Human posture recognition for intelligent vehicles

Wachs, Juan P.; Kölsch, Mathias; Goshorn, Deborah

Springer

Wachs, Juan P., Mathias Kölsch, and Deborah Goshorn. "Human posture recognition for intelligent vehicles." Journal of real-time image processing 5.4 (2010): 231-244.
http://hdl.handle.net/10945/63559

J Real-Time Image Proc (2010) 5:231–244
DOI 10.1007/s11554-010-0150-0

SPECIAL ISSUE

Human posture recognition for intelligent vehicles

Juan P. Wachs · Mathias Kölsch · Deborah Goshorn

Received: 17 July 2009 / Accepted: 5 January 2010 / Published online: 12 February 2010

Abstract Pedestrian detection systems are finding their way into many modern "intelligent" vehicles. The body *posture* could reveal further insight about the pedestrian's intent and her awareness of the oncoming car. This article details the algorithms and implementation of a library for real-time body posture recognition. It requires prior person detection and then calculates overall stance, torso orientation in four increments, and head location and orientation, all based on individual frames. A syntactic post-processing module takes temporal information into account and smoothes the results over time while correcting improbable configurations. We show accuracy and timing measurements for the library and its utilization in a training application.

Keywords Articulated body posture recognition · Pose detection · Computer vision · Gesture recognition · Syntactical behavior classifiers · Error correction

1 Introduction

Current research on smart vehicles focuses on driver assistance systems that help reduce the driver's manipulative and cognitive load [1] in order to improve vehicle safety and to enhance the driving experience. Common approaches suggest models to determine the car handling performance considering: (a) real time driver activity recognition, (b) car-internal state estimation, and (c) context recognition. Most systems currently in use measure car-related aspects such as inertia, steering and power input. Much research is oriented on driver-related systems (e.g., [2]). However, modeling and determining the vehicle's context on the street (including pedestrians, road and traffic conditions, weather, sight, etc.) is harder.

Intelligent vehicle systems (IVS) for car-external conditions are gaining importance and have already made their way into cars with automatic parking assistants. A promising future direction for such systems is facilitating human behavior prediction, based on pedestrian detection and body posture. For example, a pedestrian's torso direction, her body's center of gravity, her walking and her gaze direction are all indicators for future behaviors and intent. Hence, the vehicle can warn the driver about dangerous events. Most fatalities involving pedestrians occur outside crosswalks and traffic intersections, when the driver presumably is less alert and an ever-vigilant IVS would be most beneficial (see Fig. 1).

The recent push for unmanned and autonomous systems in the military domain further increases the demand for IVSs. The US Army desires reconnaissance vehicles that can avoid harming pedestrians while assisting soldiers with the detection of potential threats through automated video identification and, for example, alert of behavioral indicators for suicide bombing attempts (suicide bombers have been known to "repeatedly pat themselves to verify that the bomb vest or belt is still attached" [3]).

Vehicle-based pedestrian detection and posture recognition presents several challenges:

J. P. Wachs (✉)
School of Industrial Engineering, Purdue University,
West Lafayette, IN, USA
e-mail: jpwachs@purdue.edu

M. Kölsch · D. Goshorn
MOVES Institute, Naval Postgraduate School,
Monterey, CA, USA

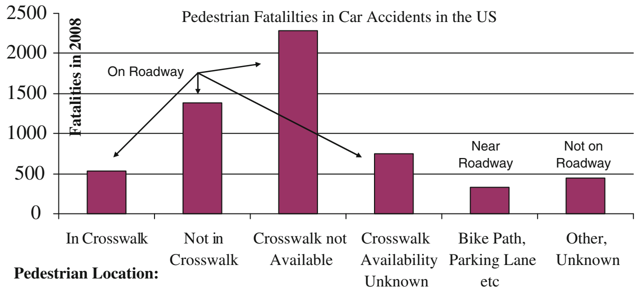


Fig. 1 Crossing roads outside crosswalks kills. Not shown: crossing at non-intersections is 3 times more deadly than crossing at intersections. Source: NHTSA

(a) Every frame is vastly different: background modeling is impractical and temporal coherency is non-trivial.
(b) Pedestrians are often partially occluded by parked cars, trees etc.
(c) The system's latency and speed are critical to its usefulness.

In this paper, we describe a real-time human body detection and posture recognition system that we developed for the US Marine Corps for improving training feedback (see Fig. 2). While our training data and the test cases are specific to training ranges and USMC uniforms, most of the system's characteristics make it directly suitable to an IVS: it performs per-frame image analysis, without considering the background or motion history, it works in real-time, and it can handle partial occlusions. Additionally, temporal post-processing (in recognition space) improves the result's plausibility and accuracy.

Our system was compared with two state of the art efficient and robust pedestrian detectors: Viola-Jones' [26] and Tuzel's [29]. Our method achieved comparable results; moreover the main advantage is that it capitalizes on the commonality between postures/poses. Thus, the number of features needed grows roughly logarithmically with the number of postures, as opposed to the other two, on which the training shows linear growth with increasing number of postures (when used for multi-class detectors). Lower number of features results in faster performance and requires fewer training images to reach the same accuracy as the other two detectors.

2 Related work

Human detection and posture recognition in still-images for a vehicle is described for pedestrian detection in infrared images [4]. Human pose is only used to reject false alarms but the system cannot deliver pedestrian pose. For a comprehensive review of pedestrian detection techniques for advanced driver assistance systems refer to [5].

There are two main approaches for detection and recognition of human postures recognition: three dimensional (3D) with the aid of a skeletal or dynamic model of the human body, and two dimensional (2D). The knowledge of 3D body configuration is a tremendously powerful feature vector for posture recognition, hence, many projects are headed that way. 2D view-based human posture recognition, on the other hand, requires the collection of a large set of independent 2D view images of the scene, which is computationally expensive. Moreover, installation, calibration, object matching, switching, data fusion and occlusion are the main obstacles to practicable multiple camera systems [6]. Alternatively, the work presented in [7], illustrates how 3D human posture can be recovered from still images efficiently and accurately, which goes hand in hand with the requirement of low-cost monocular solutions relying heavily on simple visual features obtained from a single view.

Many approaches have been proposed for human posture recognition. Most of their recognition accuracy is affected by the high variability that exists within the same postures performed by different people.

Three dimensional human body pose is recovered from monocular image sequences in [8] by applying non-linear regression on histogram-of-shape context descriptor vectors. This system was validated with human walking

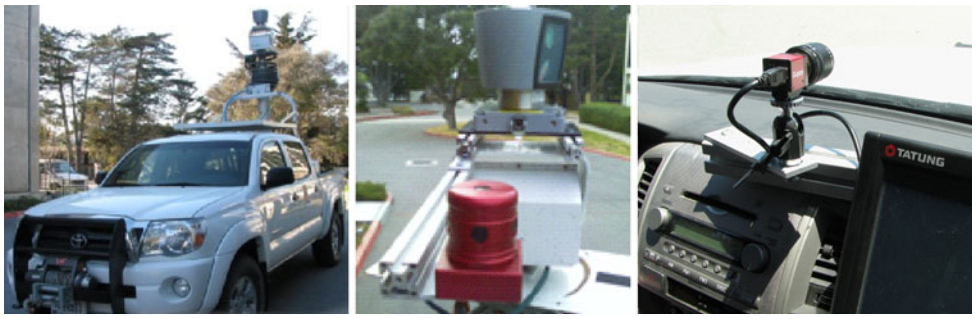


Fig. 2 Our vehicle setup. The Velodyne LiDAR is visible on top and a PtGrey Ladybug panoramic camera in front. The forward-looking camera is mounted on the dashboard next to a touchscreen display for visual feedback

sequences on a low-clutter background. In [9] a system was developed to monitor human behavior for safety purposes. Four main postures in each of three views (frontal, heading left and right) were classified: standing, crouching, sitting and laying. The classification was performed by using a Bayesian framework utilizing features extracted from projection histograms of the silhouette. Projection histograms [10] use an HMM for classification and a multi-camera setup to overcome occlusions. Four main postures were detected: standing, crawling, sitting, and lying. The authors rely strongly on a successful segmentation in the preprocessing stage since the features are extracted from the human silhouette blobs. Juang and Chang's work [11] also requires a successful segmentation for their visual surveillance system to recognize four postures: standing, bending, sitting, and lying. The features are obtained from the DFT coefficients from projected histograms, similar to [9, 10] and the classification is done through a neural fuzzy inference network. In [12], 2D images are collected from different view angles and Fourier descriptors are extracted from the contours. Classification is obtained using aspects-graph representation to recognize eight human postures, including standing, kneeling, sitting and laying down. Image-matching on successive convexification and linear programming [13] can successfully recognizing human postures in cluttered images and videos Yoga poses, skating and baseball postures are recognized using this approach.

In summary, there are two main approaches for recovering human pose from images: model-based approach (or direct approach) and the learning based approach (or indirect approach) [14]. Model-based approaches assume that a parametric body model is known. A cost function representing the pose is optimized through incrementally predicting and updating pose variables. Learning-based approaches do not require a detailed human body model. The model is learned by training examples representing typical human poses and followed by search and comparison, where the new poses are interpolated.

The approach we adopted for our system learns the appearance of body parts and their spatial layout without an explicit body model. The parts are stored in codebooks and later detected in promising regions (i.e. regions found with interest point detectors) rather than searching in all the possible subwindows of an image. The detection is based on casting votes for the object center from the parts matched. This approach was recently adopted for articulated objects detection such as pedestrians and showed very good performance [15–18]. Our work adopts the multi-class approach presented in [19], but applies it to posture recognition. This is an extension of our previous work on monocular posture recognition [20]. This will be integrated with into the BASE-IT system, an intelligent methodology for US Marine training evaluation using behavior analysis [21]. The postures' multiview appearance (e.g., front and back torso) is very similar (compared to the very different views in [19]), making for a challenging classification task.

We discuss the parts-based method in Sect. 3 for a single class and for a multiclass problem. Our dataset and a detailed description of the experiments and results are presented in Sect. 4. Section 5 suggests further directions of improvements and concludes the paper.

3 Parts-based object detection

In this section, we review our approach for body detection and posture recognition using a multi-class detector. First, we describe the feature extraction process from patches, followed by a description of the basic and shared classifiers. The objective is a detector for humans in eight body postures: standing or kneeling, with four view directions each (torso facing the camera, facing left, right, or the person's back towards the camera).

3.1 Dictionary of parts

The dictionary consists of features (patches) extracted from a set of eight images per class, similar to [16]. The images are size-normalized (128 $\times$ 48 pixels for standing postures and to 64 $\times$ 48 pixels for kneeling). Afterwards, the images are then convolved with a delta function, x- and y-derivatives and a Gaussian. For each of the images resulting from the convolutions, 20 patches are selected randomly. The interest point locations fall within the boundary of the annotated silhouette (object of interest). Such patches are extracted in all filtered images and subsequently grayscale-normalized. Each patches' size is selected randomly between 9 $\times$ 9 and 25 $\times$ 25, and its "depth" stems from applying the four filters. Each patch is associated with the location where it was extracted, relative to the center of the object. This location information is represented in a binary location mask centered at the object center. This mask is blurred with a Gaussian function and then separated into two 1D filters (containing the x,y offset distances $\{l_x, l_y\}$) for computational speed reasons, since apply two 1D filters later will be faster than applying a 2D filter to the image. Hence, each entry i in the dictionary has the form $v_i = \{filter, patch, l_x, l_y, image\ number\}$. A total of 640 entries per class were obtained using the procedure which is depicted in Fig. 3, and Algorithm 1 below.

Algorithm 1: Dictionary Creation

```
// Creates a dictionary of parts including all classes
$\mathcal{D} \leftarrow \emptyset$ //initialize the dictionary
for every class do
    for every image I in the dictionary creation set do
        Normalize the image to standard size for the class {128,48} or {64,48}, get $I_N$
        Convolve the image $I_N$ with 4 filters $f_j$ resulting in $I_j$ ,where j=1,..4
        Apply interest point detector and find N interesting points inside the region of interest
        for every point x,y do
            Extract a patch $P_{ij}$ (0<i<N) in every image $I_j$, the patches' sizes varies between [9x9 to 25x25]
            Normalize the brightness on the patch $P_{ij}$
            Find the relative x',y' coordinates relative to the center of the object of interest
            Create a location mask M. Where M(x',y') location is '1' while the rest is '0'
            Decompose the mask into two 1D vectors, lx,ly
            Add the entry $v_k$ to the dictionary $\mathcal{D} \leftarrow \mathcal{D} \cup v_k$ , where $v_k$={$f_j$, $patch_{ij}$, lx, ly}.
        end for
    end for
end for
```

3.2 Computing the training vectors

The training set images contain each of the eight postures. From each image, a few feature vectors are obtained using the following method:

1. Scaling the objects to within a bounding box of 128×48 or 64×48 pixels for standing and kneeling, respectively, and cropping the image to 200×200.
2. The image j is convolved with the filter in dictionary entry i, then convolved with a Gaussian to smooth the response.
3. Cross-correlation with the patch in entry i, yielding a strong response where this patch appears in the filtered image.
4. Application of the 1D filters l_x and l_y to the cross-correlated image, effectively "voting" for the object center. This is summarized in Eq. 1:

$$v_i(x, y) = [(I * f_i) \otimes P_i] * l_x^T l_y \quad (1)$$

where is the convolution operator, $\otimes$ is the normalized cross correlation operator, $v_i(x,y)$ is the feature vector entry i, f is a filter, P is a patch, and l_x and l_y are the x,y location vectors.
5. For each image in step 2, we extract feature vectors $v(x,y)$. A positive sample vector is obtained by retrieving v at location x,y. The negative training samples were a subset of 20 vectors retrieved from x,y locations that had a high response to (1).

Each of these vectors is accompanied with a class label (1–8) and -1 for negative samples. Given 25 images per class, we obtain a training set of 4,000 negative and 200 positive samples, each with 640 features, see Fig. 4 and Algorithm 2.

Algorithm 2: Training Set of Feature Vectors Creation

```
// Computes feature vectors
$\mathcal{F} \leftarrow \emptyset$ //initialize the set of feature vectors
N $\leftarrow$ number of entries in the dictionary $\mathcal{D}$
for every class c do
    for every image $I_{jc}$ with index $_j$ belonging to class c in the training set set do
        Scale up or down the image $I_{jc}$ so the object of interest is {128,48} or {64,48}, (for standing or kneeling)
        Crop the image to [w,h]=200x200
        $\mathcal{V}_{jc} \leftarrow \emptyset$ //initialize the voting matrix
        for every entry $v_k$={$f_k$, $p_k$, lx, ly} in the dictionary $\mathcal{D}$ do
            Convolve image $I_{jc}$ with filter $f_j$ and smooth the response with a Gaussian $\mathfrak{G}$ obtaining image $R_k$
            Cross-Correlate $R_k$ with the patch $p_k$ obtaining $q_k$
            Convolve $q_k$ with each of the 1D filters lx and ly obtaining $M_k$
            $\mathcal{V}_{jc} \leftarrow \mathcal{V}_{jc} \cup M_k$
        end
        Average of the 3D voting matrix $\mathcal{V}_{jc}$ resulting in $V_{jc}$
        Find all the local maxima with coordinates x,y over $V_{jc}$
        Extract the N dimensional feature vector from the voting matrix $\mathcal{V}_{jc}$ on coordinates x,y, obtaining $f_{jc}$
        If the maxima coincide with the true center of the object of interest
            Assign the label $f_{jc}$='c' (positive sample) to the feature vector $f_{jc}$
        Otherwise assign '-1' (negative sample)
        $\mathcal{F} \leftarrow \mathcal{F} \cup f_{jc}$
    end
end
```

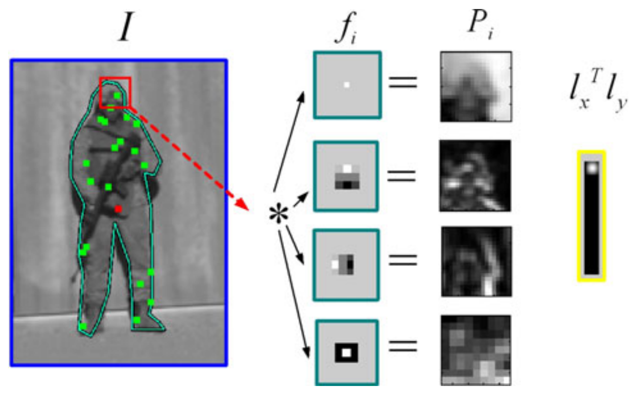


Fig. 3 Dictionary entries: patches selected randomly (on the *left* image) are convolved with a bank of filters. The position is represented with the location matrix (*right*): Since the highlighted patch is directly above the object center (*red dot*), the matrix is a blurred spot at the respective vertical offset

3.3 Multiclass Adaboost with shared features

In this section, we briefly describe the joint boosting algorithm used for multi-class multi-view object detection. For a more detailed discussion, refer to [19].

A boosting algorithm sequentially adds weak learners to form a strong classifier. For the multiclass case, the strong learner is defined as:

$$H(v,c) = \sum_{m=1}^{M} h_m(v,c) \tag{2}$$

where v is the input feature vector, M is the number of boosting iterations, c is a specific class and $H(v,c) = log\, P(z^c = 1|v)/P(z^c = -1|v)$ is the logistic function where z^c is the class c membership label (± 1). When the expectation is replaced by an average over the training data, the cost function can be written as:

$$J_{wse} = \sum_{c=1}^{C} \sum_{i=1}^{N} w_i^c \left(z_i^c - h_m(v_i,c)\right)^2 \tag{3}$$

where N is the number of training vectors, w_i^c is the weight for sample i in class c, and z_i^c is the membership label for sample i in class c. The weak shared learner for multi-class learning, also called the "regression stump," is defined as:

$$h_m(v,c) = \begin{cases} a_S & \text{if} \quad v_i^f > \theta \text{ and } c \in S(n) \\ b_S & \text{if} \quad v_i^f \leq \theta \text{ and } c \in S(n) \\ k_S^c & \quad\quad \text{if } c \notin S(n) \end{cases} \tag{4}$$

where v^f is the *f*th component of v, θ is a threshold, a_S and b_S are regression parameters, and $S(n)$ is a subset of the class labels. Each round of boosting consists of selecting the shared stump and the shared feature f that minimizes (3), from the subset of classes $S(n)$, using the following procedure: Pick a subset of classes $S(n)$. Search all the components f of the feature vector v, for each component, search over all the discrete values of θ and for each pair $\{f, \theta\}$, find the optimal regression parameters a_S and b_S using (5–7), where δ is the indicator function. Finally, select $\{f, \theta, a_S, b_S\}$ that minimizes (3).

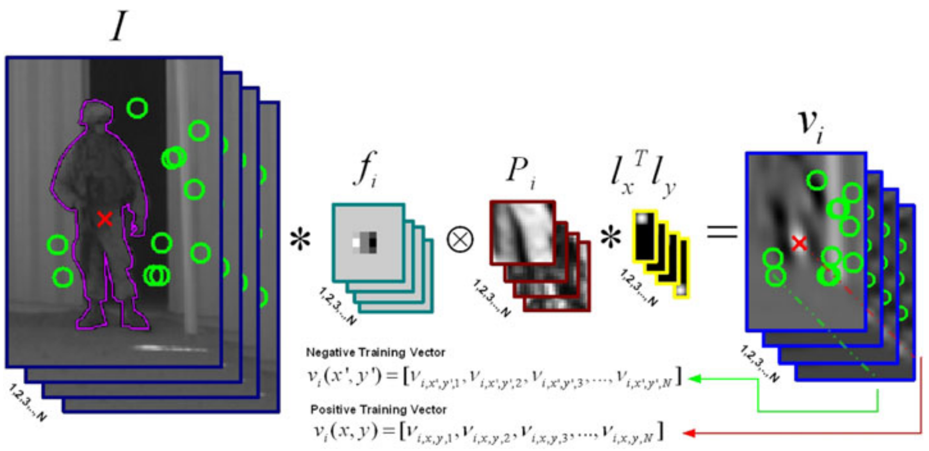
$$a_S(f,\theta) = \frac{\sum_{c\in S(n)} \sum_i w_i^c z_i^c \delta\left(v_i^f > \theta\right)}{\sum_{c\in S(n)} \sum_i w_i^c \delta\left(v_i^f > \theta\right)} \tag{5}$$

$$b_S(f,\theta) = \frac{\sum_{c\in S(n)} \sum_i w_i^c z_i^c \delta\left(v_i^f \leq \theta\right)}{\sum_{c\in S(n)} \sum_i w_i^c \delta\left(v_i^f \leq \theta\right)} \tag{6}$$

$$k^c = \frac{\sum_i w_i^c z_i^c}{\sum_i w_i^c} \tag{7}$$

Therefore, a shared weak learner is associated with these six parameters $\{f, \theta, a_S, b_S, k_c, S_n\}$ of the subset of classes selected. It is more efficient to keep a pointer to the entry in the dictionary from where f was obtained rather than keeping the whole feature vector. This also references to the patch, filter and location vector entries in the dictionary during the detection stage. This new weak learner is added to the accumulated classifier, for each training example:

Fig. 4 Positive and negative vector set creation using the dictionary entries and sampling the center out of the silhouette points


I
f_i
P_i
$l_x^T l_y$
v_i
1,2,3,...,N
Negative Training Vector
$v_i(x', y') = [v_{i,x',y',1}, v_{i,x',y',2}, v_{i,x',y',3}, \ldots, v_{i,x',y',N}]$
Positive Training Vector
$v_i(x, y) = [v_{i,x,y,1}, v_{i,x,y,2}, v_{i,x,y,3}, \ldots, v_{i,x,y,N}]$

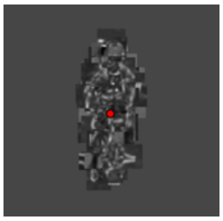


Fig. 5 Dictionary entries selected by the multiclass Adaboost

$H(v_i, c) = H(v_i, c) + h_m(v_i, c)$, where h_m is chosen for the subset of classes that minimizes the classification error. When all the rounds are finished, only the dictionary entries associated with the feature selected are considered for the detection stage. Figure 5 shows all the patches selected from the dictionary after training the multiclass Adaboost for class "torso-0 degrees".

3.4 Detection

To detect an object of class c, a score is calculated for every pixel as the value of the strong classifier $H(v,c)$. If the score exceeds a fixed threshold, the object is detected. $H(v,c)$ is determined with the following procedure.

For every shared weak learner that shares class c, do:

1. Apply the weak learner's four-tuple $\{f, \theta, a_S, b_S\}$ and filter, patch and vectors L_x, L_y (obtained via f from the dictionary), to the test image using (1).
2. Calculate $h_m(v) = a\delta(v_f > \theta) + b$, where V_f is the result from the step 1. Hence, each weak learner casts (continuous-valued) votes for the object of interest.
3. The accumulated voting array (for all h_m) indicates the probability distribution of the object in the image. High values indicate that the weak learners "agreed" on the center of the object.

We obtain one voting array for every strong detector, that is, for every class. As some postures are quite similar, multiple strong detectors often vote for the same or nearby pixel coordinates. We apply a non-maxima suppression algorithm to cluster votes into peaks. The final result is the class of the peak with the highest maximum.

3.5 Torso and head determination

In an analogous fashion, we train a multi-class classifier to detect the head in four orientations. The full-body detection is performed first and determines a limited search region for the head. Experiments yielded the top 1/7th of the body detection area with added margins above the top to be a sufficient head search region when the marines are standing. For kneeling, the top 1/4th of the body detection area should be used, however in the videos used in our experiments, the marines were standing during the whole sequence. If multiple bodies are detected, a heuristic increases the search region, taking nearby body detections into account.

4 Syntactic-temporal analysis and error correction

High level sequential behavior classifiers may be used to enhance the low level computer vision classifiers [22]. In this case, a high level behavior classifier will take as input the class labels that are output from the low level classifiers at each time step. If there is more than one low level classifier that detects objects that are conditionally dependent upon each other, then the high-level classifier may be able to capitalize on any such dependency. This section describes two different syntactic-temporal high-level classifiers used for error correction.

Note that this is different from obtaining temporal features from image sequences, such as frame-to-frame differencing or motion flow. Instead, this considers sequences of still-image detections, avoiding limitations on camera movement.

4.1 Dynamic Bayesian Networks

A Dynamic Bayesian Network (DBN) [23] is a graphical probability model that can express multiple random processes in a single model, making it suitable to enhance the low-level detections of both torso and head orientations. Let Q_1 and Q_2 represent the hidden random process for head orientation and torso orientation, respectively. These hidden processes have observable random processes as outcomes, represented by O_1 and O_2, respectively. DBNs allow a conditional dependency between two random processes (signified by an edge between process nodes), and we arbitrarily chose the torso orientation to dependent on the head orientation. The DBN evolves the system state over time based on transition probabilities between the nodes. This process dependency and temporal progression implement syntactic-sequential error correction and ultimately enhance the posture recognition results (see Fig. 6).

Let Q_1^t represent the hidden head-orientation random variable and let O_1^t represent the observed head-orientation random variable, both at time t. Then the probability $P(O_1^t|Q_1^t)$ of observing O_1^t while the true head orientation is Q_1^t, may be interpreted as modeling the accuracy of the low-level head-orientation classifier. Similarly, letting Q_2^t represent the hidden torso-orientation random variable and letting O_2^t represent the observed torso-orientation random variable at time t then the probability $P(O_2^t|Q_2^t)$ of observing O_2^t while the true torso orientation is Q_2^t, may be interpreted as modeling the accuracy of the low-level torso-orientation classifier. The dependency between hidden and

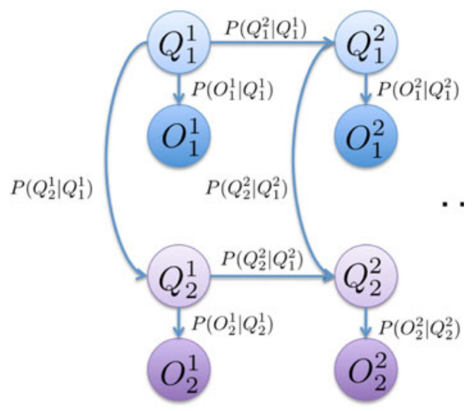


Fig. 6 Two time slices of the DBN for low-level classifier enhancement

observed random variables of one low-level classifier type (e.g. hidden and observed torso orientation) models and "undoes" the errors statistically observed from the low-level classifier. Hence, the observed probabilities correspond to the confusion matrix of each low-level classifier.

The dependency (edge) between the hidden nodes for head-orientation and torso-orientation captures the syntactic context, modeled by the conditional probability of torso-orientation given a particular head-orientation, denoted $P(Q_2^t|Q_1^t)$. The Markovian properties of the head- and torso-orientation's hidden random processes Q_1 and Q_2 are captured in the DBN's temporal transition probabilities $P(Q_1^{t+1}|Q_1^t)$ and $P(Q_2^{t+1}|Q_2^t)$, respectively.

4.2 Hidden Markov Models

Figure 7 illustrates high-level syntactic-temporal error correction with a Hidden Markov Model (HMM) [24]. In its simplest form, an HMM has one hidden and one observed node. Hence, we formulate our random variables differently and introduce a new random variable, $Q_3 = Q_1 \times Q_2$, which is a joint random variable between the head-orientation random variable, Q_1 and the torso-orientation random variable, Q_2.

The observation probability is derived as $P(O_3|Q_3) = P(O_1,O_2|Q_1,Q_2) = P(O_1|Q_1)P(O_2|Q_2)$ due to the independence

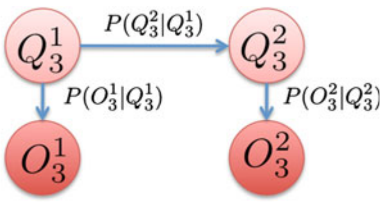


Fig. 7 Two time slices of the Hidden Markov Model for low-level classifier enhancement

assumption between head-orientation and torso-orientation. The transition probability is similarly derived as $P(Q_3^{t+1}|Q_3^t) = P(Q_1^{t+1}, Q_2^{t+1}|Q_1^t, Q_2^t) = P(Q_1^{t+1}|Q_1^t)P(Q_2^{t+1}|Q_2^t)$, where Q_j^t represents the jth random variable at time t.

In summary, all conditional dependencies between low-level classifier results (both temporal within one type of low-level classifier and spatial between two types of low-level classifiers at the same time step) are modeled with one high-level classifier, either a Dynamic Bayesian Network or a Hidden Markov Model. Statistics obtained from training data flow into the models in the following way:

- head-orientation confusion matrix $\rightarrow P(O_1|Q_1)$,
- torso-orientation confusion matrix $\rightarrow P(O_2|Q_2)$,
- observed head orientation changes $\rightarrow P(Q_1^{t+1}|Q_1^t)$,
- observed torso orientation changes $\rightarrow P(Q_2^{t+1}|Q_1^t)$, and
- combinations of head- and torso ors. $\rightarrow P(Q_2^t|Q_1^t)$.

Should the temporal model be applied to a new situation with different characteristics, statistics can be obtained from a bootstrapping run to create a customized high-level classifier.

5 Experiments

We performed two experiments with the multi-class detector. In the first, we applied the detector to still images of Marines (see Fig. 8) with four different orientations and two different postures (standing and kneeling). In the second experiment we used videos of Marines in actual training scenarios and applied a torso and head orientation detectors. The objective is sufficiently accurate posture classification and orientation estimation for subsequent behavior analysis [25]. The posture recognition must occur at interactive frame rates and run on a standard PC or laptop computer. We first analyze the classifier's accuracy in the first two experiments, then discuss the impact of temporal syntactical processing on the video images. Finally, we address speed performance with several metrics and by comparison to another recent pedestrian detector. Additionally, we show how our application fits into the framework of our IVS.

5.1 Multi-class detection and posture recognition in still images

In this experiment we consider two tasks: (a) marine detection and, (b) posture/orientation recognition on still images and we compared our system to Viola-Jones detector [26] and Leibe detector [17].

For testing and training we used an open source online image and annotation repository called LabelMe [27]. Our version currently includes 4,166 annotated images of Marines performing several exercises, from which we selected eight different classes: two postures (standing and kneeling) and four orientations each, based on the torso (frontal, oriented left, right or away from the camera). For the dictionary creation, eight samples per class were used, for training 25 images and for testing another 25 per class. The images were resized to 128 × 48 for standing, and for kneeling to 64 × 48 and then images were cropped to a size of 200 × 200 for the training set and 256 × 256 for the testing set.

Our first task consisted of object detection regardless of class. We normalize each detection score to a probability by dividing each score by the maximum score detected in the testing set. By varying the threshold of the scores (between 0 and 1), we obtain a ROC curve (Fig. 9). False alarms are counted per test area/per image which had an average of 30,000 test areas.

Our results of the ROC curves show that the Viola-Jones "complete" method (non-parts based) performs better than the Adaboost shared detector and Leibe's approach.

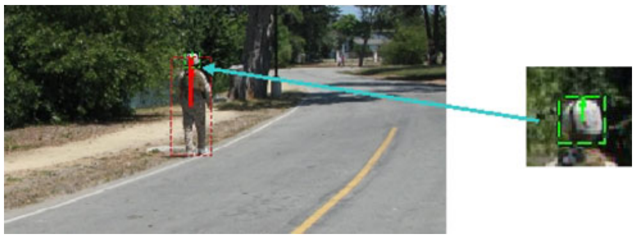


Fig. 8 Detection of uniformed person from vehicle view

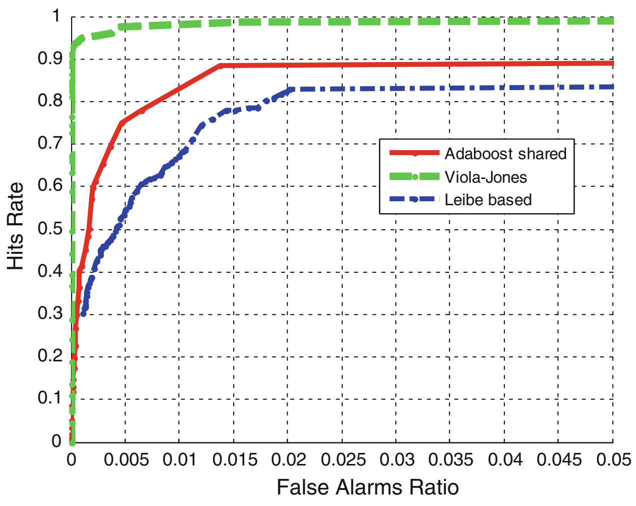


Fig. 9 ROC plot for the marine location detector

The second experiment shows how well the multi-class detector can deal with the two specific postures and four orientations in the images. The performance of posture recognition is summarized in Table 1. We used an operating point on the ROC where FA = 0.015. This parameter was empirically found.

The confusion matrices shows the number of postures classified correctly (diagonal values) and the confused postures (off-diagonal). See Fig. 10 for examples of typical detections and posture recognitions.

These results indicate that the multi-class sharing method achieves higher posture recognition rates than the Leibe and Viola-Jones classifiers. The multi-class sharing method selects dictionary entries based on the amount of common features between classes, while the Leibe-based approach does not take this into account. However, the Leibe-based approach is significantly faster since it requires fewer convolutions.

Table 1 Confusion matrix for eight posture categories: (a) multi-class sharing; (b) Leibe's based, and (c) Viola-Jones

Given class	Assigned class							
	1	2	3	4	5	6	7	8
(a)								
Standing 0	19	4	5	6	0	0	1	0
Standing 90	4	11	8	6	0	1	0	0
Standing 180	9	4	12	4	0	0	0	0
Standing 270	3	5	0	20	0	0	0	0
Kneeling 0	0	0	0	0	16	2	2	2
Kneeling 90	0	0	0	0	2	17	3	2
Kneeling 180	0	0	0	0	0	1	26	0
Kneeling 270	0	0	0	0	0	1	2	20
(b)								
Standing 0	17	10	6	12	9	0	5	1
Standing 90	15	18	3	12	7	1	8	1
Standing 180	11	15	6	11	7	0	5	0
Standing 270	5	12	8	17	2	1	9	1
Kneeling 0	0	0	0	0	16	1	1	5
Kneeling 90	0	0	0	0	6	8	4	7
Kneeling 180	0	0	0	0	5	3	10	5
Kneeling 270	0	0	0	0	8	5	1	14
(c)								
Standing 0	12	2	0	2	7	0	3	0
Standing 90	0	4	2	2	2	1	1	1
Standing 180	3	1	6	0	5	0	3	0
Standing 270	0	3	0	3	0	0	0	1
Kneeling 0	0	1	0	0	19	7	6	1
Kneeling 90	0	2	0	5	4	21	5	1
Kneeling 180	1	0	0	0	8	1	4	6
Kneeling 270	1	0	0	2	5	3	3	6

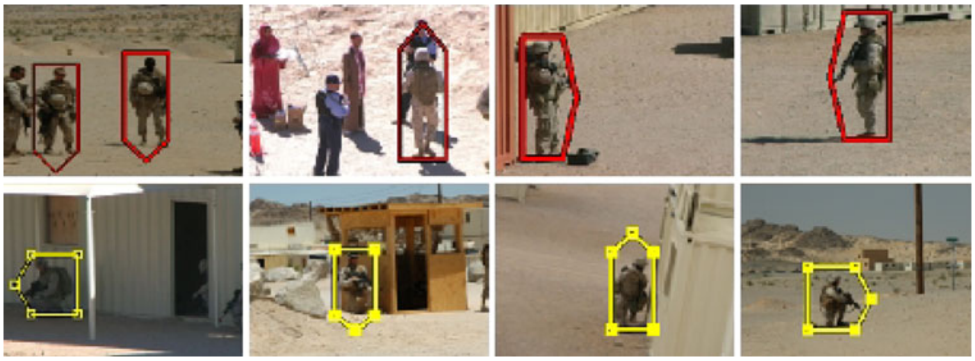


Fig. 10 Example of detections in still images

Fig. 11 Snapshots of movies MOV007, MOV01B, and MOV026

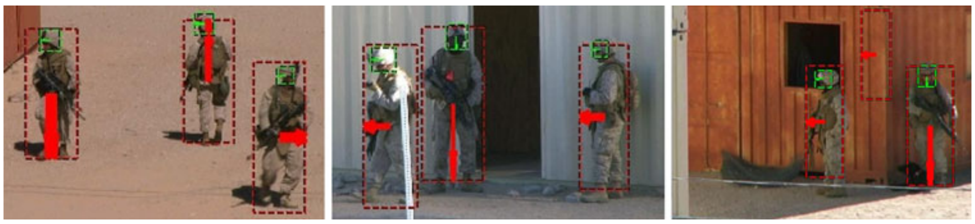


Fig. 12 Snapshots of recognitions in movies MOV007, MOV01B, and MOV026

5.2 Marine posture and head orientation in videos

The training process was similar to the previous section with the difference that we also created a head orientation detector. This means that each detector is capable of classifying four torso and heads orientations each (frontal, oriented left, right or away from the camera). The same number of images was used for dictionary creation and the parameters were identical, too.

For testing, we used three video sequences with 169, 200, and 300 frames and a length of 5–10 s, respectively. The videos can be viewed at http://vision.movesinstitute.org, two snapshots can be seen in Fig. 11. The frames were annotated with ground truth to which our results were compared. For the purposes of this evaluation, we resized and cropped the frames to 200 $\times$ 200 pixels. Two videos had three Marines of interest, MOV026 four (and occasionally other training personnel that was not to be included for detection). None of the video clips showed any kneeling postures.

From the confusion matrices we found that the number of samples (per frame and per marine) classified correctly was 76.43 and 72% for the torso and head, respectively. See Fig. 12 for examples of typical detections and posture recognitions including head position and orientation estimation. If the body was not detected, no head was sought. That explains the relative low number of head detections.

A chart depicting the temporal performance can be found in Fig. 13. For the three movie's respective 169, 200, and 300 frames it shows the ground truth, the raw detections, and the syntactic-temporally post-processed results. The accuracy is calculated as the number of correct recognitions over the total number of detections.

We chose the Bayes Net Toolbox [23] to implement the syntactic-temporal high-level classifier intended to improve the recognition accuracy. The optimal "path" through the Hidden Markov Model is found with an online Viterbi algorithm (as discussed in Sect. 3.2), where the path represents the estimated optimal hidden head-orientation and torso-orientation at each video frame. The Viterbi algorithm corrects the observed torso orientations from the low-level classifiers from an average (per-marine) accuracy of 75.0 to 77.0% and from 71.5 to 75.2% for heads, respectively. This is an improvement of 3.89% overall, 2.65% for the torso, and 5.16% for the head. The performance on the individual movies and Marines is shown in Fig. 14.

5.3 Detector speed performance

We first prototyped the training and testing algorithms in Matlab and profiled the runtime performance. The computationally most expensive operation–calculation of the normalized cross-correlation of dictionary entries with the

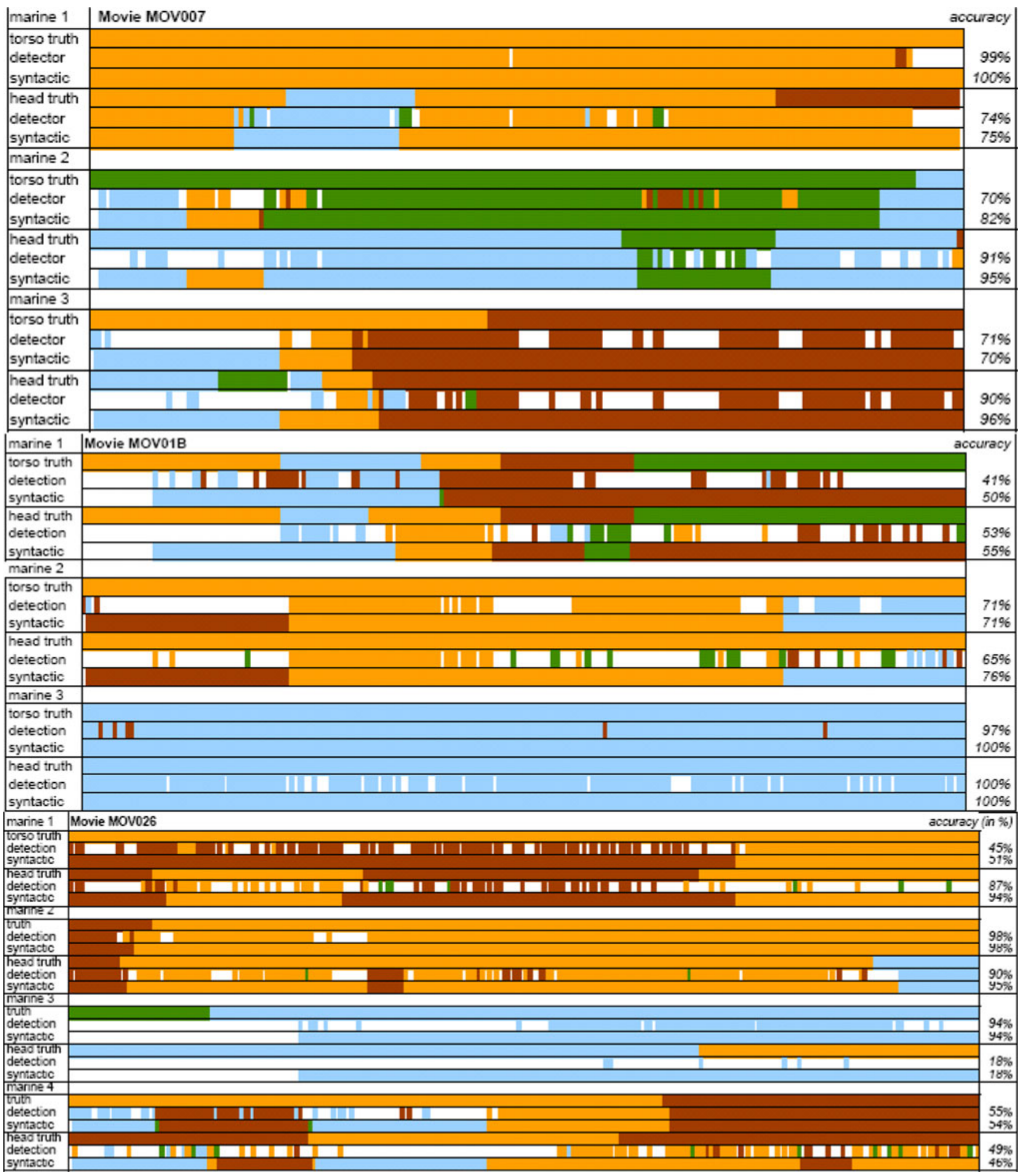


Fig. 13 The temporal progression for the three movies with three, three and four Marines, respectively. The *graph* shows for each marine three *rows* for the torso and three row for the head. Each set of three *rows* shows ground truth, pure detection, and after syntactic-temporal processing. See text for accuracy calculation

image, the convolution of the location 1D filters in the dictionary entries with the image and the 2D convolution with the bank of filters—were then compiled from C code and called from Matlab. This cut the detection time in half. To further speed processing, we implemented the whole system in C/C++. The decision to speed up training in

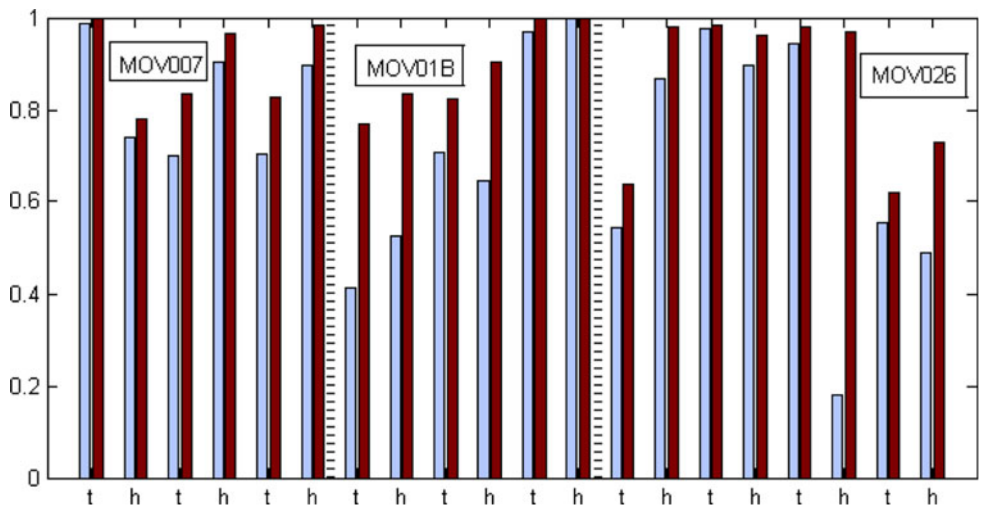


Fig. 14 Accuracy of raw detection (*light blue left bars*) and after syntactic-temporal post-processing (*right, dark bars*), of (t)orso and (h)ead. The first two movies have three Marines, the third movie has four Marines visible

Table 2 Detection time for the different procedures using three implementations

Method operation	Matlab (s)	Matlab and C++ (s)	C++ (s)
Dictionary creation[a]	43	43	13
Feature computation	124.4	71.17	13.3
Training boost	692,000	4,232.95	2,228
Detection	20.15	10.4	1.9

[a] For the dictionary creation there was either a Matlab or C++ implementation

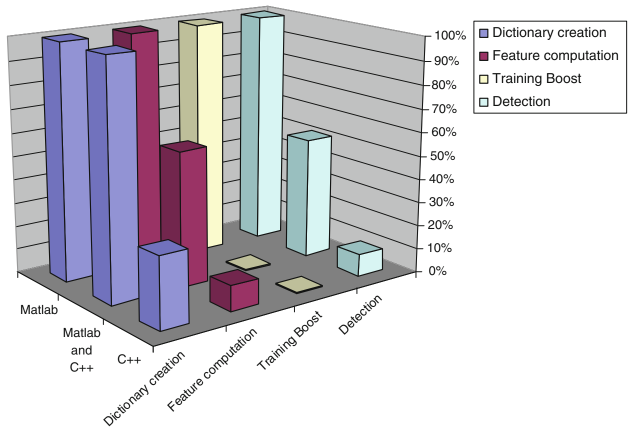


Fig. 15 Computation time of each procedure of the multiclass detector for each of the three implementations

addition to detection was made to (a) permit faster re-training for modified data sets, and (b) to avoid negative effects of inconsistencies that training in Matlab and detection in C/C++ would have caused. For example, the precise implementation of interest point detection matters.

Regarding the Adaboost multiclass classifier, the bottleneck occurs at the computation of the regression parameters a_c and b_c, for unshared classes. The shared classes can obtain the values a_s and b_s using an efficient method described in [19]. This procedure was implemented in C code in order to speed up the classifier training process.

The first experiment consisted on the evaluation of the runtime performance of the main functions in our algorithm in the three mentioned implementations: (a) dictionary creation, (b) feature computation, (c) training the multiclass Adaboost, and (d) the detection stage. The results were all obtained on an Intel Quad core with 2,92 GB of memory with a 2.4 GHz CPU running MS Windows XP, see Table 2 and Fig. 15.

We can see from the graph in Fig. 15 that the most significant improvement occurs when training the multiclass classifier; however there is no significant difference between the Matlab & C++ compared to C++ alone. The detection procedure is the fastest using C++ alone, while the pure Matlab implementation is significantly slower.

To assess the time performance as a function of the area scanned, we performed detections on increasing areas of an image (with increments of 30 pixels in both width and height) and recorded the detection time. Results are illustrated in Fig. 16, illustrating that the pure C++ implementation scales almost linearly as opposed to the other two implementations where the detection time increases in a quadratic fashion.

To assess the speed and accuracy performance of the three different methods and in comparison to a recent successful detector suggested by Tuzel et al. [28], we measured the detection speed (in any posture) while varying the detection quality. We gradually reduced the number of features (strong classifiers, filters etc.) and evaluated the accuracy (true positives divided by all detections) on the same set of test images for all detectors. The results are shown in Fig. 17.

Figure 17 suggests that the Viola-Jones' method is fast but does not scale well with increasing number of features, meaning that longer computation times (or faster

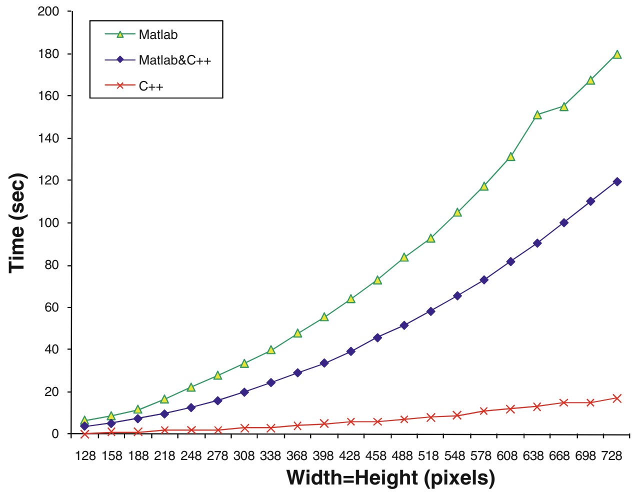


Fig. 16 Detection time versus scanned area

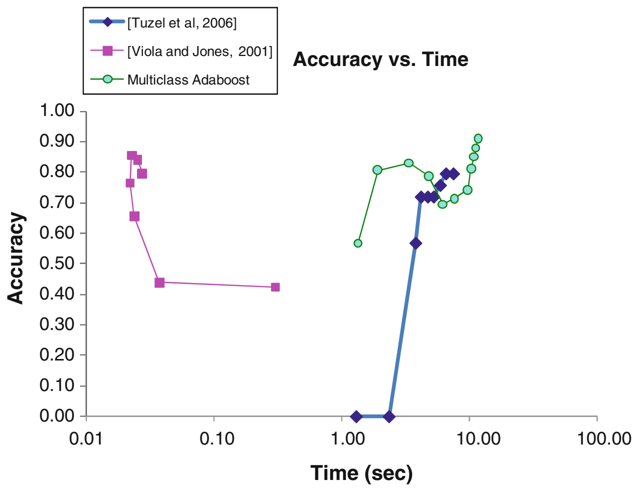


Fig. 17 The tradeoff between accuracy and speed. Non-monotonic curves are caused by expensive post-processing and due to the particular test image set

processor) will not improve the overall performance. Tuzel's method produces reasonable accuracy only for higher processing times. Note that the increasing number of features does not have a major impact on the algorithm's performance. It does, however, result in higher computational complexity.

For the Tuzel et al. [28] approach we used their nine-dimensional feature matrix suggested in their paper: pixel locations—x,y; color (RGB) values and the norm of the first and second order derivatives of the intensities with respect to x and y. To plot the graph, we gradually removed one feature (from the nine) and sampled the accuracy. We found that the time to compute the covariance for different scales is very low (only the time taken to access the integral covariances), where the main delay was the distance computation in Riemannian manifolds. With increasing number of features this delay becomes significant.

6 Discussion and future work

We implemented a joint boosting algorithm to address the problem of human posture recognition for behavior analysis purposes in the scope of intelligent vehicles. The weak detectors were trained jointly so the number of shared features was maximized. We found that the multiclass detector was able to successfully detect Marines when they were not cluttered, with up to 98.73% detection, where in situations where the Marines were occluded, the classifier succeeded in classifying only up to 53% of the torso orientations correctly. The results between posture detection and recognition classifiers show that the Viola-Jones method performed better on the detection task only, while the multiclass Adaboost outperformed both Viola-Jones and Leibe's method for posture detection and recognition. We suspect that this is due to the better ability of the parts-based multiclass Adaboost approach to capitalize on the commonality between postures/poses without loosing their characteristics. In order to overcome the obstacle of recognition on highly cluttered images with occluded Marines (often common with local object detector methods) syntactic-temporal analysis was adopted for correcting low-level classification errors. HMMs are able to correct short-term incorrect recognitions. Additionally, HMMs can recover from situations where there are no head and/or torso observations from the low-level classifiers (i.e. low-level classifier *misses*), exploiting the Markovian property. In other words, the current observation, whether it be the null observation or not, is dependent only on the previous observation. In the case of the null observation, the Hidden Markov Model therefore recovers using the predicted hidden state of the previous state to predict the hidden state of the current state. This is true for both the head-orientation random variable and the torso-orientation random variable. However, the current HMM implementation cannot correct sequences made of a consistent, long duration of incorrect recognitions.

In addition to comparing various posture detection and recognition classifiers, followed by high-level syntactical correction of low-level posture classifiers, we compared software implementations for eventual deployment in real time systems: pure Matlab, pure C++, and a hybrid implementation. Performance tests showed the pure C++ implementation to be up to ten times faster, particularly for the detection task (1.9 vs. 20.15 s). Surprisingly, the C++ implementations scaled sub-logarithmically with increasing image size.

The shared-features of the multiclass Adaboost low-level classifier provide a fast first stage for posture recognition. The HMM does not cause noticeable delays in the recognition system since it operates on a much reduced feature space (4×4 discrete states) and only at regions where Marines have been detected.

A natural extension of this work will be experimenting with features that can be computed faster, such as HoG (Histogram of Gradients). This will remove the need for the computationally expensive cross-correlation to obtain features, and the HoG features are robust enough to cope with affine transformations. Thus, this will result in shorter times and possibly a more robust detector. In addition, we have begun to optimize the dictionary creation step that precedes the computationally expensive Adaboost selection. Hence, it is imperative to start with a dictionary whose entries were selected in some optimal fashion, and permitting use of a one step cost function.

Regarding hardware implementation, we are currently looking into several other hardware options to further improve the speed performance. Most computationally expensive operations are 2D image filtering (convolution and cross-correlation), which are easily parallelized on SIMD (single instruction, multiple data) architectures such as graphics processors (GPU) or even on programmable logic devices (Field Programmable Gate Array, FPGA). The further advantages of FPGAs make them well suited to applications in vehicles: low power consumption, low heat dissipation, and no opportunity for software failures.

7 Conclusions

DARPA's Urban Challenge wisely excluded the recognition of pedestrians and pedestrian avoidance from the competition: IVS must still improve upon these capabilities before autonomous vehicles can be trusted in real-life situations. This paper presented a system for combined detection and posture recognition of US Marine Corps (USMC) members, which can be used for enhancing any IVS system. After algorithm evaluation and prototyping in Matlab, we implemented a real-time library and evaluated its performance on various test videos. Syntactic-temporal post-processing was found to improve the accuracy to just over 76%, which includes both body detection, and torso and head orientation estimations.

The presented system has applications for automatic USMC training performance evaluation as well as for people detection and collision avoidance for intelligent vehicle systems.

Acknowledgments This research was supported by a grant from the Office of Naval Research and it was performed while the first author held a National Research Council Research Associateship Award at the Naval Postgraduate School.

References

1. Riener, A., Ferscha, A., Matscheko, M.: Intelligent vehicle handling: steering and body postures while cornering. Lecture Notes in Computer Science. 21st International Conference on Architecture of Computing Systems (ARCS 2008), System Architecture and Adaptivity, vol. 4934, pp. 68–82 (2008)
2. Kisacanin, B.: Method of detecting vehicle-operator stat—US Patent App. 11/317,431 (2005)
3. Balser, B.B., Foxman, A.H.: Armed Assaults and Suicide Bombers. Anti-Defamation League 73 (2005)
4. Bertozzi, M., Broggi, A., Fascioli, A., Graf, T., Meinecke, M.M.: Pedestrian detection for driver assistance using multiresolution infrared vision. IEEE Trans. Vehicular Technol. **53**, 0018–9545 (2004)
5. Gerónimo, D., López, A.M., Sappa, A.D., Graf, T.: Survey on pedestrian detection for advanced driver assistance systems. IEEE Trans. Pattern Anal. Mach. Intell. (2009)
6. Hu, A.W., Tan, T., Wang, L., Maybank, S.: A survey on visual surveillance of object motion and behaviours. IEEE Trans. Syst. Man Cybern. C **34**(3), 334–352 (2004)
7. Moeslund, T.B., Granum, E.: A survey of computer vision-based human motion capture. Comput. Vis. Image Underst. **81**, 231–268 (2001)
8. Agarwal, A., Triggs, B.: Recovering 3D Human Pose from Monocular Images. IEEE Trans. Pattern Anal. Mach. Intell. **28**(1), 44–58 (2006)
9. Cucchiara, R., Grana, C., Prati, A, Vezzani, R.: Probabilistic posture classification for human-behavior analysis. SMC-A(35), No. 1, pp. 42–54 (2005)
10. Cucchiara, R. Prati, A. Vezzani, R.: Posture classification in a multi-camera indoor environment. IEEE Intl Conf on Image Proc, ICIP 2005, I-725-8 (2005)
11. Juang, C.F., Chang, C.M.: Human body posture classification by a neural fuzzy network and home care system application. IEEE SMC-A (37), No. 6, pp. 984–994 (2007)
12. Hu, J.S. Su, T.M., Li, P.C.: 3-D Human posture recognition system using 2-D shape features. In: 2007 IEEE International Conference on Robotics and Automation, pp. 3933–3938 (2007)
13. Jiang, H., Li, Z.N., Drew, M.: Recognizing posture in pictures with successive convexification and linear programming. IEEE Multimedia (2007)
14. Bouchard, G. Triggs, B.: A hierarchical part-based model for visual object categorization. In CVPR (2005)
15. Fergus, R., Perona, P., Zisserman, A.: A sparse object category model for efficient learning and exhaustive recognition. In CVPR (2005)
16. Murphy, K., Torralba, A., Eaton, D., Freeman, W.T.: Object detection and localization using local and global features. Lecture Notes in Computer Science. Sicily Workshop on Object Recognition (2005)
17. Leibe, B., Seemann, Schiele B.: Pedestrian detection in crowded scenes. CVPR (1) 2005, (2005):878–885
18. Felzenszwalb, P. McAllester, D., Ramanan D.: A discriminatively trained, multiscale, deformable part model. Proceedings of the IEEE CVPR 2008, (2008)
19. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing visual features for multiclass and multiview object detection. IEEE PAMI, 29:5, pp. 854–869 (2007)
20. Wachs, J.P. Goshorn, D, Kolsch, M.: Recognizing human postures and poses in monocular still images. In: Proceedings. of the

IPCV 2009. Intl. Conference on Image Processing, Computer Vision & Pattern Recognition, vol 2, pp. 665–671 (2009)
21. BASE-IT: Behavior Analysis and Synthesis for Intelligent Training. On-line: http://www.movesinstitute.org/base-it/index.html
22. Goshorn, R., Goshorn D., Kölsch, M.:The enhancement of low = level classification for ambient assisted living, in Behavioral Monitoring and Interpretation Workshop at the German Conference on Artificial Intelligence, September (2008)
23. Murphy, K.: Dynamic Bayesian Networks: representation, inference, and learning. PhD Dissertation, Computer Science. University of California, Berkeley (2002)
24. Rabiner, L. Juang, B.: An introduction to hidden markov models. In: IEEE ASSP Mag. vol. 3, pp. 4–16, IEEE Signal Processing Society (1986)
25. Sadagic, A. Welch, G., Basu, C., Darken, C., Kumar R., Fuchs, H., Cheng, H., Frahm, J.M., Kölsch, M., Rowe, N., Towles, H., Wachs, J., Lastra, A.: New Generation of Instrumented Ranges: Enabling Automated Performance Analysis, I/ITSEC (2009)
26. Viola, P., Jones, M.: Robust real-time face detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
27. Russell, B., Torralba, A., Murphy, K., Freeman, W. T.: LabelMe: a database and web-based tool for image annotation. Int. J. Comput. Vis. **77**(1–3) (2008)
28. Tuzel O., Porikli F., Meer P.: Region covariance: a fast descriptor for detection and classification. In: Proceedings of 9th European Conference on Computer Vision, Graz, Austria, vol. 2, pp. 589–600 (2006)
29. Tuzel, O., Porikli, F., Meer, P.: Pedestrian detection via classification on Riemannian manifolds. IEEE Trans. Pattern Anal. Machine Intell. **30**, 1713–1727 (2008)

Author Biographies

Juan Wachs is an assistant professor at Purdue University in the IE department. He completed a postdoctoral training at the Naval Postgraduate School's MOVES Institute. His research interests are machine and computer vision, robotics, tele-operations, human factors, assistive technologies and health support systems. His work on the Gestix project led to the first in vivo implementation of a hand gesture system for the manipulation of medical images during surgery in 2006. From 2000 to 2007, he was part of a team of researchers working in the Virtual Reality and Telerobotics Lab at Ben-Gurion University of the Negev. Juan Wachs is a member of IEEE and the Operation Research Society of Israel (ORSIS). He has published in journals including IEEE Trans. Systems, Man, and Cybernetics, Journal of American Medical Informatics, Journal of Image and Graphics, and the International Journal of Semantic Computing. He has taught digital electronics, multimedia and web development at ORT and Intel Colleges, Israel. He received his B.Ed.Tech in Electrical Education from the ORT Academic College in Jerusalem, his M.Sc and Ph.D in Industrial Engineering and Management, Information Systems and Intelligent Systems tracks, respectively, from the Ben-Gurion University of the Negev.

Mathias Kölsch (Ph.D., University of California, Santa Barbara, 2004) is an Assistant Professor of Computer Science at the Naval Postgraduate School in Monterey, CA. He is also affiliated with the MOVES Institute, Chair of the MOVES Academic Committee and the Academic Associate for the MOVES curriculum. His research interests include computer vision, hand gesture recognition, augmented and virtual environments, sensor networks and mobile/embedded computing.

Deborah Goshorn has been an engineer and scientist for the US Navy for the past 5 years. Deborah spent 4 years at SPAWAR Systems Center Pacific as an engineer and computer scientist for various projects including DSP rapid prototyping for mitigating interference in various SATCOMs and statistical analysis for serially correlated time series data and artificial intelligence systems. Deborah spent the last year as a Faculty Research Associate in the Modeling Virtual Environments Simulation (MOVES) Institute researching computer vision techniques for US Military systems that require posture recognitions and sequential, syntactical models for posture recognition error recovery. Deborah is an instructor for courses in Systems Engineering at Naval Postgraduate School. Deborah is also Adjunct Faculty at Monterey Peninsula College in Statistics. Deborah, as a SMART scholar, is completing her PhD in Computer Science from University California, San Diego this summer with her dissertation titled, *The Systems Engineering of a Net-Centric Distributed Intelligent System of Systems for Human Behavior Classification.* Deborah has her MS in Statistics, MEng in Electrical and Computer Engineering, BS in Computer Engineering, and BA in Applied Mathematics, all from UCSD.