, if only the appropriate processing were used. Yet the pedestrian application is very challenging from machine vision perspective. It combines the difficulties of a moving camera, a wide range of possible (deformable) object appearances, cluttered backgrounds, stringent performance criteria and hard real-time constraints.

Figure 1. Driver inattention can instantly lead to dangerous situations with pedestrians.

The outline of the paper is as follows. Section 2 discusses previous work in this area. As will become apparent, a multitude of interesting approaches have been proposed in the literature, mostly focusing on subproblems (i.e. detection vs. tracking, lateral gait). Yet there has been comparatively little effort spent on how to effectively integrate multiple visual cues in an overall detection and tracking system, as guided by quantitative system (component) analysis. Furthermore, there have been few instances in which presented concepts were backed up by extensive experimental validation.

Section 3 presents the proposed pedestrian system called PROTECTOR. The detection component involves a cascade of modules, each utilizing complementary visual criteria to successively narrow down the image search space, balancing robustness and efficiency considerations. Four detection modules are considered: (sparse) stereo-based ROI generation, shape-based detection, texture-based classification and (dense) stereo-based verification. These are complemented by a tracking module. We only provide short descriptions of the individual modules, as most were described previously. We focus instead on explaining *why* certain modules were selected and *how* they were integrated in the overall system. Among others, we cover a novel mixture-of-experts architecture, involving several texture-based component classifiers, which are weighted by the outcome of shape matching.

We obtain quantitative insights in the performance of the independent modules and their interaction on the basis of Receiver Operator Characteristics (ROCs). Section 4 describes a system parameter optimization technique to incrementally combine individual ROCs into an overall optimized ROC. After choosing in the latter an application-dependent ROC point, this method allows to automatically determine the suitable module parameter settings. Any application-specific bound on computational cost can be incorporated as a hard constraint in the optimization procedure. This systematical approach to parameter optimization is undoubtedly preferable to trial-and-error parameter tuning, so frequently employed previously.

Section 5 introduces a methodology for the validation of an integrated pedestrian detection and tracking system. Quantitative performance results of the proposed PROTECTOR system are presented based on many thousands of images from driving in real urban traffic. Illustrating the benefit of the stereo vision module, we compare the PROTECTOR system with a mono version. Section 6 puts the resulting system performance in context to that of other published systems and lists possible improvements. We conclude in Section 7.

Although we primarily cover people detection in the context of intelligent vehicles, the presented techniques could also be adapted to other important domains, such as surveillance or service robots.

2. Previous Work

Many interesting approaches for pedestrian detection have been proposed, for a survey see (Gavrila, 2001). Most work has pursued a learning-based approach, bypassing a pose recovery step and describing human appearance directly in terms of simple low-level features from a region of interest (ROI).

A number of ways exist to obtain such ROIs. Standard background subtraction, as frequently used in surveillance applications, is unsuitable because of the moving camera. Viable alternatives include sliding windows, detection of independently moving objects, and stereo-based obstacle detection.

The sliding window approach shifts ROI windows of all possible sizes, at all locations over the images while performing feature extraction and pattern classification. The brute-force approach in combination with powerful classifiers (i.e. Mohan et al., 2001; Papageorgiou and Poggio, 2000; Zhao and Thorpe, 2000) is currently computationally too intensive for real-time application. Recently however, Viola et al. (2005) demonstrated an efficient variant of the sliding window technique, which involves a detector cascade using simple appearance and motion filters (similar to the Haar-wavelets). Simpler detectors, with a smaller number of features, are placed earlier in the cascade

and more complex detectors later. At each detector stage, AdaBoost (Duda et al., 2001) incrementally selects those features with the lowest weighted error on the training set, until a user-supplied correct and false-detection rate is achieved on a validation set.

Another approach for obtaining ROIs involves detecting independently moving objects in monocular images. This approach typically assumes translatory camera motion and detects deviations in the optical flow field from the expected background motion (Elzein et al., 2003; Polana and Nelson, 1994; Thompson and Pong, 1990).

A third effective approach for obtaining ROIs is by stereo vision. Zhao and Thorpe (2000) obtain a foreground region by clustering in the disparity space. Broggi et al. (2003) and Grubb et al. (2004) consider the x- and y-projections of the disparity space following the 'V-disparity' technique (Labayrade et al., 2002).

Once ROIs have been established, different combinations of features and pattern classifiers can be applied to make the distinction between pedestrian and non-pedestrian. For example, Broggi et al. (2004) employ vertical symmetry features. Zhao and Thorpe (2000) apply a high-pass filter and normalize the ROI for size, thereafter applying a feed-forward neural network. Papageorgiou and Poggio (2000) pioneered the use of Haar-wavelet features in combination with a Support Vector Machine (SVM); this approach was subsequently adapted by Elzein et al. (2003) and others.

Component-based approaches have been utilized to reduce the complexity of pedestrian classification. Shashua et al. (2004), for instance, extract a feature vector from each of 9 fixed sub-regions. Other approaches attempt to directly identify certain body parts. Mohan et al. (2001), for example, extend the work of Papageorgiou and Poggio (2000) 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. Very recently, additional attempts have been made towards reducing classification complexity by manually separating the pedestrian training set in non-overlapping sub-sets (i.e. based on pedestrian heading direction) Shashua et al. (2004), Grubb et al. (2004), and Shimizu and Poggio (2004).

There has been separately extensive work on pedestrian tracking. Representations such as Active Shape Models (Baumberg and Hogg, 1994; Cootes et al., 1995; Philomin et al., 2000), shape exemplars (Stenger et al., 2003; Toyama and Blake, 2001) and color blobs (Heisele and Wöhler, 1998) have been combined with Kalman- (Baumberg and Hogg, 1994; Cootes et al., 1995) or particle filtering (Philomin et al., 2000; Stenger et al., 2003; Toyama and Blake, 2001) approaches. Given temporal ROI data, previous work has detected people walking laterally to the viewing direction, either using the periodicity cue (Cutler and Davis, 2000; Polana and Nelson, 1994) or by learning the characteristic lateral gait pattern (Heisele and Wöhler, 1998).

Orthogonal to the above, there has been increased interest in the FIR domain (Broggi et al., 2004; Fang et al., 2003; Liu and Fujimura, 2003) driven by the arrival of cheaper, uncooled cameras. Detecting pedestrians by their body heat is attractive, certainly when considering images shot on a cold winter night, where the pedestrians stick out as white regions before a black background. However, the situation is less appealing on sunny summer days, when there is an abundance of additional hot spots. In the latter case, one needs to resort to a similar set of detection techniques as in the visible domain.

Table 1 summarizes the main pedestrian detection systems, distinguished by image sensor type, area of coverage, detection performance, use of tracking, processing speed and test set, where-ever specified by the respective authors. As we will discuss in Section 6, performance comparisons are hazardous, since data sets are typically small (and diverse) (Mohan et al., 2001), Papageorgiou and Poggio (2000), Viola et al. (2005), Zhao and Thorpe (2000) and often relate to single computational components (e.g. classification). Even where test data is more abundant, many important test criteria remain unspecified (e.g. exact coverage area, localization tolerances, data assignment rule). The latter motivates us to spell out a more detailed test methodology (Section 5.1).

The issue of automatic system parameter optimization has so far not been covered in the context of pedestrian detection, to our knowledge. A large body of literature exists on numerical methods dealing with the minimization of non-linear objective functions. Such approaches are impractical for the given problem of optimizing a complex system, because (a) a single evaluation of the objective function is computationally expensive, and (b) the number of parameters involved is relatively high. As a way out, authors have either tried to model system behavior by simpler functions, e.g. Bayesian networks (Sarkar and Chavali, 2000), or developed specialized solutions for the particular problem

Table 1. Overview of current pedestrian detection systems. Detection performance derived by visual inspection of ROC graphs in respective publications (approximation). "f.pos." denotes false positives.

Authors	Sensor type	Coverage area	Detection Performance (per frame)	Tracking (per trajectory)	Processing speed	Test set
Papageorgiou and Poggio (2000)	mono, visible	–	70% correct, 0.15 f.pos. (full set color wavelets) 70% correct, 3 f.pos. (red. grey wavelet set)	no	20 min (200 MHz PC) 10 Hz	123 ped. images, 50 non-ped. images
Mohan et al. (2001)	mono, visible	–	85% correct, 0.03 f.pos	no	–	123 ped. images 50 non-ped images
Viola et al. (2005)	mono, visible	–	80% correct, 0.5 f.pos	no	4 Hz (2.8 GHz PC)	2 sequences of 2000 images (stat. camera)
Elzein et al. (2003)	mono, visible	–	69% correct, 61% precision	no	95s (500 MHz PC)	16 ped. images + few sequences
Shashua et al. (2004)	mono, visible	3–25 m	96% correct, 5.6×10^{-6} f.pos (inward moving) 93% correct, 8.3×10^{-5} f.pos (stationary in-path) 85% correct, 2.9×10^{-3} f.pos (stationary out-path)	yes	10 Hz	1–5 hrs urban drive
Zhao and Thorpe (2000)	stereo, visible	–	85% correct, 3% f.pos (per stereo ROI)	no	3–12 Hz (450 MHz PC)	–
Bertozzi et al. (2004)	stereo, visible	–	83% correct, 0.46 f.pos	no	–	1500 images, 1897 ped. instances
Grubb et al. (2004)	stereo, visible	–	84% correct, 0.004 f.pos	yes	23 Hz (2.4 GHz PC)	2500 images, 14 different peds.
Gavrila and Munder (current paper)	stereo, visible	10–25m in front up to 4m lateral	61–81% correct, $[0.7–23] \times 10^{-3}$ f.pos (speed unoptimized) 59–75% correct, $[1.0–27] \times 10^{-3}$ f.pos (speed optimized)	78–100% cor., 0.3–3.5 f.pos/min 78–100% cor., 0.4–5.1 f.pos/min	3–7 Hz 7–15 Hz (2.4 GHz PC)	sequence of 17390 images, 694 ped. instances 17067 non-ped. images
Fang et al. (2003)	mono, FIR	–	84% correct, 19% f.pos (summer) 92% correct, 3% f.pos (winter)	no	–	289 ped. images

at hand. Within this work, we aim to exploit the cascade coupling of system modules.

Cascaded classifiers (Viola et al., 2005) have recently received increasing interest due to their computational efficiency, and a number of publications addressed their optimization. For example, Sun et al. (2004) and Luo (2005) 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 (2004), 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 (2001) 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 (Huo and Chen, 2004; Provost and Fawcett, 2001) by developing a technique to sequentially optimize a cascade of complex system modules, each controlled by a number of parameters, and by incorporating user-defined constraints.

This paper builds on earlier work described in Gavrila et al. (2004). We consider the main contributions of this paper the integration of modules in a

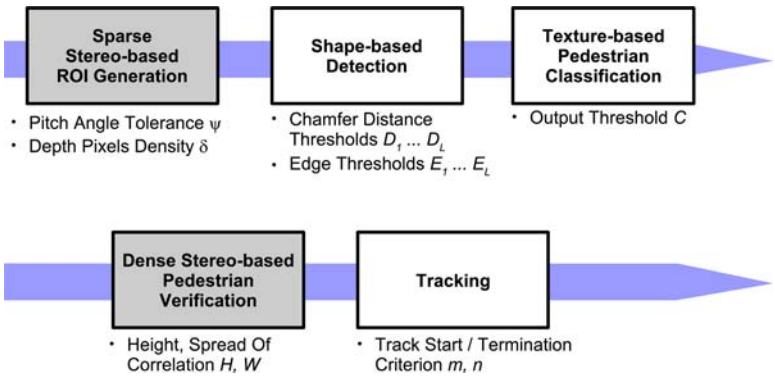

Figure 2. Overview of the PROTECTOR modules. Modules shaded grey depend on stereo imaging. Parameters selected for optimization are listed under the corresponding module.

multi-cue system for pedestrian detection and tracking (Section 3.2), and furthermore, the systematic procedure for parameter setting and system optimization (Section 4).

3. The PROTECTOR System

See Fig. 2 for an overview of the modules of the PROTECTOR system. Modules which rely on stereo are shown in shaded grey, they would be discarded in a mono version of the system. Figure 2 also lists the main module parameters subject to optimization, as covered in Section 4.

A first system design consideration involves the use of stereo vision. Shashua et al. (2004), for example, questions the benefit of depth disparity cues for segmenting out pedestrians in urban traffic due to heavy disparity clutter. Approaches that are overly "bottom-up", such as those clustering sparse depth (e.g. Zhao and Thorpe, 2000) or those based on detecting obstacles using the V-disparity technique (Broggi et al., 2003; Grubb et al., 2004), indeed tend to break down in such cluttered scenarios, when pedestrians are not well separated from other objects in the environment and/or when road surface is significantly occluded by other obstacles. In such cases, pedestrian features are frequently merged with other objects in the environment and a subsequent pattern classifier has difficulties dealing with the resulting spatial misalignment. We overcome this problem by not relying on depth segmentation, instead by instantiating "top-down" pedestrian matching based on the (raw) depth data, see Section 3.2.1. In the experiments in complex urban traffic, we show that the stereo version of PROTECTOR significantly outperforms its mono variant, as result of the improved foreground-background separation (Section 5).

Another notable design choice, absent in other systems, is the use of a shape detection module prior to texture-based pedestrian classification. Shape is a powerful cue for the problem at hand; its property to reduce the variation in pedestrian appearance as induced by lighting or clothing, makes it an appealing "filter" before the use of texture, especially when it involves a hierarchical matching approach which efficiently provides accurate target localization. Moreover, shape information can be used to instantiate pose-specific texture classifiers, for increased recognition performance (Section 3.2.2).

3.1. Individual Modules

3.1.1. 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 (Bouguet, 2000). 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 Franke (2000) (alternate choices would have been possible). The outcome is a relatively sparse disparity map, see upper left part of Fig. 5.

3.1.2. Shape-Based Detection. We follow an exemplar-based approach to shape-based pedestrian detection, as previously described in Gavrila and Philomin (1999). Pedestrians are represented by a set

of training shapes or templates which ideally cover the set of object appearances due to transformations (i.e. scale) and intra-class variance (i.e. different pedestrians, different poses). At the core, we perform template matching based on the (chamfer) distance transform (Borgefors, 1986). This correlation approach is quite robust to missing features due to segmentation problems (i.e. edge "gaps") or occlusion. Furthermore, the distance transforms has the advantage to provide a gradual measure of shape dissimilarity. This allows the use of an efficient search algorithms to "lock onto" the correct solution, as discussed below.

Several thousand of exemplars have been collected to describe the pedestrian shape distribution. We use a template hierarchy to efficiently match whole sets of templates, see lower right part of Fig. 5. Offline, this hierarchy is constructed automatically from available example templates, in 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 (Gavrila and Philomin, 1999).

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 Gavrila and Philomin (1999), 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 $E_1, \ldots, E_3$ control the amount of ROIs passed onto the next stage; they are determined automatically using the optimization approach described in Section 4.

3.1.3. Texture-Based Pedestrian Classification. Whereas the preceeding 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. The selection of the particular type of classifier was guided by the need to represent a complex decision boundary in high dimensional feature space with only a relatively small training set at the disposal, see Table 2 (negative samples consisted of the false positives of the cascade up to this point from previous bootstrapping steps (Papageorgiou and Poggio, 2000)).

Table 2. Data sets used for training and testing of the texture classifier. The data sets have been further increased by mirroring and shifting the bounding box position in *x*- and *y*-direction by few pixels (accounting for small remaining localization errors).

	Pedestrians	Non-Pedestrians
Training set	5752	20434
Test set	4632	24668

The SVM (Mohan et al., 2001; Papageorgiou and Poggio, 2000), in combination with PCA for dimensionality reduction, fits the above profile, but we decided against it after an extensive experimental evaluation (Munder and Gavrila, 2006) showed its classification performance to be similar yet its processing requirements to be substantially higher (by half order of magnitude) than the approach we finally selected: a neural network with local receptive fields (Wöhler and Anlauf, 1999). The use of local receptive fields with shared weights reduces the degrees of freedom for the training process compared to a fully-connected feed-forward network (Zhao and Thorpe, 2000), allowing relatively small training sets. As Munder and Gavrila (2006) shows, the network is nevertheless able to capture relevant feature information from a local neighborhood.

3.1.4. 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 Fig. 3. 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 both the maximum of the

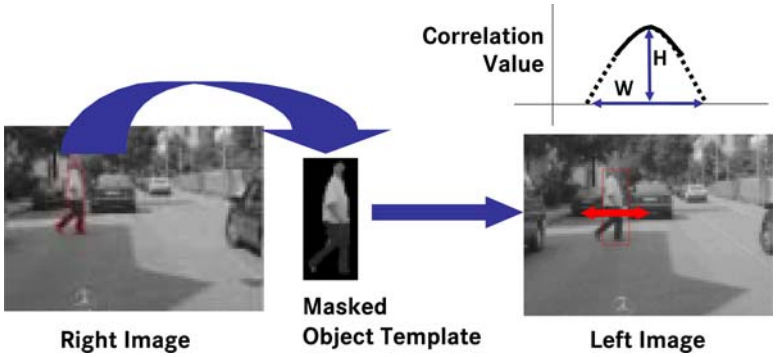

Figure 3. Stereo Verification: The shape template masks out background pixels for a dense cross-correlation between both stereo images within a certain disparity search range. A threshold is enforced on both height and spread of the resulting correlation function.

polynomial is above, and the normalized "spread" of the polynomial is below a threshold (the latter is a measure of localization confidence). The two associated parameters, H and W, control the amount of ROIs passed onto the next stage; they are determined automatically using the optimization approach described in Section 4.

For some intuition in the above, see Fig. 4. The candidate solutions shown in black may have passed the stereo, shape and texture module in the cascade. Based on the ground plane constraint, however, they are estimated at 10–15 m 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).

3.1.5. Tracking. Tracking allows us to overcome gaps in detection, to suppress spurious measurements and to obtain trajectory information for a subsequent risk assessment module. Mainly because of computational cost and because shape variations have already been 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.

Figure 4. Stereo Verification: Examples of removal of background-corrupted detections (accepted solutions shown white, rejected black).

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 (Kuhn, 1955). 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. For the latter, we use the chamfer distance (which was pre-computed off-line for all pairs of shape exemplars in the process of the construction of the exemplar hierarchy, see Section 3.1.2). 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, control system output; they are determined automatically using the optimization approach described in Section 4.

3.2. Module Integration

We aim for a tight integration of modules in Fig. 2, incorporating as much information as possible from a previous module into the next one, for efficiency and recognition performance reasons.

3.2.1. Depth and Shape. Figure 5 illustrates how the depth and shape modules are integrated, while taking into account the ground plane constraints (i.e. flat world assumption, pedestrian on the ground). The outcome

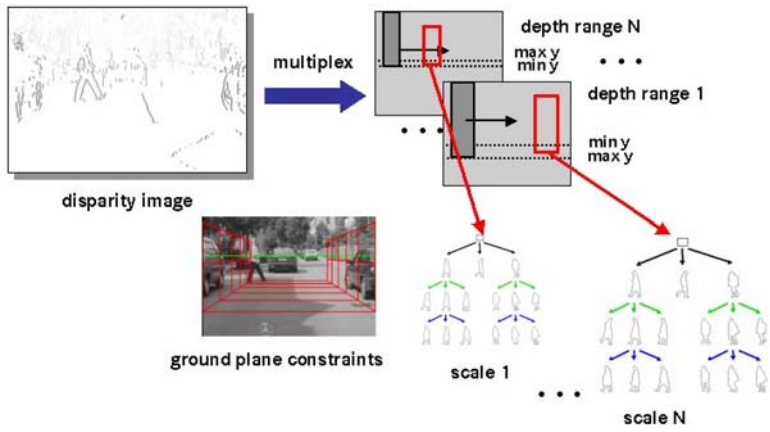

Figure 5. Integration of stereo, shape and ground plane constraints at "hand-over" of obstacle detection to pedestrian classification.

of the stereo module, the disparity map, is multiplexed into N discrete depth ranges. The associated 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 determined automatically using the optimization approach described in Section 4.

For a tight integration, the exemplar-based approach of Section 3.1.2 computes separate template hierarchies corresponding to a discrete set of pedestrian sizes. Links are established off-line between possible ROI window sizes/locations and the template hierarchies of corresponding pedestrian size, allowing an efficient online "hand-over". See Fig. 5. A similar technique applies the mono case; this takes into account solely the ground plane constraints, without benefitting from the pruning of window locations based on the existence of sufficient depth features.

3.2.2. Shape and Texture. Texture-based classification of pedestrians is challenging because of the high variability of the target class, which arises from three sources: foreground texture (induced by the pedestrian's clothing and illumination), shape (body pose and viewing direction), and background texture. One approach to tackle this problem known from the literature is to partition the feature space into regions of reduced variability, and to train separate classifiers, or "local experts" for each sub-region (Jacobs et al., 1991). This *mixture-of-experts* architecture is particularly suitable for the problem at hand because partitioning information is already available from the shape detection module. Hence, instead of trying to learn a clustering of the texture patterns anew, we exploit the shape matching results as an additional source of information for the texture classification module. In order to compensate for shape matching errors, as e.g. caused by background clutter, we derive a probabilistic assignment of texture patterns to shape clusters, used to weight the results of the individual classifiers. Figure 6 provides an illustration of this architecture.

In Section 3.1.2, a hierarchy of shape templates has been constructed. At the top level, this hierarchy defines a partitioning of all underlying shape templates into a number of clusters, each roughly representing a distinct body pose. Let $\mathcal{C}_i$, $i = 1 \ldots K$, denote the shape clusters, K being the number of template nodes at the top level of the tree. In order to assign a texture pattern $\mathbf{x}$ to a cluster $\mathcal{C}_i$, we define a distance measure between $\mathbf{x}$ and $\mathcal{C}_i$ by

$$D_i(\mathbf{x}) = \min_{\mathbf{t} \in \mathcal{C}_i} D_{\text{Chamfer}}(\mathbf{t}, \mathbf{x}), \tag{1}$$

i.e., the minimum chamfer distance (Borgefors, 1986) over all shape templates $\mathbf{t}$ in cluster $\mathcal{C}_i$. By modeling the cluster-conditional density function as an exponential distribution,

$$p(\mathbf{x} \mid \mathcal{C}_i) \propto \alpha_i e^{-\alpha_i D_i(\mathbf{x})}, \tag{2}$$

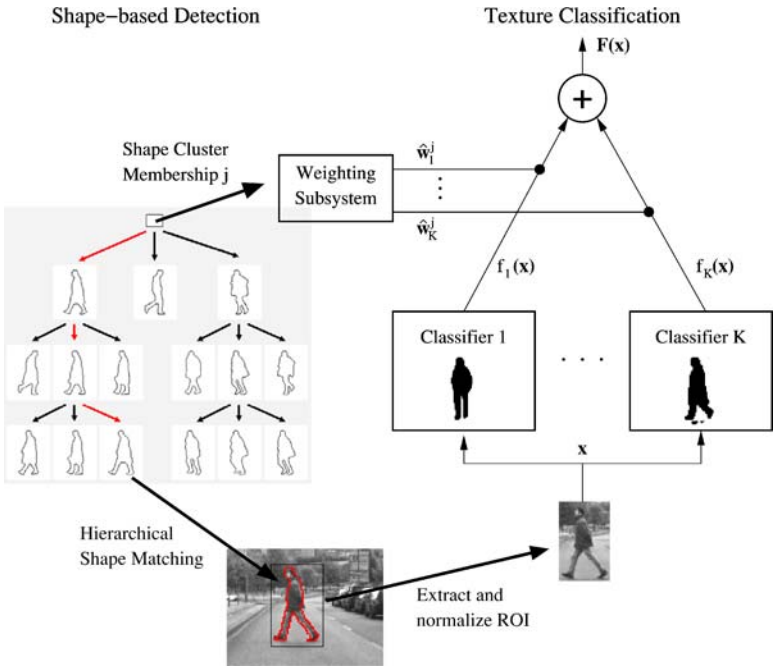

Figure 6. Integration of shape-based localization and texture classification. The shape cluster membership of the matching shape template determines the weighting of the K classifiers in a *mixture-of-experts* architecture.

and assuming equal prior probabilities, we derive the probability of pattern $\mathbf{x}$ falling into cluster $\mathcal{C}_i$ according to Bayes formula as

$$w_i(\mathbf{x}) \equiv \frac{\alpha_i e^{-\alpha_i D_i(\mathbf{x})}}{\sum_{j=1}^{K} \alpha_j e^{-\alpha_j D_j(\mathbf{x})}} \approx P(\mathcal{C}_i \mid \mathbf{x}). \tag{3}$$

Parameters α_i are determined on the training set via maximum likelihood.

One could partition the training set into K disjoint subsets based on these probabilities, and train a separate classifier on each subset. We decided against it as preliminary experiments have shown degraded performance, possibly due to wrong sample-to-cluster assignment or the reduced number of training samples remaining in each cluster. Instead, we train K classifiers f_i on the same full training set, but apply the cluster membership probability $w_i(\mathbf{x})$ as a weight to each training example. The output of each classifier can be considered as an approximation of the posterior probability that pattern $\mathbf{x}$ represents a pedestrian:

$$f_i(\mathbf{x}) \approx P(\text{pedestrian} \mid \mathcal{C}_i, \mathbf{x}). \tag{4}$$

The final classification result is consequently given by the weighted sum of the cluster classifiers:

$$F(\mathbf{x}) = \sum_{i=1}^{K} w_i(\mathbf{x}) f_i(\mathbf{x}) \tag{5}$$

$$\approx \sum_{i=1}^{K} P(\mathcal{C}_i \mid \mathbf{x}) P(\text{pedestrian} \mid \mathcal{C}_i, \mathbf{x})$$

$$= P(\text{pedestrian} \mid \mathbf{x}). \tag{6}$$

Classification result $F(\mathbf{x})$ is compared to a user-supplied threshold C, which controls the amount of ROIs passed onto the next stage. It is determined automatically using the optimization approach described in Section 4.

Determination of the weights $w_i(\mathbf{x})$ is rather time-consuming as it requires the computation of the chamfer distance of a previously unseen texture pattern to all shape templates (Eqs. (1) and (3)). For online processing, we approximate $w_i(\mathbf{x})$ by the expectation of the weight vector distribution of samples in the training set that fall in that particular cluster:

$$\hat{w}_i^j = \mathcal{E}_{\mathcal{D}_j}[w_i(\mathbf{x})] \tag{7}$$

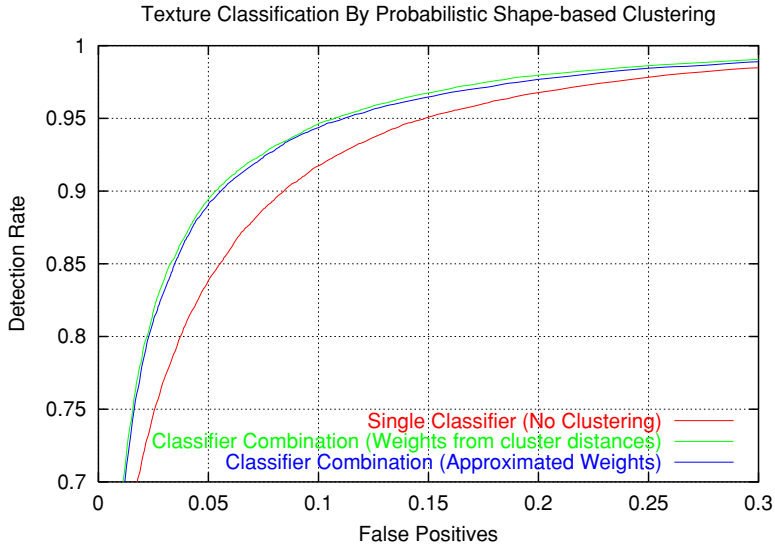

Figure 7. ROC curves of the texture classification module, determined on the test set. The solid curve shows the performance of the classifier as given in Eq. (5), whereas approximated weights according to Eq. (7) have been used for the dashed curve. For comparison, ROC performance for a single classifier without clustering is shown as well.

where

$$\mathcal{D}_j = \{\mathbf{x} \in \mathcal{D} \mid j = \arg \min_{j'=1\ldots K} D_{j'}(\mathbf{x})\}\,. \qquad (8)$$

In this manner, $\hat{w}_i^j$ can be pre-computed off-line, and index j follows online from hierarchical shape matching.

Figure 7 illustrates the benefit of the proposed mixture-of-experts architecture. The ensemble of texture classifiers attuned to particular pedestrian poses clearly outperforms the single classifier trained on the entire data set. Furthermore, approximating $w_i(\mathbf{x})$ by $\hat{w}_i^j$ does not appreciably affect performance.

4. System Optimization

After having introduced the individual modules, we turn to modeling their interaction and to finding good overall parameter settings in a systematic and optimized fashion. Figure 2 shows the system modules to be optimized and their respective parameters. In total, 13 parameters which mainly influence system performance and processing speed have been identified for optimization. Our optimization objective is the entire system ROC curve in order to remain flexible regarding the ROC point to be used in a particular application. In order for the outcome to be practically usable, we integrate the inevitable processing constraints (i.e. average processing time per frame) into the optimization process.

The idea can be sketched as follows: We search over a set of parameters by computing a ROC point, i.e. the false positive and the detection rate, for each parameter setting, as illustrated in Fig. 8. After filtering this set of ROC points to meet user-defined constraints, the desired system ROC curve is given by its frontier (black curve in Fig. 8). For computational reasons, the system modules are successively included into the optimization process.

The theory of the optimization approach is given in the next subsection, implementation details and the incorporation of constraints are described in Section 4.2.

4.1. *Sequential Parameter Optimization*

We first consider the problem of adjusting the parameters of a cascade of n system modules. Each module represents a classifier that assigns an input pattern $\mathbf{x}$ to either the target or non-target class, depending on some parameter vector $\mathbf{q}_i$, $i = 1,\ldots,n$, subject to optimization. The output of the entire cascade is "target class" if, and only if, all classifiers agree on that decision. If $d_i(\mathbf{x}, \mathbf{q}_i) \in \{0, 1\}$ de-

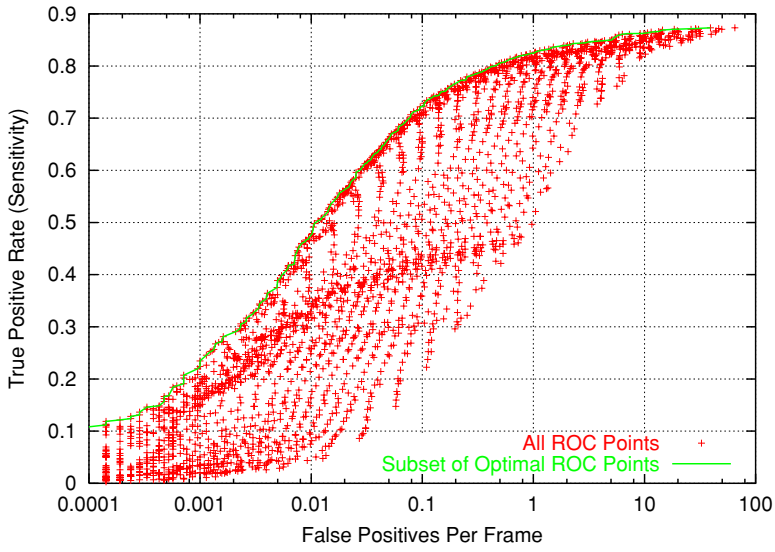

Figure 8. Illustration of the optimal subset of ROC points: A cloud of ROC points of the texture classification module has been generated by varying its output threshold as well as the parameters of the preceeding shape detection module, shown by grey crosses. The subset of optimal ROC points (for this particular module) is depicted by the black curve.

notes the output of the ith classifier where 1 and 0 denote the target class and non-target class, respectively, then the result of the entire cascade is given by $D_{1:n}(\mathbf{x}, \mathbf{q}_{1:n}) = d_1(\mathbf{x}, \mathbf{q}_1)d_2(\mathbf{x}, \mathbf{q}_2)\ldots d_n(\mathbf{x}, \mathbf{q}_n)$, where $\mathbf{q}_{1:n} = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$ denotes the concatenation of all parameter vectors. Furthermore, let $\mathcal{Q}_i$ denote the set of admissible parameter vectors $\mathbf{q}_i$, and $\mathcal{Q}_{1:n} = \mathcal{Q}_1 \times \cdots \times \mathcal{Q}_n$.

The performance of the cascade for a given parameter vector $\mathbf{q}_{1:n}$ is characterized by a pair of false positive rate and corresponding detection (or hit) rate,

$$F_{1:n}(\mathbf{q}_{1:n}) = \Pr[D_{1:n}(\mathbf{x}, \mathbf{q}_{1:n}) = 1 | \text{non-target pattern } \mathbf{x}],$$
$$H_{1:n}(\mathbf{q}_{1:n}) = \Pr[D_{1:n}(\mathbf{x}, \mathbf{q}_{1:n}) = 1 | \text{target pattern } \mathbf{x}],$$

which represents a point in ROC space. The set $\mathcal{Q}_{1:n}$ of all admissible parameter vectors generates a set of ROC points, of which we seek the dominating, or *Pareto optimal* (Boyd and Vandenbergh, 2004) ones along the frontier, see Fig. 8 for an illustration. More formally, we seek for the subset $\mathcal{Q}^\star_{1:n} \subset \mathcal{Q}_{1:n}$ of parameter vectors $\mathbf{q}_{1:n}$ for which there is no other parameter vector that outperforms both, false positive rate and the detection rate rate: (temporarily dropping subscripts)

$$\mathcal{Q}^\star = \{\mathbf{q} \in \mathcal{Q} \,|\, \forall_{\mathbf{q}' \in \mathcal{Q}} F(\mathbf{q}') \geq F(\mathbf{q}) \vee H(\mathbf{q}') \leq H(\mathbf{q})\}. \tag{9}$$

Performing such an optimization simultaneously for all parameters, however, becomes computationally infeasible even for a moderate number of cascade nodes and parameters. But *suppose* the following precondition holds: Each (concatenated) parameter vector $\mathbf{q}_{1:i} \in \mathcal{Q}^\star_{1:i}$ optimal for a cascade of nodes $1, \ldots, i$ is also optimal if only nodes $1, \ldots, i-1$ are considered. That is,

$$\mathbf{q}_{1:i} = (\mathbf{q}_{1:i-1}, q_i) \in \mathcal{Q}^\star_{1:i} \;\Rightarrow\; \mathbf{q}_{1:i-1} \in \mathcal{Q}^\star_{1:i-1}, \tag{10}$$

for $i = 2, \ldots, n$. Then, the search space $\mathcal{Q}_{1:n}$ in Eq. (9) can be restricted to contain the solution $\mathcal{Q}^\star_{1:n-1}$ of a truncated cascade. Applying this idea recursively leads to the method of sequentially computing $\mathcal{Q}^\star_{1:2}$, $\mathcal{Q}^\star_{1:3}$, $\ldots$, $\mathcal{Q}^\star_{1:n}$, where the search space $\mathcal{Q}'_{1:i}$ in each step can be restricted to contain the preceeding solution:

$$\begin{aligned} \mathcal{Q}'_{1:i} &= \mathcal{Q}^\star_{1:i-1} \times \mathcal{Q}_i, \\ \mathcal{Q}^\star_{1:i} &= \{\mathbf{q} \in \mathcal{Q}'_{1:i} \,|\, \forall_{\mathbf{q}' \in \mathcal{Q}'_{1:i}} F(\mathbf{q}') \geq F(\mathbf{q}) \vee H(\mathbf{q}') \\ &\qquad \leq H(\mathbf{q})\}. \end{aligned} \tag{11}$$

It is easy to show that the precondition (10) is met when the cascade nodes are independent, i.e. if

$$F_{1:n}(\mathbf{q}_{1:n}) = \prod_{i=1}^{n} f_i(\mathbf{q}_i), \quad H_{1:n}(\mathbf{q}_{1:n}) = \prod_{i=1}^{n} h_i(\mathbf{q}_i), \tag{12}$$

Initialize $\mathcal{Q}^{\star}_{1:0} = \emptyset$, the empty set.
For $i = 1, \ldots, n$:

- Select $\mathcal{Q}^{s}_{1:i-1} \subset \mathcal{Q}^{\star}_{1:i-1}$ by subsampling at most M ROC points from $\mathcal{Q}^{\star}_{1:i-1}$, and let $\mathcal{Q}'_{1:i} = \mathcal{Q}^{s}_{1:i-1} \times \mathcal{Q}_i$.
- 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 evaluate the constraint function $C_i(\mathbf{q}_{1:i})$.
- Apply the user-defined constraint C_i:

$$\mathcal{Q}^{C}_{1:i} = \{\mathbf{q}_{1:i} \in \mathcal{Q}'_{1:i} \,|\, C_i(\mathbf{q}_{1:i}) \geq 0\}\,.$$

- Determine the subset of optimal parameter vectors:

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

- Continue with $i \leftarrow i + 1$.

Return the optimized parameter set $\mathcal{Q}^{\star}_{1:n}$.

Figure 9. System optimization procedure.

where $f_i(\mathbf{q}_i)$ denotes the false positive rate of node i which is now only affected by its associated parameters $\mathbf{q}_i$ and independent of all others, and analogously for $h_i(\mathbf{q}_i)$. Substituting F and H in Eq. (9) with these products immediately implies (10). In other words, the successive computation scheme (11) is provably optimal if the cascade nodes are independent.

In practice, the 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. An example is given in Section 5.2.

4.2. Implementation

In many applications there are constraints that restrict allowable parameter settings. For example, an (online) application will involve bounds on allowable processing speed. For real-time pedestrian detection on-board a moving vehicle, we place an upper limit on the average processing time per frame.

In general, let a parameter vector $\mathbf{q}_{1:i}$ under consideration at optimization be required to meet some user-defined constraint $C_i(\mathbf{q}_{1:i}) \geq 0$. Then the search space $\mathcal{Q}'_{1:i}$ in Eq. (11) is restricted to those parameter vectors that meet the constraint, i.e. it is replaced by

$$\mathcal{Q}^{C}_{1:i} = \{\mathbf{q}_{1:i} \in \mathcal{Q}'_{1:i} \,|\, C_i(\mathbf{q}_{1:i}) \geq 0\}. \tag{13}$$

Our particular constraint of limiting the average processing time per frame is implemented as follows. At optimization stage i and parameter $\mathbf{q}_{1:i}$ under consideration, processing time $T^{\text{consumed}}_{1:i}(\mathbf{q}_{1:i})$ of modules $1, \ldots, i$ can be measured, while the processing time of the remaining modules $i+1, \ldots, n$ is estimated to be a linear function of the (average) number of ROIs to be processed by these modules. The latter is given by the number of outputs $k_i(\mathbf{q}_{1:i})$ of module i, while the parameters of the linear model, $T^{\text{fix}}_{i+1:n}$ and $T^{\text{obj}}_{i+1:n}$, are determined in advance by linear regression over the training set. With T^{limit} denoting the user-defined processing time limit, C_i is given by

$$C_i(\mathbf{q}_{1:i}) = T^{\text{limit}} - T^{\text{consumed}}_{1:i}(\mathbf{q}_{1:i}) - T^{\text{fix}}_{i+1:n} - k_i(\mathbf{q}_{1:i}) T^{\text{obj}}_{i+1:n}. \tag{14}$$

Two simplifications are made in order for the optimization algorithm to be computationally feasible. The sets of admissible parameters $\mathcal{Q}_i$ are required to be finite, so that Eq. (11) can be implemented as a simple search over the restricted set $\mathcal{Q}^{C}_{1:i}$. Finiteness is achieved via discretization. Secondly, we limit the number of ROC points to be considered from the preceeding solution set. At most M ROC points are selected from $\mathcal{Q}^{\star}_{1:i-1}$, drawn uniformly along the ROC curve. The resulting system parameter optimization algorithm is listed in Fig. 9.

Table 3. "Real World" statistics.

	Run1	*Run2*
Total images	21053	17390
Images containing pedestrians	1021	855
Events (pedestrian instances): all/risky	733/112	694/89
Event trajectories: all/risky	45/17	50/10

5. Experiments

We tested the PROTECTOR system on data from urban traffic environment. Two video sequences (*Run1* and *Run2*) were recorded on the same route through suburbia and inner city of Aachen, Germany, lasting 27 min and 24 min, respectively. 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 3. See Fig. 10 for a view inside the vehicle demonstrator.

Run1 was used to perform the parameter optimization as described in Section 4, system evaluation was done on *Run2*. None of these sequences was used for the training process of a system module (as shape or texture pattern examples), *Run1* merely provided a basis for the adaption of our system to the urban environment.

Figure 10. Inside the vehicle demonstrator: stereo cameras visible lateral to rear-mirror, display for system prototyping.

5.1. Test Methodology

At the core, system evaluation involves comparing entries from ground truth with system output, both related to 3D object position relative to the vehicle (we prefer to evaluate the system in 3D rather than in image space, because application-specific considerations can more easily be described in 3D terms). Ground truth data is obtained by a human operator diligently labeling objects in monocular images, and by using scene knowledge to back-project into 3D. For the case of pedestrians, the latter means making the "flat world" assumption coupled with the reasonable conjecture that the pedestrian feet stand on the ground plane.

In comparing system output and ground truth, we consider two performance metrics: **sensitivity** and **precision**. Sensitivity relates to the percentage of true solutions that were found by the system, whereas precision relates to the percentage of system solutions that were correct. A sensitivity and precision of 100% is ideal: the system finds all real solutions and produces no false positives. For additional insight, we consider the two performance metrics on both the frame- and trajectory-level. For the latter, we further distinguish two types of trajectories: "class-B" and "class-A" trajectories that have at least one entry or at least 50% of their entries matched, respectively. Thus, all "class-A" trajectories are also "class-B" trajectories, but "class-A" trajectories pose stronger detection demands that might be necessary in some applications. Furthermore, the following items need to be specified.

Sensor Coverage Area. The sensor coverage area represents the space surrounding the vehicle where the defined object detection capability is required. Outside this area, we consider detection capability optional in the sense that the system is not rewarded/penalized for correct/false/missing detections. The PROTECTOR sensor coverage area is shown in Fig. 11.

Localization Tolerance. Given an object detected by the system at a certain location ("alarm"), and given a true object location ("event"), the localization tolerance is the maximum positional deviation that still allows us to count the alarm as a match. This localization tolerance is the sum of an application-specific component (how precise does the object localization have to be for the application) and a component related to measurement error (how exact can we determine true object location).

For the PROTECTOR field tests, we define object localization tolerance as percentage of distance, for

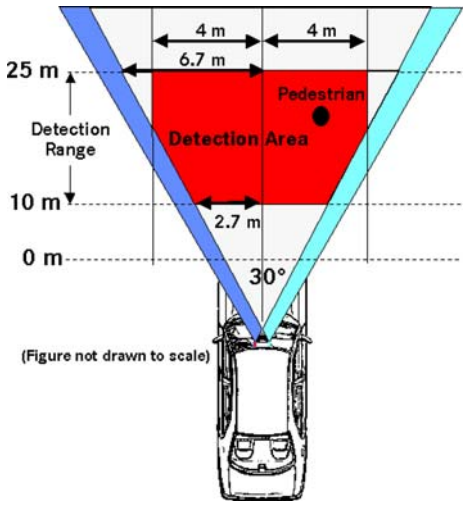

Figure 11. PROTECTOR system coverage area.

lateral and longitudinal direction (X and Z), with respect to the vehicle. Regarding the application-specific component, values of $X_a = 5\%$ and $Z_a = 15\%$ appear reasonable; for example, this means that, at 20 m distance, we tolerate a localization error of ± 1 m and ± 3 m in the position of the pedestrian, lateral and longitudinal to the vehicle driving direction, respectively. Regarding the measurement-specific component, $X_m = 5\%$ and $Z_m = 15\%$ appear necessary (with the larger Z_m value to account for non-flat road surface and/or vehicle pitch in case of ground truth by monocular image labeling). For the PROTECTOR field tests, we then use overall tolerances of $X = 10\%$ and $Z = 30\%$.

Data Assignment. For the PROTECTOR application we allow many-to-many correspondences. An event is considered matched if there is at least one alarm matching it. In practice, this means that in the case a group of pedestrians walking sufficiently close together in front of the vehicle, the system does not necessarily have to detect all of them in isolation, it suffices if each true pedestrian is within the localization tolerance of a detected pedestrian.

5.2. *Validation of the Optimization Procedure*

The first test aims to validate the system optimization procedure described in Section 4 using real-world data. For comparison, we determine the true optimum by exhaustive search of the entire (discretized) parameter space. In order for this to be computationally feasible, we select only two out of the five system modules, namely the "Shape-based Detection" and "Texture-based Classification" modules. Furthermore, we fix the parameters of the non-leaf levels of the shape template hierarchy, so that three system parameters remain for optimization: the leaf-level edge binarization and chamfer distance thresholds E_L and D_L, and the classifier output threshold C, see Fig. 2. 11 discrete values are chosen for parameter E_L, and 100 for D_L and C, so that in total $111,000$ different parameter settings have to be considered for the exhaustive search, but only $11 \times 100 + 127 \times 100 = 13,800$ for the sequential approach, where 127 is the number of ROC points obtained in the first optimization step.

In addition, we apply the independent optimization procedure described by Luo (2005). Here, we first determine a single ROC curve for each of the two system modules. Since two parameters are involved for the first module, its ROC curve is computed from the Pareto front, Eq. (9). An overall optimization objective, constructed by considering log-scale ROC curves and by using the Lagrange multiplier method, is to minimize $\log(F(\mathbf{q})) - \lambda \log(H(\mathbf{q}))$. By assuming independence and substituting F and H with Eq. (12), this decomposes into independent optimization problems, one for each module, parametrized by the Lagrange multiplier λ. The overall ROC curve is then obtained by varying λ within the range $(0, \infty)$ and by choosing optimized parameters independently for each module, given by the point on their log-scale ROC curve with slope equal to λ.

ROC results for all three methods obtained on the optimization sequence *Run1* are given in Fig. 12. The performance of the sequential optimization procedure is almost identical to the optimal ROC curve obtained by exhaustive search, whereas results of the independent optimization approach are considerably degraded.

5.3. *PROTECTOR System Results*

We compare three variants of our system: The PROTECTOR system as described in Section 3 with and without processing time constraint, and a mono system, without time constraint, where the stereo-related modules have been discarded (Stereo-based ROI Generation and Pedestrian Verification, leaving only the ground plane as ROI constraint). The upper bound on average processing time was specified at 100 ms.

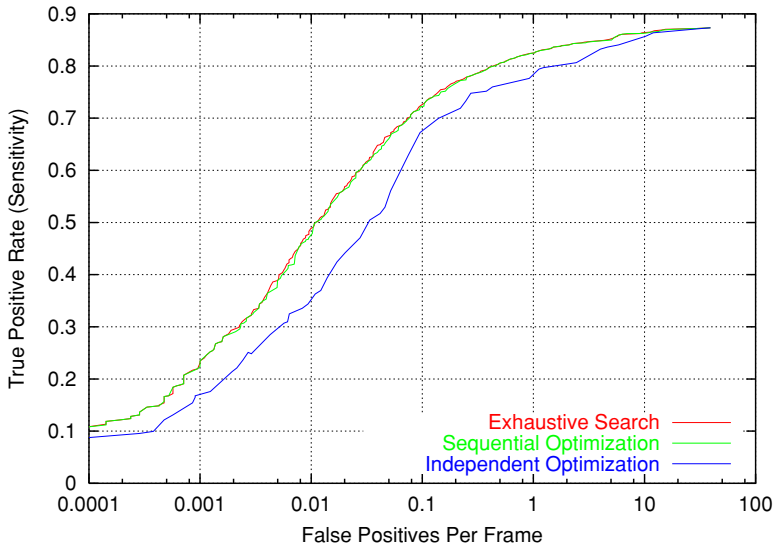

Figure 12. Comparison of three parameter optimization approaches: Exhaustive search, the sequential optimization method described in Section 4, and the independent optimization described in Luo (2005).

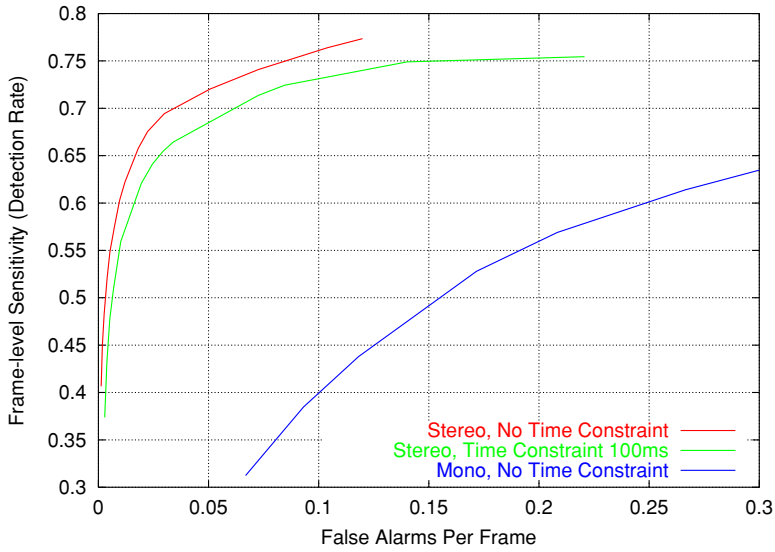

Figure 13. ROC curves of 3 configurations: Stereo/Mono system, optimized without processing time constraint, and stereo system optimized under average processing time constraint of 100 ms.

The system parameter optimization procedure described in Section 4 is employed for each of the three variants using sequence *Run1*. Resulting ROC curves are shown in Fig. 13. The x- and y-axis denote average number of false detections per image, and frame-level sensitivity, respectively. The ROC point at Sensitivity $= 60\%$ has been selected on each of these curves for system evaluation, to ensure a common reference for performance comparison between the 3 variants.

Results on the test sequence *Run2* are given in Table 4 on the frame and trajectory level. Enforcing

Table 4. "Real World" Performance for Sequence *Run2*: "F" denotes frame-level performance, "A"/"B" denote A-class/B-class trajectory performance, respectively. Numbers of false alarms (FA) are given per 10^3 frames for frame-level performance, and per driving minute for trajectory performance.

	Stereo			Stereo, Time constraint			Mono		
	F	A	B	F	A	B	F	A	B
Sensitivity (all)	61.0%	62.0%	78.0%	58.8%	64.0%	78.0%	62.8%	68.0%	82.0%
Precision (all)	52.6%	37.2%	38.0%	46.1%	29.1%	30.3%	22.7%	14.3%	15.2%
FA 10^3 fr, min (all)	23	3.6	3.5	27	5.2	5.1	110	15.3	15.1
Sensitivity (risky)	80.9%	90.0%	100%	75.3%	80.0%	100%	74.2%	80.0%	90.0%
Precision (risky)	83.1%	61.9%	61.9%	73.1%	57.1%	57.1%	30.4%	25.8%	25.8%
FA 10^3 fr, min (risky)	0.69	0.33	0.33	1.0	0.38	0.38	9.9	2.0	2.0
Avg. Proc. (Frame)		162 ms			101 ms			638 ms	

the processing time constraint lowers the precision of the stereo system by about 6–8%, but leads to a significant speed-up of almost 40%. This speed-up is mainly achieved by choosing very strict parameters for the first two modules, stereo-based ROI generation and shape-based detection, so that fewer ROIs have to be processed by the system compared to the unconstrained variant. The same overall sensitivity is then attained by relaxing the parameters of the subsequent modules texture classification and stereo verification. The mono system suffers from a high number of false detections made on cluttered background, which are otherwise cut out by the stereo-based ROI generation.

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 1.5 m (instead of 4 m). Performance significantly increases when only those "risky" pedestrians are considered.

Finally, Fig. 14 provides three screen shots of the PROTECTOR system in action. The top image illustrates a test track scenario, the lower images two urban scenes. The left sub-images show the results of stereo-based ROI generation (the bounding boxes of shape templates activated by stereo are shown in grey, as discussed in Section 3.2.1). Middle sub-images contain detection results superimposed. The right sub-images contain a top view of the scene in front of the vehicle. Shown is the sensor coverage area, with distance scale in meters. Detected pedestrians are denoted by white dots, (relative) velocity vectors by white line segments. The bottom image shows a typical false detection on traffic infrastructure.

6. Discussion

One is naturally inclined to put the obtained system performance in perspective by comparing it to that of previous systems and, secondly, to one that might be expected of a future commercially-viable system. Yet both type of comparisons are difficult, for different reasons. A thorough comparison with previous systems is only possible if all test criteria have been laid out and applied uniformly, on the identical test data set. We have already argued that test criteria were insufficiently spelled out in previous work, and addressed this here by Section 5.1. Furthermore, the use of merely hundreds of images for the test set (see last column of Table 1) can lead to very significant performance variations depending on the "ease" of the selected data set. Performance swings related to the number of false positives up to *orders of magnitude* are not uncommon. In our test set, using many thousands of images, we have observed much smaller performance variations, in the order of a couple of percentages.

But size is not all that matters. Special caution should be taken when, for example, comparing performance reached on data from a surveillance scenario (e.g. Viola et al., 2005) with that of a moving vehicle (this paper). In the former case, the downward tilted camera typically sees the uniform "greyish" background of the road as the backdrop of pedestrians, background diversity is limited by the stationary nature of the camera. This helps improve pedestrian/non-pedestrian discrimination, compared to the case where the camera is vehicle-mounted and non-tilted, where pedestrian are seen against the backdrop of an ever changing complex environment (e.g. vehicles, traffic infrastructure). With all the necessary caveats in place, can now turn to

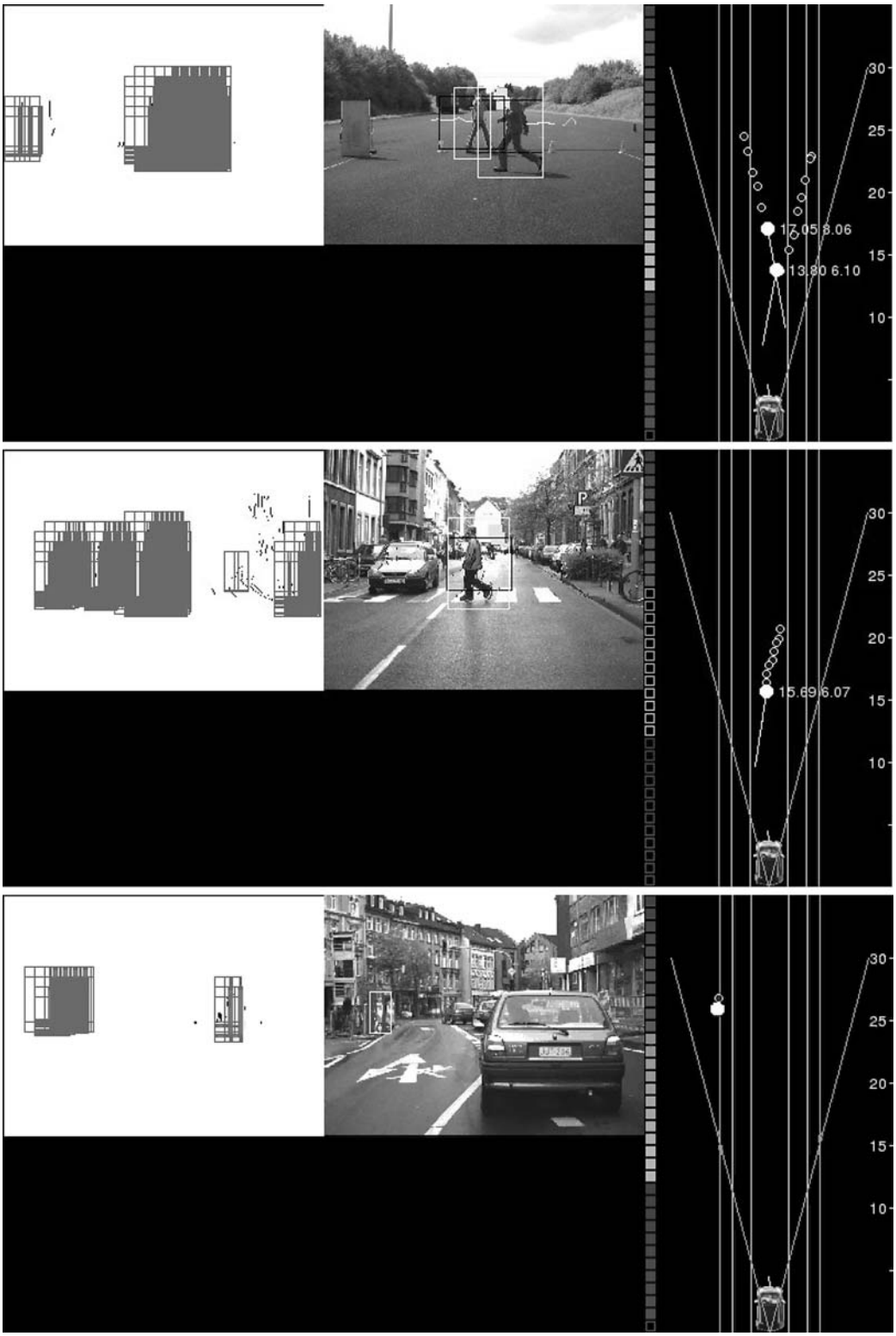

Figure 14. PROTECTOR system results: stereo preprocessing, detections and trajectories.

Table 1 and observe that, in terms of the performance metric typically listed (i.e. correct/false detections per frame), the proposed PROTECTOR stereo-system is very competitive, certainly when considering processing speed constraints. The advantage of using stereo furthermore becomes clear when comparing it to a mono version, it produces an order of magnitude more false positives (see Fig. 13).

A number of methodical improvements could help overcome the above performance gap. Related to the early attention focusing stages, we are interested in detecting independently moving objects using the so-called flow-depth constraint (Heinrich, 2002) and in the more general problem of ego-motion estimation from time-varying stereo data. Computational resources permitting, processing could start with the computation of dense stereo over the entire image, for more accurate object segmentation. We are also interested in more sophisticated tracking by means of a spatio-temporal object representations using distinct linear subspace models or Dynamic Point Distribution Models (DPDMs) (Giebel et al., 2004). State propagation is achieved by a particle filter which combines the three cues shape, texture and depth in its observation density function. Detections are integrated into tracking by means of importance sampling.

7. Conclusions

This paper introduced a multi-cue vision system for real-time pedestrian detection and tracking from a moving vehicle, called PROTECTOR. The detection component involved a cascade of modules, each utilizing complementary visual criteria to focus on relevant image regions. Tight integration ensured that valuable information (constraints) is passed on between successive modules. A novel mixture-of-experts architecture, involving texture-based component classifiers weighted by the outcome of shape matching, was shown to outperform the single texture classifier approach.

In order to cope with the complexity of such large vision system, the paper analyzed the performance of individual modules and their interaction by means of ROCs. A sequential optimization heuristic was shown to result in near-optimal system parameter settings, while incorporating processing time constraints.

Extensive experiments in difficult urban traffic conditions showed that PROTECTOR reaches a correct recognition percentage of 62–100% at the cost of 0.3–5 false classifications per minute. The stereo version of the system clearly outperformed a mono version; performance was furthermore enhanced by a restriction of the sensor coverage area and more processing time. Although overall results are promising, much more research is needed before PROTECTOR-like systems can reach ROC performance adequate for real-world use.

References

Baumberg, A. and Hogg, D. 1994. Learning flexible models from image sequences. In *Proc. of the European Conference on Computer Vision*, pp. 299–308.

Bertozzi, M., Broggi, A., Grisleri, P., Tibaldi, and Rose, M.D. 2004. A tool for vision based pedestrian detection performance evaluation. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Parma, Italy, pp. 784–789.

Borgefors, G. 1986. Distance transformations in digital images. *CVGIP*, 34(3):344–371.

Bouguet, J.-Y. 2000. Camera calibration toolbox for Matlab. In *http://www.vision.caltech.edu/bouguetj/calib_doc/*.

Boyd, S. and Vandenbergh, L. 2004. *Convex Optimization*. Cambridge University Press.

Broggi, A., Fascioli, A., Fedriga, I., Tibaldi, A., and Rose, M.D. 2003. Stereo-based preprocessing for human shape localization in unstructured environments. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Ohio, USA, pp. 410–415.

Broggi, A., Fascioli, A., Carletti, M., Graf, T., and Meinecke, M. 2004. A multi-resolution approach for infrared vision-based pedestrian detection. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Parma, Italy, pp. 7–12.

Cootes, T., Taylor, C., Cooper, D., and Graham. J. 1995. Active shape models—their training and applications. *Computer Vision and Image Understanding*, 61(1):38–59.

Cutler, R. and Davis, L. 2000. Robust real-time periodic motion detection, analysis and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):781–796.

Duda, R., Hart, P., and Stork, D. 2001. *Pattern Classification, 2nd edition*. John Wiley and Sons, New York.

Elzein, H., Lakshmanan, S., and Watta. P. 2003. A motion and shape-based pedestrian detection algorithm. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Ohio, USA, pp. 500–504.

Fang, Y., Yamada, K., Ninomiya, Y., Horn, B., and Masaki, I. 2003. Comparison between infrared-image-based and visible-image-based approaches for pedestrian detection. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Ohio, USA, pp. 505–510.

Franke, U. 2000. Real-time stereo vision for urban traffic scene understanding. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Detroit, USA.

Gavrila, D.M. 2001. Sensor-based pedestrian protection. *IEEE Intelligent Systems*, 16(6):77–81.

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

Gavrila, D.M. and Philomin, V. 1999. Real-time object detection for "smart" vehicles. In *Proc. of the International Conference on Computer Vision*, Kerkyra, Greece, pp. 87–93.

Giebel, J., Gavrila, D.M., and Schnörr, C. 2004. A Bayesian framework for multi-cue 3d object tracking. In *Proc. of the European Conference on Computer Vision*, Prague, Czech Republic.

Grubb, G., Zelinsky, A., Nilsson, L., and Ribbe, M. 2004. Pedestrian detection for driver assistance systems: Single-frame classification and system level performance. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Parma, Italy, pp. 19–24.

Heinrich, S. 2002. Fast obstacle detection using flow/depth constraint. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Versailles, France.

Heisele, B. and Wöhler, C. 1998. Motion-based recognition of pedestrians. In *Proc. of the International Conference on Pattern Recognition*.

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

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

Kuhn, H.W. 1955. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.*, 2:83–97.

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

Liu, X. and Fujimura, K. 2003. Pedestrian detection using stereo night vision. In *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, Bejing, China, pp. 334–339.

Luo, H. 2005. Optimization design of cascaded classifiers. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Conf.* pp. 480–485.

Mohan, A., Papageorgiou, C., and Poggio, T. 2001. Example-based object detection in images by components. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(4):349–361.

Munder, S. and Gavrila, D.M. 2006. An experimental study on pedestrian classification. Accepted for publication in *IEEE Trans. on Pattern Analysis and Machine Intelligence*.

Papageorgiou, C. and Poggio, T. 2000. A trainable system for object detection. *Int. J. of Computer Vision*, 38(1):15–33.

Philomin, V., Duraiswami, R., and Davis, L. 2000. Quasi-random sampling for condensation. In *Proc. of the European Conference on Computer Vision*, Dublin, Ireland, pp. 134–149.

Polana, R. and Nelson, R. 1994. Low level recognition of human motion. In *Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, Austin, pp. 77–82.

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

Sarkar, S. and Chavali, S. 2000. Modeling parameter space behavior of vision systems using Bayesian networks. *Computer Vision and Image Understanding*, 79:185–223.

Shashua, A., Gdalyahu, Y., and Hayun, G. 2004. Pedestrian detection for driver assistance systems: Single-frame classification and system level performance. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Parma, Italy.

Shimizu, H. and Poggio, T. 2004. Direction estimation of pedestrian from multiple still images. In *Proc. of the IEEE Intelligent Vehicle Symposium*, Parma, Italy.

Stenger, B., Thayananthan, A., Torr, P., and Cipolla, R. 2003. Filtering using a tree-based estimator. In *Proc. of the International Conference on Computer Vision*, Nice, France, pp. II:1063–1070.

Sun, J., Rehg, J.M., and Bobick, A. 2004. Automatic cascade training with perturbation bias. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Conf.* 2, pp. 276–283.

Thompson, W. and Pong, T.-C. 1990. Detecting moving objects. *Int. J. of Computer Vision*, 4:39–57.

Toyama, K. and Blake, A. 2001. Probabilistic tracking in a metric space. In *Proc. of the International Conference on Computer Vision*, pp. 50–57.

Viola, P.A., Jones, M.J., and Snow, D. 2005. Detecting pedestrians using patterns of motion and appearance. *Int. J. of Computer Vision*, 63(2):153–161.

Wöhler, C. and Anlauf, J. 1999. An adaptable time-delay neural-network algorithm for image sequence analysis. *IEEE Trans. on Neural Networks*, 10(6):1531–1536.

Zhao, L. and Thorpe, C. 2000. Stereo- and neural network-based pedestrian detection. *IEEE Trans. on Intelligent Transportation Systems*, 1(3).