Learning a Family of Detectors via Multiplicative Kernels

Quan Yuan, *Member*, *IEEE*, Ashwin Thangali, *Student Member*, *IEEE*, Vitaly Ablavsky, *Student Member*, *IEEE*, and Stan Sclaroff, *Senior Member*, *IEEE*

Abstract—Object detection is challenging when the object class exhibits large within-class variations. In this work, we show that foreground-background classification (detection) and within-class classification of the foreground class (pose estimation) can be jointly learned in a multiplicative form of two kernel functions. Model training is accomplished via standard SVM learning. When the foreground object masks are provided in training, the detectors can also produce object segmentations. A tracking-by-detection framework to recover foreground state in video sequences is also proposed with our model. The advantages of our method are demonstrated on tasks of object detection, view angle estimation, and tracking. Our approach compares favorably to existing methods on hand and vehicle detection tasks. Quantitative tracking results are given on sequences of moving vehicles and human faces.

Index Terms—Object recognition, object detection, object tracking, pose estimation, kernel methods.

1 Introduction

A computer vision system for object recognition typically has two modules: a detection module [11], [58] and a foreground state estimation module [1], [3], [49]. The detection module is often implemented as a scanning window process where each window location in an image is evaluated by a binary classifier, i.e., foreground class versus background class. The invocation of the foreground state estimation module is conditioned on the detection of an instance of the foreground class; in other words, the second module is tuned to the variations within the foreground class. This second module can be implemented in numerous ways. For discrete state spaces—for example, face ID, hand shape class, or vehicle type—estimation can be framed as a multiclass classification problem [43]: Given an input feature vector, produce an estimate of the class label. For continuous state spaces—for example, face age, hand joint angles, vehicle orientation—estimation can be formulated in terms of regression [1]: Map a given input feature vector to its most likely location in the foreground state space. Another common approach for foreground state estimation is to use nearest neighbor methods [3], [49].

In any case, when object classes exhibit large within-class variations, detection and foreground state estimation can be chicken-egg problems. Assuming the objects are detected and segmented from the background, foreground state estimation is relatively straightforward. Assuming specific variations of the foreground class, detection can be achieved as in [42]. However, if neither the foreground state nor detection is given, then challenges arise. For example, it is difficult for a single detector to cope with all variations of the foreground class while at the same time providing reliable discrimination between the foreground and background—especially in applications where there are widely varying, or even unconstrained, backgrounds.

A common strategy in this setting is divide-and-conquer [28], [37], [57], [60]: Divide the foreground class into subclasses by partitioning the space of within-class variations, and then train a separate detector for each partition. Thus, a set of detectors is trained, where each detector discriminates between the background class and its subset of the foreground class. This strategy has been employed in hand detection [37], multipose vehicle detection [60], and multipose face detection [28], [57]. An additional advantage of such a strategy is that coarse estimation of the object pose can also be obtained during the detection process. For example, in multipose face detection, the detector of the correct face pose tends to have a high response. However, in a divide-and-conquer strategy, the partitioning of a foreground class is oftentimes arbitrary. Moreover, to keep ample training examples in each subclass, the partitioning of the foreground class is usually coarse, which limits the ability of pose estimation.

We propose a different strategy that avoids explicit partitioning of the foreground class in this paper: learning a family of detectors, where the detectors themselves are parameterized over the space of within-class variations. Our formulation utilizes a product of two kernel functions: a within-class kernel k_θ to handle foreground state variations and feature sharing, and a between-class kernel k_x to handle foreground-background classification. This kernel formulation is used in a Support Vector Machine (SVM) [9] training algorithm that outputs support vectors and their weights, which can be used to construct a family of detectors that are tuned to foreground variations. After SVM training, a sample set of detectors can be generated, where each detector is associated with a particular foreground state parameter value. All of the samples from the detector family share the

- *Q. Yuan is with the US Research Center, Sony Electronics, Inc., 1730 N 1 St., San Jose, CA 95112. E-mail: quan.yuan@am.sony.com.*
- *A. Thangali, V. Ablavsky, and S. Sclaroff are with the Computer Science Department, Boston University, 111 Cummington St, Room 138, Boston, MA 02215. E-mail: {tvashwin, ablavsky, sclaroff}@cs.bu.edu.*

Manuscript received 21 Jan. 2009; revised 14 Oct. 2009; accepted 28 Mar. 2010; published online 7 June 2010.
Recommended for acceptance by G. Hager.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2009-01-0049.
Digital Object Identifier no. 10.1109/TPAMI.2010.117.

0162-8828/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

same support vectors, but the weights of these support vectors vary depending on the within-class state value. A useful side effect of this support vector sharing is that features are implicitly shared across the whole detector family.

The proposed formulation is useful in solving detection, state estimation, and tracking problems. For detection using a scanning window process, an image window can be classified as foreground if at least one of the detectors in the family produces a score that is above a predefined threshold. For a given image window, the foreground state can be estimated simply by examining the state values associated with detectors that produce the highest responses for that input. For particle filter-based tracking methods, like CONDENSATION [22], importance sampling from the detector family can be driven by a dynamical model at each frame, where the objects are allowed to undergo a range of state variations over time.

With proper nonparametric kernel functions, our formulation can be extended to nonparametric cases when explicit parameter annotation of the foreground class examples is too expensive to obtain. A mode-finding method is proposed that selects a representative subset of samples from the detector family in the nonparametric case. This generally reduces the number of detectors to be invoked, and thereby makes detection more efficient. If state estimation or tracking is desired, then the user can label the state for each sample in the representative subset. This alleviates the burden of assigning ground truth states for the complete training set, and instead focuses only on labeling the smaller representative subset.

The proposed framework is evaluated in three application areas. The first involves hand detection, segmentation, and shape estimation for images taken from videos of Flemish and American Sign Language. There is a wide variation of hand shapes and orientations in these videos. The framework is also tested in estimating index finger angles. The second application involves detection, orientation estimation, and tracking of vehicles driving on highways, and the more challenging case of race cars careening on dirt roads. The third application focuses on the problem of detecting and tracking multiple human faces while simultaneously estimating the left-right rotation angles under illumination variations.

2 Related Work

A large amount of work in computer vision has been proposed to handle the issue of detecting an object class that exhibits large appearance variations. For instance, generative models were proposed in [32], [40] to learn a set of low-dimensional representations (eigenspaces) that cover a broad range of appearance variations. Given a novel input image, the position of its projection on these eigenspaces determines its state.

Recent work in multiview face detection builds detectors for different head pose ranges. In [27], [57], subclasses according to the face orientation are created and corresponding detectors are learned for each subclass. In [19], [28], coarse to fine hierarchies (trees) of face orientation are created. Each nonleaf node in the hierarchy is a face subclass and is further partitioned into subclasses. At each node, a detector is learned for this subclass. During detection, an input is examined by the detectors of a sequence of nodes, starting from the root. If the input is classified by the detector of the current node as from the corresponding face subclass, it is passed to its children (one or multiple) to be examined; otherwise, it is rejected as from the background class. The annotation of the leaf node at the end of the sequence determines the face orientation of the input, if it passes all detectors along the sequence.

Similar approaches that partition the foreground class according to foreground state annotation [54] or via unsupervised clustering [16], [26], [37], [48], [60] are employed in pedestrian and human hand detection. However, the trained subclass detectors have limited power when there are too few training samples in each subclass. A large foreground training set is required to handle a large number of foreground subclasses.

To make the best use of limited training data, feature sharing is important for multiclass detection. Explicit feature sharing approaches proposed in [55], [62] improve detection accuracy, but they also tend to make training more expensive due to the combinatorial complexity in choosing classes or training examples to share features. In both works, greedy strategies to select sharing classes or training examples of each feature are employed as a trade-off for training speed.

Given the above-mentioned issues, hybrid methods that unify detection and foreground state estimation are of great interest. Some approaches combine bottom-up part-based detectors with top-down geometric constraints [15], [20], [39], [52]. These methods are applied to handle large appearance variations of articulated objects like the human body. However, the number of possible configurations exhibits combinatorial complexity in terms of the number of detected parts. The search for satisfactory part configurations can be expensive in practice. A recent work [14] applies the idea of part-based model in a latent-SVM formulation. The detection accuracy is improved with this discriminatively trained model on various object detection tasks. However, in this model, each part may not necessarily correspond to a meaningful object part. Body pose or foreground state information is not directly available from the detection result.

Some other approaches employ a recognition-verification strategy (e.g., [44]), where a one-to-many mapping is used to produce estimates of body pose (bottom-up), and then recognition models are used to verify pose estimates (top-down). Nevertheless, bottom-up recognition from images with background clutter remains difficult, and the verification step cannot correct an error when the recognition is already wrong.

Other approaches use probabilistic methods to learn a generative model to predict a human pose and then verify it [5], [13], [53] using a recognition model. In [5], a generative model of the background class is also obtained. However, for recognition, generative models may not be as robust as discriminative models. One of the major reasons is that generative models often make conditional independence assumptions that are not well justified (not supported by the data).

Some other works [12], [31], [56] propose kernel combinations or kernel parameter optimization to improve the performance of kernel methods. In these works, kernel combinations are used for a single classification or regression task, but not both, whereas, in our approach, both the

foreground-background classification and foreground state estimation problems are jointly solved.

A recent approach [21] jointly solves human body localization and pose estimation by a structural SVM using a product kernel. In this work, both localization and pose estimation are defined as continuous parameter estimation problems that can be solved via nonlinear optimization. However, the optimization process still needs help from a traditional localization method and a greedy scheme due to the nonconvexity of the model.

3 Problem Definition and Our Approach

Given a feature vector $\mathbf{x} \in \mathbb{R}^n$ computed for an image patch,[1] our goal is to decide whether or not the image patch depicts an instance of the object with parameter $\boldsymbol{\theta} \in \mathbb{R}^m$, which parameterizes certain variations of the foreground object class, e.g., object pose, view angle, or latent factors that can be obtained via unsupervised learning. Basically, we want to have following outputs given an input $\mathbf{x}$: 1) *If* $\mathbf{x}$ *is a foreground object, the output is an estimate of the foreground parameter*, and 2) *if* $\mathbf{x}$ *is a background patch, the output is simply a label "background."*

In a traditional multipose object detection approach, e.g., multipose face detection [28], the foreground class is first partitioned in to subclasses. Each subclass is associated with a rotation angle, e.g., $\theta \in \{0°, 20°, 40°, \ldots, 180°\}$. Then, a set of detectors $C_0(x), C_{20}(x), C_{40}(x), \ldots, C_{180}(x)$ is trained with corresponding training examples. The detection process is organized into a hierarchy for efficiency. Because the detector of the correct rotation angle tends to have the highest response, a rotation angle estimate can be obtained by comparing detection scores from all detectors.

In our work, we aim to learn a unified function $C(\mathbf{x}, \boldsymbol{\theta})$ that tells whether $\mathbf{x}$ is an instance of the object with parameters $\boldsymbol{\theta}$,

$$C(\mathbf{x}, \boldsymbol{\theta}) \begin{cases} > 0, & \mathbf{x} \text{ is an instance of the object with } \boldsymbol{\theta}, \\ \leq 0, & otherwise. \end{cases} \quad (1)$$

The function $C(\mathbf{x}, \boldsymbol{\theta})$ can be viewed as a global function to "generate" individual detectors $C_0(x), C_{20}(x), C_{40}(x), \ldots, C_{180}(x)$ by plugging in a $\boldsymbol{\theta}$ value into $C(\mathbf{x}, \boldsymbol{\theta})$. These individual detectors can be regarded as points in a space of detectors. The main idea of our approach is to learn the detector space as a function of $\boldsymbol{\theta}$, rather than learning individual detectors separately. During detection, we use a discretization of the continuous model to generate discrete samples in $\boldsymbol{\theta}$ space and apply corresponding detectors on an input. Rather than considering a sample set of detectors, it is also possible to optimize $C(\mathbf{x}, \boldsymbol{\theta})$ for a given $\mathbf{x}$; however, in practice, this optimization-based approach tends to be slower than running a sampled set of detectors that are just linear classifiers. We experimentally compare the performance of the detector optimization versus sample set approaches in Section 7.2.

3.1 Multiplicative Kernel Construction

Assume $C(\mathbf{x}, \boldsymbol{\theta})$ can be factorized into the product of two Hilbert space representations $\boldsymbol{\phi}_x(\mathbf{x})$ and $\boldsymbol{\phi}_\theta(\boldsymbol{\theta})$ with a matrix $\mathbf{V}$:

$$C(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}_x(\mathbf{x})^T \mathbf{V} \boldsymbol{\phi}_\theta(\boldsymbol{\theta}). \quad (2)$$

If we use the kernel trick, the above factorization yields a dual representation given by a product of two kernels:

$$C(\mathbf{x}, \boldsymbol{\theta}) = \sum_{i \in SV} \alpha_i k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta}) k_x(\mathbf{x}_i, \mathbf{x}), \quad (3)$$

where $k_\theta(\boldsymbol{\theta}, \boldsymbol{\theta}') = \boldsymbol{\phi}_\theta(\boldsymbol{\theta})^T \boldsymbol{\phi}_\theta(\boldsymbol{\theta}')$ and $k_x(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_x(\mathbf{x})^T \boldsymbol{\phi}_x(\mathbf{x}')$. α_i is the weight of the ith support vector. To apply a standard SVM classifier training algorithm on (3), we can virtually concatenate $\boldsymbol{\theta}$ with $\mathbf{x}$ into a single variable:

$$C([\mathbf{x}, \boldsymbol{\theta}]) = \sum_{i \in SV} \alpha_i k_c([\mathbf{x}_i, \boldsymbol{\theta}_i], [\mathbf{x}, \boldsymbol{\theta}]), \quad (4)$$

where $k_c([\mathbf{x}_i, \boldsymbol{\theta}_i], [\mathbf{x}, \boldsymbol{\theta}]) = k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta}) k_x(\mathbf{x}_i, \mathbf{x})$. The product of two positive semidefinite functions is still a positive semidefinite function. Each training example is a tuple $(\mathbf{x}, \boldsymbol{\theta})$ in our method, with a label $y \in \{+1, -1\}$. We give details of SVM training process in Section 4.

There is also an interesting interpretation of feature sharing if we combine α_i with $k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta})$,

$$C(\mathbf{x}, \boldsymbol{\theta}) = \sum_{i \in SV} \alpha_i'(\boldsymbol{\theta}) k_x(\mathbf{x}_i, \mathbf{x}), \quad (5)$$

where $\alpha_i'(\boldsymbol{\theta}) = \alpha_i k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta})$. Feature sharing is implicitly achieved by sharing support vectors. When $k_\theta(\cdot, \cdot)$ is strictly nonnegative, e.g., a radial basis function (RBF) kernel, (5) can be interpreted as reweighting the support vectors so that only those having parameters similar to $\boldsymbol{\theta}$ are assigned high weights. Fewer support vectors have to be taken into account in a local subregion in $\boldsymbol{\theta}$ space.

3.2 Construction of Detectors for Prediction When k_x Is a Linear Kernel

We are particularly interested in the case when k_x is a linear kernel, because linear classifiers are fast in practice. Furthermore, when the foreground variation is fixed to a particular $\boldsymbol{\theta}$ (e.g., faces of a fixed rotation angle 45 degrees), a linear classifier is usually good enough as a detector.

When k_x is a linear kernel, a linear classifier $\mathbf{w}(\boldsymbol{\theta})$ can be constructed as a weighted sum of support vectors following (5):

$$\begin{aligned} C(\mathbf{x}, \boldsymbol{\theta}) &= \sum_{i \in SV} \alpha_i'(\boldsymbol{\theta}) k_x(\mathbf{x}_i, \mathbf{x}) \\ &= \sum_{i \in SV} \alpha_i'(\boldsymbol{\theta}) (\hat{\mathbf{x}}_i^T \hat{\mathbf{x}}) = \mathbf{w}(\boldsymbol{\theta})^T \hat{\mathbf{x}}, \end{aligned} \quad (6)$$

where $\mathbf{w}(\boldsymbol{\theta}) = \sum_{i \in SV} \alpha_i'(\boldsymbol{\theta}) \hat{\mathbf{x}}_i^T$ and $\hat{\mathbf{x}} = [1, \mathbf{x}^T]^T$. Thus, a set $\{\mathbf{w}(\boldsymbol{\theta}_1), \mathbf{w}(\boldsymbol{\theta}_2), \ldots, \mathbf{w}(\boldsymbol{\theta}_s)\}$ can be precalculated at the offline stage, and used in the detection stage as subclass detectors in a partition-based method. The set of $\{\boldsymbol{\theta}_k\}$ can be sampled denser in our approach than in a partition-based method because we do not need to partition the foreground class and maintain ample training examples for each subclass. The details of the sampling process for generating a set of detectors are given in Section 5. Note that once this set of detectors is constructed, the within-class kernel k_θ does not have to be evaluated during detection. Only the dot products between $\mathbf{w}(\boldsymbol{\theta}_k)$ and input $\mathbf{x}$ are evaluated during detection, as in (6).

1. In this paper, all vector variables are column vectors.

3.3 Discussion on Regression Methods and High-Dimensional $\boldsymbol{\theta}$

The proposed method is different from regression methods, e.g., [1], [6]. A regular regression method $\boldsymbol{\theta} = f(\mathbf{x})$ takes a feature vector x as input and outputs an estimated $\boldsymbol{\theta}$. However, it does not answer the question—*Is* $\mathbf{x}$ *a foreground object or a background patch?* A regression method will output a $\boldsymbol{\theta}$ anyway even if the input $\mathbf{x}$ is a background patch. To handle this question, a method must make a binary decision—yes or no, besides a $\boldsymbol{\theta}$ estimate. However, it is difficult to combine a binary classification with continuous $\boldsymbol{\theta}$ estimation in a single function or classifier.

Recent work [7] proposed a structured output regression approach for object localization. The object location and scale y are obtained via maximizing a function $\hat{y} = argmax_y f(x, y)$. This formulation is similar to our approach in that the learned function f also takes two inputs jointly—one for the input instance and one for the parameter being optimized. The search over y space can be made into a very efficient branch-and-bound process when a bag-of-features representation is used. However, it is not clear whether the branch-and-bound process can be applied with general feature representations, e.g., histogram of orientated gradient (HOG) features [11] or Haar wavelet features [58] which have been widely used in object detection tasks. Similar issues will arise if we use this method for $\boldsymbol{\theta}$ estimation.

Our method deals with this problem by running a number of linear classifiers at the detection stage. The output of each linear classifier is first compared with its threshold to make the binary decision: *Is* $\mathbf{x}$ *a foreground object or a background patch?* If all detection scores are below threshold, $\mathbf{x}$ is classified as a background patch. Otherwise, the detection scores of all linear classifiers are compared with each other to see which has the highest score. The $\boldsymbol{\theta}$ associated with the winner linear classifier is the $\boldsymbol{\theta}$ estimate of input $\mathbf{x}$. Although a number of detectors must be evaluated, each detector is a fast linear classifier. Furthermore, the classification process can always be made into a two-stage process, where a simple linear classifier eliminates trivial background patches quickly in the initial stage, as in [46], [58]. Our detectors are only evaluated at the second stage on those patches that pass the initial stage. The whole detection process is slower than running a single classifier, but still acceptable in practice.

Our method does not handle high-dimensional $\boldsymbol{\theta}$ directly. A standard option to deal with high-dimensional $\boldsymbol{\theta}$ is via a dimensionality reduction method, e.g., PCA, LLE [45], etc. The detector sampling can be done in the low-dimensional space and then mapped back to the original high-dimensional space if necessary. However, learning such a mapping or a low-dimensional manifold is beyond the scope of our work.

3.4 An Example on a Synthetic Data Set

Fig. 1 illustrates the basic idea of our approach using a synthetic data set, where the foreground class of brown points is parameterized by an angle θ. In the multiplicative formulation, k_θ is a Gaussian RBF kernel and k_x is a linear kernel. The local linear boundary (6) can be reconstructed as a weighted sum of support vectors in x_1, x_2 space. The figure shows two constructed linear classifiers $\mathbf{w}(23^\circ)$ and $\mathbf{w}(-30^\circ)$. Ideal local linear classifiers in this case are tangent lines to the circular boundary, and our result follows the tangent lines very well.

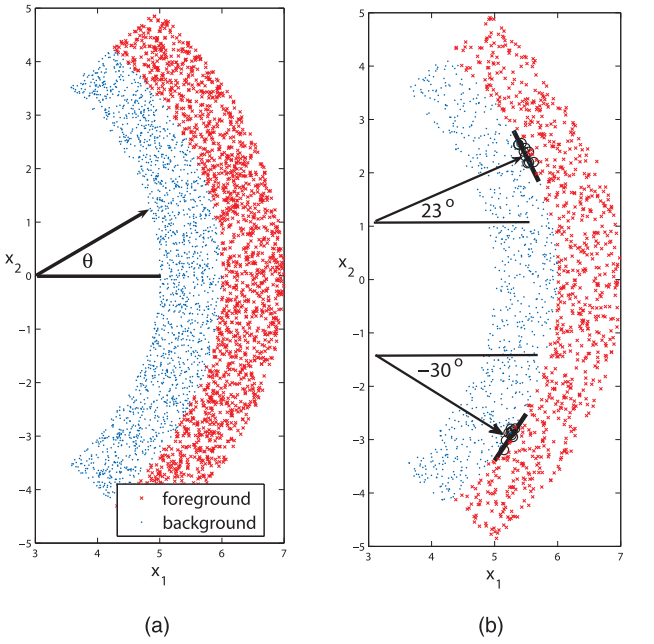


Fig. 1. An experiment on synthetic data. A family of multiplicative kernel classifiers is learned, where k_θ is an RBF kernel defined on θ, and k_x is a linear kernel defined on $\mathbf{x} = (x_1, x_2)^T$. The linear boundaries, for example, detectors $\mathbf{w}(23^\circ)$ and $\mathbf{w}(-30^\circ)$ are shown in (b). The circle points along each linear boundary are the reweighted support vectors (5) of weights > 0.5. These synthetic "foreground" and "background" classes were chosen to illustrate the idea that local discriminants can be learned jointly via multiplicative kernels, and then constructed at a given $\boldsymbol{\theta}$. (a) The original data (b) Two example detectors.

To illustrate the effect of reweighting support vector weights (5), we plot support vectors with $\alpha'(\boldsymbol{\theta}) > 0.5$ as circles for each of the two linear boundaries. For each of the two example linear classifiers, the highest weighted support vectors are always those close to the boundary and have similar angles to the chosen angles 23° and -30°.

3.5 Nonparametric k_θ

In some problems, parametric forms of foreground within-class variations may not be readily available. For example, there are numerous degrees of freedom in the human hand and the human body. Manual annotation of a large real image data set of hand shapes or body poses can be very expensive, tedious, and prone to errors. For such cases, we propose a nonparametric formulation for the within-class kernel k_θ.

To understand the usage of the nonparametric k_θ, we need to examine the role of the parametric k_θ in feature sharing, as outlined in Section 3.1. When k_θ is defined over continuous $\boldsymbol{\theta}$, two training samples with close $\boldsymbol{\theta}$ values should obtain a high k_θ score, and thus are more likely to share features. Conversely, training samples that are far from each other in $\boldsymbol{\theta}$ space are less likely to share features, and should obtain a small k_θ score. The nonparametric formulation attains the same behavior. Another way to understand the usage of the nonparametric k_θ is by considering k_θ as it varies the weights of support vectors. Intuitively, different foreground examples may have different "confusing" background examples. Thus, a support vector from the background class may not be globally useful for all foreground examples. The k_θ only gives higher

weights to the most useful support vectors for a particular foreground state.

We define a kernel $k_\theta(i,j)$ on a finite set I, which is the set of indices of foreground training examples. According to the Moore-Aronszajn theorem, there is a unique Hilbert space of functions $\boldsymbol{\phi}_\theta(i)$ on I for which $k_\theta(i,j)$ is a reproducing kernel, as long as $k_\theta(i,j)$ is a symmetric positive-definite kernel on I. Thus, as long as we can define a symmetric positive-definite kernel $k_\theta(i,j)$ on I, we can carry on the SVM training process, just like the parametric case in Section 3.1.

A straightforward design of a symmetric positive-definite kernel k_θ employs a nonparametric similarity/distance measure, e.g., bidirectional chamfer edge distance [3], [16] or shape context distance [4]. These distance metrics have been used successfully to measure within-class similarities for object classes like hand shape and body pose. Based on a distance measure D, a kernel function can be defined [35] as

$$k_\theta(i,j) = exp\big(-\eta D(\mathbf{z}_i, \mathbf{z}_j)\big), \tag{7}$$

where $\mathbf{z}_i$ and $\mathbf{z}_j$ are representations (e.g., edge images) of the foreground training samples indexed by i and j to calculate distance D. The representation $\mathbf{z}$ is selected to be suited for describing within-class variations. By adjusting η, we can make k_θ symmetric positive-definite on all pairs $i,j \in I$. Note that when this $k_\theta(i,j)$ is not defined on a finite set I, it may not always be a valid Mercer Kernel as the $k_\theta(i,j)$ defined in (7) is not guaranteed to be positive semidefinite on any $\mathbf{z}_i$ and $\mathbf{z}_j$. Thus, this nonparametric kernel representation cannot be used in a conventional SVM classifier. But when it is defined on a finite set as a within-class kernel in our approach, it can be made a valid Mercer kernel by selecting proper η.

When k_x is a general linear kernel, we can obtain a linear classifier $\mathbf{w}(i)$ for a particular training sample indexed by i:

$$\begin{aligned} C(\mathbf{x}, i) &= \sum_{j \in SV} \alpha_j k_\theta(i,j) k_x(\mathbf{x}_j, \mathbf{x}) \\ &= \sum_{j \in SV} \alpha'_j(i)\big(\hat{\mathbf{x}}_j^T \hat{\mathbf{x}}\big) = \mathbf{w}(i)^T \hat{\mathbf{x}}. \end{aligned} \tag{8}$$

After SVM training, we are able to construct linear classifiers $\mathbf{w}(i)$, for any foreground example i. This can be regarded as an extreme case of a partition-based method where each subclass is a singleton of just one foreground example. However, there may exist redundancy among $\mathbf{w}(i)$ since some of $\mathbf{w}(i)$s may be very similar to each other. One representative will be enough for a group of similar $\mathbf{w}(i)$s. We discuss mode-finding methods in Section 5 to handle this issue.

4 Detector Training

In this section, we give details on how to train the model defined in the previous section. A constraint generation process for SVM learning is described first. Then, we describe how to incorporate image masks in training, if they are available. This can help reduce the influence of background clutter and can also enable foreground object segmentation during detection.

4.1 Constraint Generation Process

The training samples are in the form of tuples. In the following paragraphs, we present formulations in the parametric case when $\boldsymbol{\theta}$ is available in training. Thus, the tuples are in the form $(\mathbf{x}, \boldsymbol{\theta})$. The nonparametric case has a similar formulation by replacing $\boldsymbol{\theta}$ with foreground example index i. The constraints in the SVM formulation are as follows:

$$\begin{aligned} min \quad & \frac{1}{2}\|\mathbf{V}\|_2^2 + C\sum_k \xi_k, \\ s.t. \quad & \xi_k \geq 0, \forall k \\ & y_k[\boldsymbol{\phi}_x(\mathbf{x}_k)^T \mathbf{V} \boldsymbol{\phi}_\theta(\boldsymbol{\theta}_k)] \geq 1 - \xi_k, \forall k, \end{aligned} \tag{9}$$

where $\|\mathbf{V}\|_2$ is the Frobenius norm and $y_k \in \{+1, -1\}$ is the annotation of a training tuple $(\mathbf{x}_k, \boldsymbol{\theta}_k)$, assigned by the definition in (1). Because we use a linear kernel for k_x, the constraint in (9) can be simplified as:

$$\begin{aligned} y_k \hat{\mathbf{x}}_k^T \mathbf{w}(\boldsymbol{\theta}_k) \geq 1 - \xi_k, \forall k, \\ \mathbf{w}(\boldsymbol{\theta}_k) = \mathbf{V}\boldsymbol{\phi}_\theta(\boldsymbol{\theta}_k). \end{aligned} \tag{10}$$

Each foreground training sample $\mathbf{x}$ is associated with its corresponding ground truth $\boldsymbol{\theta}$ to make a positive tuple. By definition, a background training sample $\mathbf{x}$ can be associated with any $\boldsymbol{\theta}$ to make a negative tuple. The number of such combinations (constraints) is huge. We therefore employ a constraint generation process to add violated constraints as new constraints iteratively until convergence. Note, we only use $\boldsymbol{\theta}$ values that appear with foreground training examples. Thus, the total number of constraints is still finite. A similar constraint generation process has been employed in [7] for structured output regression.

The training process starts with assigning each background feature vector $\mathbf{x}$ a randomly selected $\boldsymbol{\theta}$ from foreground training examples to form initial constraints. Then, at each iteration, all constraints are evaluated to find the most violated ones, which are added as new constraints in the next iteration of SVM training. Note we are able to evaluate all of the constraints with the current model via the product in (10), although we cannot include all constraints at the same time in SVM quadratic programming. The pseudocode for this process is given in Fig. 2. In the pseudocode, the degree of a constraint violation is given by $e_k = 1 - y_k \hat{\mathbf{x}}_k^T \mathbf{w}(\boldsymbol{\theta}_k)$. Violated constraints are ranked by their e_k values and the top N_s of them are added at each iteration. The number N_s may depend on the size of the training set. A rule of thumb used in our experiment is that N_s equals one-tenth of the total number of background training examples.

When the constraint generation process stops with all constraints satisfied ($\xi_k = 0$ for those constraints not included in the final round of SVM optimization), we can show that the obtained model $\mathbf{V}_c$ is equivalent to the model $\mathbf{V}_a$ that is optimized using all constraints in one batch. Let D_c, D_a denote the sets of constraints that are taken into account in training of $\mathbf{V}_c$ and $\mathbf{V}_a$, respectively. Let g_c and g_a denote the final goal values after the model is optimized with D_c and D_a, respectively. By definition $D_c \subseteq D_a$. Because g_a is the minimized goal value with more constraints, we have $g_c \leq g_a$. On the other hand, although both $\mathbf{V}_c$ and $\mathbf{V}_a$ satisfy all the constraints D_a, $\mathbf{V}_a$ is the one that is explicitly optimized with D_a to achieve the minimal goal value g_a.

Given

Foreground training samples $X_f = \{\mathbf{x}_{f_1}, \ldots, \mathbf{x}_{f_n}\}$,

Background training samples $X_b = \{\mathbf{x}_{b_1}, \ldots, \mathbf{x}_{b_n}\}$,

Parameters of foreground training samples $\Theta = \{\boldsymbol{\theta}_{f_1}, \ldots, \boldsymbol{\theta}_{f_n}\}$.

Initialize

Assign each foreground training sample $\mathbf{x}_{f_i}$ its actual parameter $\boldsymbol{\theta}_{f_i}$, and each background training sample $\mathbf{x}_{b_i}$ a random value from Θ to form tuples $(\mathbf{x}, \boldsymbol{\theta})$. If object masks are available for foreground examples, the mask of $\boldsymbol{\theta}$ is applied on $\mathbf{x}$.

While

1) Compute Gram matrix K_c of current tuples, where $k_c([\mathbf{x}_i, \boldsymbol{\theta}_i], [\mathbf{x}_j, \boldsymbol{\theta}_j]) = k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) k_x(\mathbf{x}_i, \mathbf{x}_j)$.
2) Carry out SVM training using K_c.
3) Examine all possible tuples to find violated constraints (tuples).
4) If it converges or the max number of iterations reached, break.
5) Add a fixed number N_s of mostly violated constraints (tuples) as new tuples.

End

Fig. 2. Pseudocode for bootstrap training with parametric within-class kernel k_θ. For the case of nonparametric k_θ, the set Θ is replaced by the set of indices of foreground training samples.

Thus, $g_a \leq g_c$. So, we have $g_a = g_c$. Since SVM training is a convex optimization (quadratic programming), the optimal solution is unique and both $\mathbf{V}_c$ and $\mathbf{V}_g$ achieve it. So, the two models are equivalent.

4.2 Including Object Masks in Training

There are situations in practice when object masks can be obtained during training data acquisition, for instance, when the background is known. In such cases, masks can be exploited to reduce the influence of background regions inside the detection window during both training and testing. Ideally, whether a training object is captured with a highly textured background or with a smooth background should have minimum impact on the final classifier. However, when the bounding box is not tight, features from textures of background regions may be included in a foreground training example. For instance, in gesture recognition, many training hand images may be captured in front of a fixed camera. The same background region behind the hand may be included in hand images for many frames. There is a risk that the texture from the same background regions will be reinforced as features from "hand" during training of a hand classifier.

When each training sample has a foreground mask, features from outside the mask can be ignored. For instance, to calculate the color histogram of a foreground object, only those pixel colors from inside the mask region may be considered. When the features have local support and are ordered according to their spatial arrangement, e.g., HOG features [11] or Haar wavelet features [58], applying object masks in feature extraction means that the feature components that have support from outside the object masks have zero values.

To be consistent, masks should also be applied to the background training samples during feature extraction. As mentioned earlier, each background training sample is associated with a foreground parameter $\boldsymbol{\theta}$ or an index i of a foreground training sample. Therefore, the mask of the foreground training sample with $\boldsymbol{\theta}$ or i is applied to this background training sample during training.

Since a mask is associated with a $\boldsymbol{\theta}$ or i, each detector $\mathbf{w}(\boldsymbol{\theta})$ or $\mathbf{w}(i)$ can be associated with a corresponding mask. In our approach, the image mask of a detector is calculated as a weighted sum of image masks from foreground support vectors using the support vector weights α'_i. This mask of continuous values can be binarized by a threshold, when a binary image mask is required. For instance, segmentation can be produced by superimposing the detector's binary image mask onto a detected object.

Because applying a linear SVM classifier is equivalent to summing up dot products between support vectors and $\mathbf{x}$, the masks of support vectors are implicitly applied on $\mathbf{x}$ during this linear classification $\mathbf{w}^T\mathbf{x}$. More precisely, masks do not have to be explicitly applied on a test input when both of the following two conditions are met:

1. The features have local support like HOG [11] or Haar wavelet features [58].
2. $k_x(\mathbf{x}_1, \mathbf{x}_2)$ is based on the dot product $\mathbf{x}_1^T\mathbf{x}_2$, e.g., the linear or polynomial kernels.

Object masks or shape priors were also used in previous work [8], [61], [24], [59], [63]; however, in our method, no decomposition of the image mask into local edgelets or parts is needed. If available, context cues may help the detection process, e.g., following [11], [18].

5 Detection and Foreground State Estimation

After training as described in Section 4, we are able to construct a linear classifier as a detector for any parameter $\boldsymbol{\theta}$ or any foreground sample index i. However, in a real-world application like multiview face detection, neither object locations (there could be multiple faces or none in an image) nor object foreground states are known. Thus, during detection, a scanning window process is employed using detectors associated with a predefined representative set of $\boldsymbol{\theta}$ or i, which covers typical foreground state variations. The foreground state annotation $\boldsymbol{\theta}$ associated with the detector of the highest detection score is assigned to a detected object as a foreground state estimate. As will be described in the rest of this section, there are a number of ways to determine the representative set of $\boldsymbol{\theta}$ or i that is used in generating the set of detectors. We will focus on two methods: uniform sampling over the training set and finding representative samples via mode finding (clustering).

5.1 Generating a Sample Set of Detectors

Assume that the training set provided a fair and representative sampling of the foreground class. If the foreground states are annotated as parameters $\boldsymbol{\theta}$, e.g., view angles or rotation angles, a representative set of $\boldsymbol{\theta}$ can be obtained by uniformly sampling from the parameters of foreground training examples. In special cases when prior information about the parameter distribution is provided, e.g., in object tracking, where temporal information is propagated from frame to frame, importance sampling can be employed instead to draw parameter samples to comprise a representative set of $\boldsymbol{\theta}$. In our experiments, we obtain satisfactory results via uniform sampling for detection and parameter estimation applications, and via importance sampling for tracking.

In the nonparametric case, uniform sampling over the foreground training samples can also be used to generate the detector family, assuming that the training set provides a fair and representative sample of the foreground class. However, we have found that a mode-finding technique is more effective in practice.

5.2 Mode-Finding for Nonparametric Detectors

In the nonparametric case, we can use mode finding to reduce the redundancy in the detector set. Clustering is a straightforward option for mode finding. We define a similarity measure S_α on the support vector weights $\boldsymbol{\alpha}'(i)$ of foreground examples,

$$S_\alpha(i,j) = \frac{\boldsymbol{\alpha}'(i)^T \boldsymbol{\alpha}'(j)}{\|\boldsymbol{\alpha}'(i)\| \cdot \|\boldsymbol{\alpha}'(j)\|}, \tag{11}$$

where $\boldsymbol{\alpha}'(i) = [\alpha'_1(i), \alpha'_2(i), \ldots, \alpha'_n(i)]^T$ are the support vector weights for i, as defined in (8). Each cluster is regarded as a mode that represents a variation of the foreground class. The proper number of modes is decided via cross-validation to obtain acceptable detection accuracy. The cluster centers comprise the representative set used in the online stage.

Note that our clustering process is different from that in a partition-based method. In a partition-based method, the training examples are clustered before subclass detectors are trained. In our approach, the detectors are first trained and then the clustered are identified.

Interestingly, annotation of training examples can be made efficient with mode finding because we can start with an unannotated foreground class to train detectors first, and then after mode finding only the representative set needs to be annotated with foreground states. The complete training process with a nonparametric kernel is as follows:

1. Train the model with a nonparametric kernel. At this step, no foreground state annotations are needed. Only labels of foreground $(+1)$ versus background (-1) and a distance D are required. After training, each detector $\mathbf{w}(i)$ is associated with a unique foreground example, as in (8).
2. Use mode finding to find clusters in the foreground class. Only those $\mathbf{w}(i)$ of cluster centers will be employed in detection.
3. Annotate the cluster centers (foreground examples) with foreground states. The foreground states can be continuous variables like rotation angles and pose parameters, or nominal values like "standing," "walking," and "running."

An obvious advantage of this strategy is that only a small portion of the foreground training data must be annotated. This can save a significant amount of effort that might be needed to label all training samples for the same purpose.

6 Tracking with Multiplicative Kernel Detectors

Tracking objects that undergo large appearance changes is challenging, e.g., tracking articulated objects like human hands or multiview objects like faces and vehicles. Commonly used cylinder models [51] or edge templates [54] usually require strong temporal models and manual initialization to achieve robust tracking, particularly in cluttered scenes.

One way to cope with a cluttered background is to use detectors that are trained against representative background examples. Such a strategy was employed in the in "tracking-by-detection" approaches of [2], [29], [36], where the tracking performance is enhanced by using the detectors that handle cluttered background and variations of the foreground class.

Our detectors trained with multiplicative kernels can also be employed in a tracking-by-detection framework. A brute force way to implement tracking with parametric detectors that are trained with multiplicative kernels is via frame-by-frame detection. Although the object locations and foreground states can be recovered in this way, it can be expensive to run a dense scan on each frame with all detectors. We instead propose a tracking approach that incorporates temporal information to make the tracking process more efficient.

We formulate the tracking process in a standard prediction-update framework as in particle filtering and CONDENSATION [22]. For an existing object, given its observations $\mathbf{Z}_t = (\mathbf{z}_1, \ldots, \mathbf{z}_t)$ up to time t, we estimate the current state $\mathbf{s}_t$ by the following steps:

1. Prediction: $p(\mathbf{s}_t|\mathbf{Z}_{t-1}) = \int p(\mathbf{s}_t|\mathbf{s}_{t-1})p(\mathbf{s}_{t-1}|\mathbf{Z}_{t-1})d\mathbf{s}_{t-1}$,
2. Update: $p(\mathbf{s}_t|\mathbf{Z}_t) \propto p(\mathbf{z}_t|\mathbf{s}_t)p(\mathbf{s}_t|\mathbf{Z}_{t-1})$.

We define $\mathbf{s}_t = (\mathbf{l}_t, \theta_t)$, where $\mathbf{l}_t$ is the location (including scale) and θ_t is the pose parameter. We assume independence between $\mathbf{l}_t$ and θ_t. Thus,

$$p(\mathbf{s}_t|\mathbf{s}_{t-1}) = p(\mathbf{l}_t|\mathbf{l}_{t-1})p(\theta_t|\theta_{t-1}). \quad (12)$$

During importance sampling, $\mathbf{s}_t$ is factorized into $\mathbf{l}_t$ and θ_t to reduce the number of dimensions of samples. In practice, such factorization is reasonable since position and $\boldsymbol{\theta}$ tend to be independent. We also assume zero-mean Gaussian distributions for both $p(\mathbf{l}_t|\mathbf{l}_{t-1})$ and $p(\theta_t|\theta_{t-1})$, i.e., $\mathbf{l}_t - \mathbf{l}_{t-1} \sim N(0, \Sigma_l)$ and $\theta_t - \theta_{t-1} \sim N(0, \Sigma_\theta)$. The covariance matrices Σ_l and Σ_θ are chosen according to the typical state-changing speed of foreground objects. The Gaussian distribution assumption follows a common choice for the proposal distribution in the particle filtering framework [22].

In the update step, $p(\mathbf{z}_t|\mathbf{s}_t)$ is evaluated using our detectors, i.e., given a sample $\hat{\mathbf{s}}_t = (\hat{\mathbf{l}}_t, \hat{\theta}_t)$, the detector associated with $\hat{\theta}_t$ is evaluated at location $\hat{\mathbf{l}}_t$ to give a score $C(\hat{\mathbf{z}}_t, \hat{\theta}_t)$ that determines whether the observation $\hat{\mathbf{z}}_t$ at location $\hat{\mathbf{l}}_t$ should be accepted or rejected as an instance of the object with parameter $\hat{\theta}_t$. The sample $\hat{\mathbf{s}}_t$ is discarded if the detector classifies it as from the background class. We define $p(\mathbf{z}_t|s_t) = \frac{1}{1+exp(C(\mathbf{z}_t,\theta_t)+r)}$ as a sigmoid function, which has been shown in [41] to fit well as a probability model of SVM classification scores. r is determined by classification scores on the training examples.

Our posterior distribution is represented as a mixture over detector modes. Thus, the number of particles drawn for a mode is proportional to the weight of the mixture, which is set to be equal to the weight of the mode. The sum of sample weights is normalized to 1 in each iteration. Nonmaximum suppression is applied on locations of accepted samples to produce a set of putative locations for tracked objects in the current frame.

In our tracker implementation, to deal with the entrance of new objects, exhaustive detection is triggered at every k frames. The parameter k is selected according to the expected entrance rate for new objects. Once a foreground object is detected during exhaustive detection, a tracking process starts to track it until it exits the scene. Exitance of objects is also automatically handled; once an object exits the scene, samples that are not located on foreground objects in the next frame will be rejected by the detectors.

7 Experiments

In this section, we describe the evaluation of the proposed method in three applications: hand detection and shape estimation, multiview vehicle detection and tracking, and multiview face detection and tracking. For the purpose of these experiments, HOG [11] features are employed for $\mathbf{x}$ on all data sets; while other features could be possible, we chose the HOG feature representation since it is widely used. The detectors of our method are trained using a modified version of SVMlight [23] with multiplicative kernels. The between-class kernel k_x is always a linear kernel, and the within-class kernel k_θ is a Gaussian RBF kernel or a nonparametric kernel (7) depending on the data set. Our results are compared with results obtained via methods proposed in [37], [55], [60], [62]. All approaches are implemented in Matlab, and run on a 2.6 GHz AMD Opteron 852 processor.

The parameters of HOG features are the same as in [11]. The only difference is the detection window size for different objects. We use a 48×48 window on hand data sets, a 90×90 window on the vehicle data set, and a 60×60 window on the multipose face data set.

In our experiments, we use two clustering techniques in our approach for mode finding: agglomerative clustering and spectral clustering [50]. Agglomerative clustering is a greedy algorithm that runs faster but may create clusters of unbalanced sizes. Spectral clustering creates balanced clusters but runs slow for large numbers of examples. We use agglomerative clustering on the hand data set and spectral clustering on the vehicle data set, mainly because of the sizes of the foreground training set.

7.1 Hand Detection and Segmentation with Nonparametric k_θ

We first conduct an experiment in hand detection for sign language data. The hand data set is collected from two sources of sign language video sequences: Flemish Sign Language data [10] and American Sign Language data [33]. In total, there are 17 signers. The data set comprises a training set of 3,005 hand images and a test set of 2,270 hand images. The test set and training set are disjoint. The hand images are not annotated with hand shape parameters. For the training images, corresponding hand silhouettes are also provided. About 70 percent of the hand silhouettes are automatically segmented by skin color models or simple background models. The rest are obtained manually. Example frames are shown in Fig. 3. This data set is available for download [64].

The background image set contains images of outdoor and indoor scenes. This set is separated into disjoint training and test sets, which contain 300 images each. Five thousand image patches are collected as samples from each image set to be used as training or testing background samples.

Training of detectors is done as described in Section 4. k_x is a linear kernel. For the within-class kernel k_θ, the nonparametric form of (7) is used since parameter annotations are unavailable for the training images. The distance measure D is the bidirectional chamfer edge distance [16] between hand images. With $\eta = 1$, the Gram matrix of k_θ is positive definite on the training set. In total, about 14,300 tuples $(\mathbf{x}, i)$ are selected as support vectors during training. This is still a small portion of all possible training tuples, which can be as many as $3{,}005 + 3{,}005 \times 5{,}000$. Because each background example $\mathbf{x}$ can be combined with different i (foreground example index) to make a training tuple $(\mathbf{x}, i)$. Agglomerative clustering is used in the mode-finding process to generate the detector sample set of 1,242 hand modes. The number of modes is determined by the stopping criterion in agglomerative clustering, when the similarity measure of (11) between any two clusters is below a threshold value 0.7. The total training

Fig. 3. Example sign language sequences from which the training and test hand images are obtained.

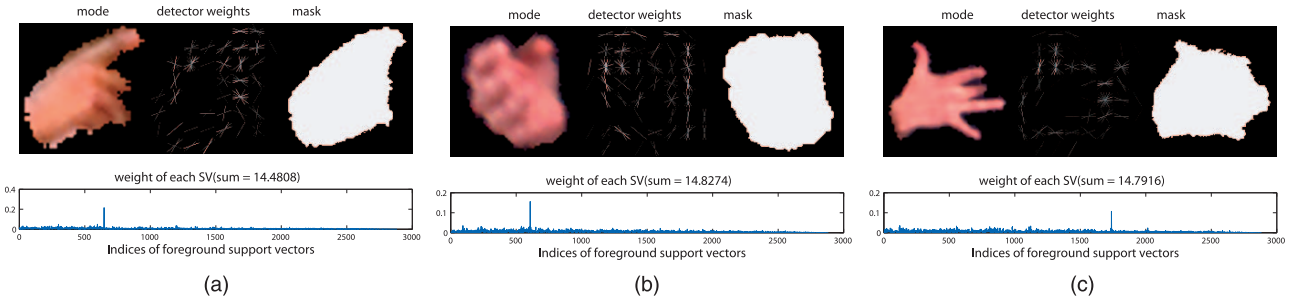


Fig. 4. Three hand clusters are displayed with their cluster medoids, positive detector weights, and hand masks. For each cluster, the weights of foreground support vectors are displayed at the bottom.

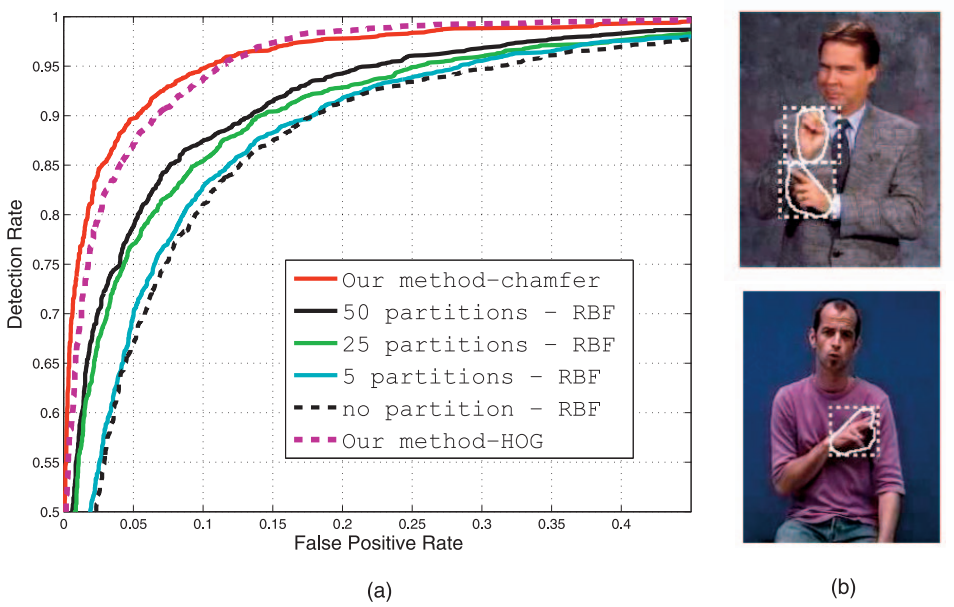


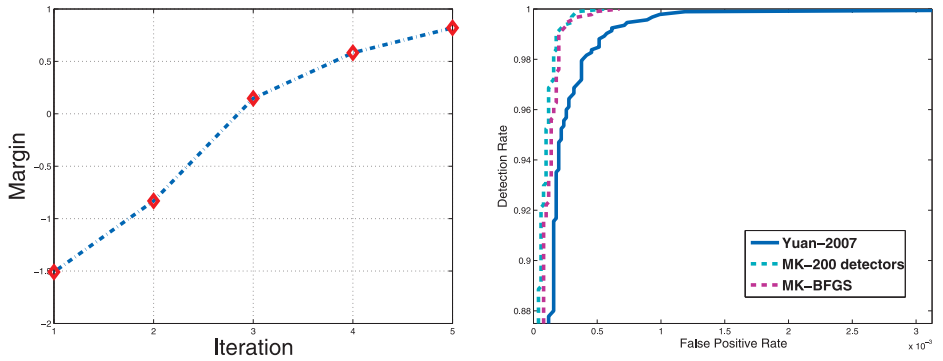
Fig. 5. Hand detection results: (a) ROC curves of different detectors for hand detection. "Our method-chamfer" uses the k_θ defined by chamfer edge distance. "Our method-HOG" uses the k_θ defined using RBF kernel in HOG feature space. (b) Example detection and segmentation results on the sign language test images.

time is about 30 minutes on a single 2.6 GHz AMD Opteron 852 processor.

Three out of the 1,242 hand clusters are illustrated in Fig. 4. The figure shows three images for each cluster: the image of the cluster medoid, the positive weights of the detector associated with the cluster medoid, and the mask for the medoid. The positive weights of a detector demonstrate how local edge orientations are weighted. The image mask of a cluster is computed as a weighted sum of image masks of support vectors for the top 50 weights, and then thresholded to obtain a binary image. While there could be different ways to construct an image mask, in our experiment, the resulting masks have appropriate sizes and shapes for this application.

For each cluster medoid shown in Fig. 4, a graph shows the distribution of support vector weights $\boldsymbol{\alpha}'$. Interestingly, although the weights have peaks on a few foreground support vectors, the sum of weights from low weight support vectors is substantial. This indicates that the contributions to the detector of a particular foreground variation come from a broad range of training samples, although each contribution may be small. One explanation is that very different hand shapes may still share segments of finger or palm boundaries.

Examples of the combined detection and segmentation results obtained with our method are shown in Fig. 5b. The segmentation result is obtained by applying the mask associated with the detector of the highest score on a detected hand. The segmentation obtained in this way is



(a) (b)

Fig. 6. Training and test performance of the proposed approach on the hand shape data set [62]: (a) The change of margin during the iterative training process. The margin is defined as the minimum classification score of positive training tuples minus the maximum score of all negative training tuples at an iteration. (b) Comparison of ROC curves on the hand shape data set with 2D parameters [62]. The detection rate is in the range between 88 and 100 percent.

only approximate; nonetheless, the shapes are matched well and the segmentation is obtained at nominal extra cost. The segmentation result from our method can be used to mask the image for a hand shape estimation module in sign language analysis or used as initialization to a method that requires segmented input.

When the detectors are applied on the frames of the Flemish and American sign language sequences [10], [33], they can detect most of the hand shapes in the test set correctly. The detectors may fail to detect a hand when there is strong motion blur or it is partially occluded. False positives happen occasionally in regions of strong textures.

For experimental comparison, a partitioning-based method [37] is trained as follows: First, clustering of hand subclasses is obtained via k-means with euclidean distance of HOG features, and then the detector for each subclass is trained using SVM with an RBF kernel. The η of the RBF kernel is 0.1, which is chosen empirically to maximize the accuracy. Each subclass is also associated with a mask which is the union of all training masks belonging to this subclass. The features from outside a subclass mask are ignored during detection. The accuracy of the partition-based method improves until the number of partitions increases to about 50 partitions. Further increases of the number of partitions do not yield significant improvement. We also implemented another version of our approach using the RBF kernel in HOG features space as within-class kernel k_θ. It gives results that are comparable with our method when the chamfer edge kernel is used.

The detection accuracy of the different methods is summarized in the ROC curves of Fig. 5a. As can be seen in the graph, our method outperforms the partition-based methods by a clear margin on this data set. Compared to the best partition-based method (50 partitions), our method using the chamfer edge kernel improves detection rate from 80 to 90 percent at a false-positive rate of 5 percent. At the detection rate of 80 percent, our method reduces the false-positive rate from 5.3 to 1.7 percent. We report the rate of false positives per window instead of false positives per image mainly because of the varying sizes of test images. The smallest image size is 98×58 and the largest is 640×480.

To better understand the accuracy trade-off in using the representative subset of detectors determined via mode finding, we compared performance against using all detectors. The detectors are trained using the chamfer edge kernel. At a fixed false-positive rate of 5 percent, when all detectors are used (3,005 in total), the detection rate is 90.1 percent. With our mode-finding approach, the detection accuracy is 90.0 percent, 88.4 percent, and 83.5 percent with 1,242 modes, 938 modes, and 300 modes, respectively. The online detectors used in our approach (1,242 detectors) achieve the same accuracy, while reducing the number of detector evaluations by about two-thirds. In contrast, when we use uniform sampling to obtain 1,242 online detectors, the average detection accuracy over 10 trials is 88.1 percent with a standard deviation of 0.49 percent at the false-positive rate of 5 percent.

7.2 Hand Detection and Shape Estimation with Parametric k_θ

In the second experiment, we detect instances of a hand shape class that is parameterized by two angles from a cluttered background and estimate the two angles simultaneously. In the hand shape data set of [62], each hand image is given two angles within the range $[0, 90]$—one for the angle of the index finger, the other one for the in-plane rotation. There are 1,605 hand images for training and 925 for testing. There are also 5,500 background training samples and 50,000 background test samples, cropped from real background images or hand images of other hand shapes not included in the target hand shape class.

In our method, HOG features are computed in the same way as in [62]. The two angle parameters θ_1 and θ_2 are both normalized to $[0, 1]$. The within-class kernel k_θ is a Gaussian RBF kernel in the 2D parameter space, with $\frac{1}{\sigma^2} = 10$. The change of margin during constraint generation process is plotted in Fig. 6a. After SVM training, 200 parameter values $\boldsymbol{\theta}$ with corresponding detectors are uniformly sampled from the 1,605 parameter values associated with foreground training examples. This number of online detectors is determined to be adequate via cross-validation using training examples. These 200 detectors are used at the online stage.

We compare the performance of our formulation with a boosting-based approach [62]. The ROC curves of the detection result (hand versus background) are shown in Fig. 6b. As can be seen from the ROC curves, our method consistently outperforms that given in [62]. At a false-positive rate of 2×10^{-4}, our approach improves the true-positive rate from 94 to 99 percent. The partition-based method of

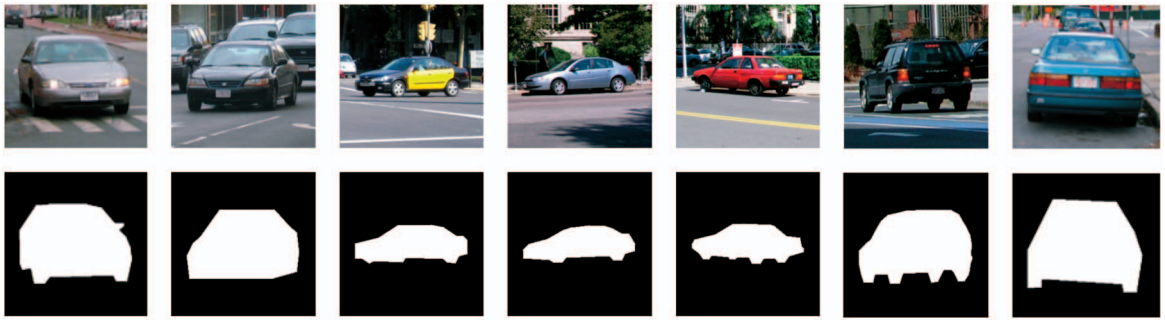


Fig. 7. Example images and their binary segmentation masks from the multiview vehicle data set used in [25].

[37] is compared in [62] on the same data set. With 25 subclasses, it achieves a true-positive rate of 91 percent at the false-positive rate 2×10^{-4}. Thus, our work is indirectly compared with [37]. In terms of training time, the multiplicative kernel-based method is about 10 times faster than the boosting-based method [62].

As noted earlier, it is possible to use nonlinear optimization of $C(\mathbf{x}, \boldsymbol{\theta})$ to determine the best $\boldsymbol{\theta}$ for a given input $\mathbf{x}$. We conducted a comparison of the sample-based versus optimization approach on this data set. Nonlinear optimization was accomplished in Matlab using a BFGS [34] quasi-Newton method with cubic polynomial search for the optimal step size. Because the multiplicative kernel may have multiple local minima, the (θ_1, θ_2) are initialized at 36 evenly distributed grid points in the box defined by the bottom-left corner $(0, 0)$ and the top-right corner $(90, 90)$. The ROC curve for this approach is included in Fig. 6b. The detection accuracy of BFGS is similar to using a sample set of 200 detectors, but BFGS runs much slower. On average, it takes about 8.2 gradient descent iterations to converge at each of the 36 initial points.

Our approach also estimates two parameters on each hand image. The mean absolute errors (MAE) of θ_1 and θ_2 are 6.7 and 4.6 degrees, respectively, by using a sample set of 200 online detectors, in contrast to 9.0 and 5.3 degrees in [62]. The BFGS optimization achieves better MAEs of 6.5 and 3.7 degrees. The partition-based approach [37] with 25 subclasses does not produce a parameter estimate, but a subclass label that is within a range of 18 degrees.

7.3 Multiview Vehicle Detection

In the next experiment, we look at a multiview vehicle detection problem. We evaluate the performance of the proposed method in two vehicle detection tasks, with comparison to previous approaches [55], [60]. In the first task, we detect vehicles appearing in city scenes. In the second task, we detect vehicles on highways.

City Scenario. For this experiment, we use a multiview vehicle data set [25], which is a subset of LabelMe [47] database. This subset contains 1,297 vehicles images. Each vehicle image also has a binary segmentation mask converted from the LabelMe annotation polygon. In [25], the data is split up into seven subcategories for car viewpoints approximately 30 degrees apart. Because of vehicle symmetry, the labeled angles cover a range of viewpoints from approximately -30 degrees to 180 degrees. Example images from this data set are shown in Fig. 7. These 1,297 vehicle images are separated into a training set of 866 images and a test set of 431 images. We collected background training and test image sets, which contain 432 and 344 outdoor street scene images, respectively. Most of the background images are from street scene images used in [11]. The rest are downloaded from the Web. The background image data sets are available for download [64].

The rotation angles only have seven distinct values at 30 degrees apart, which is too coarse for a continuous model. Therefore, in our approach, a nonparametric kernel is adopted. The nonparametric within-class kernel k_θ is an RBF kernel defined with an euclidean distance between HOG features. The kernel parameter $\eta = 0.2$. We implemented two versions of our approach. One is trained with binary image masks and the other one is without image masks. For both versions, the training process takes 10 iterations. For both versions, 280 modes are obtained by spectral clustering (normalized cuts [50]) after training. The number of modes is again determined by cross-validation of detection accuracy.

Performance is compared with Torralba's feature sharing method [55], Wu-Nevatia's tree based detector [60], and an RBF kernel SVM classifier with $\eta = 0.2$. In each method, the parameter settings were determined to optimize detection accuracy. For [55], the view angle subcategory labels of training images are provided in training since it is a multiclass detection method. In training it adds 4,000 weak classifiers in total and outputs seven subclass detectors. Each subclass detector is for a view angle subclass. In [60], the tree structure is mainly controlled by a splitting threshold θ_z. The best θ_z is found at 0.95 by cross-validation, which produces a tree of eight leaf nodes. The final numbers of weak classifiers in these eight subclass detectors are between 2,032 and 2,213. We tested our implementation of [60] on the pedestrian data set [11] and obtained comparable results to those reported in [60]. At a false-positive rate 10^{-4}, our implementation of [60] has a detection rate 93.4 percent.

In training, a bootstrap method is employed to collect nontrivial background examples for all methods, in the same way as in [11]. First, a linear SVM classifier is trained with an initial set of 10,000 training background patches. Then, we exhaustively search all the background training images with this linear SVM classifier to collect false-positive image patches ("hard examples"). In the scanning process, a total of 2,211 false-positive patches are collected. They are added to the initial 10,000 background training samples as the background training set for all methods. In total, there are about 8,300 tuples selected as support vectors in our method that utilizes image masks.

The detection performance of all methods in the first task is shown in the ROC curves of Fig. 8a. Compared with [60], our method with image masks improves the detection rate

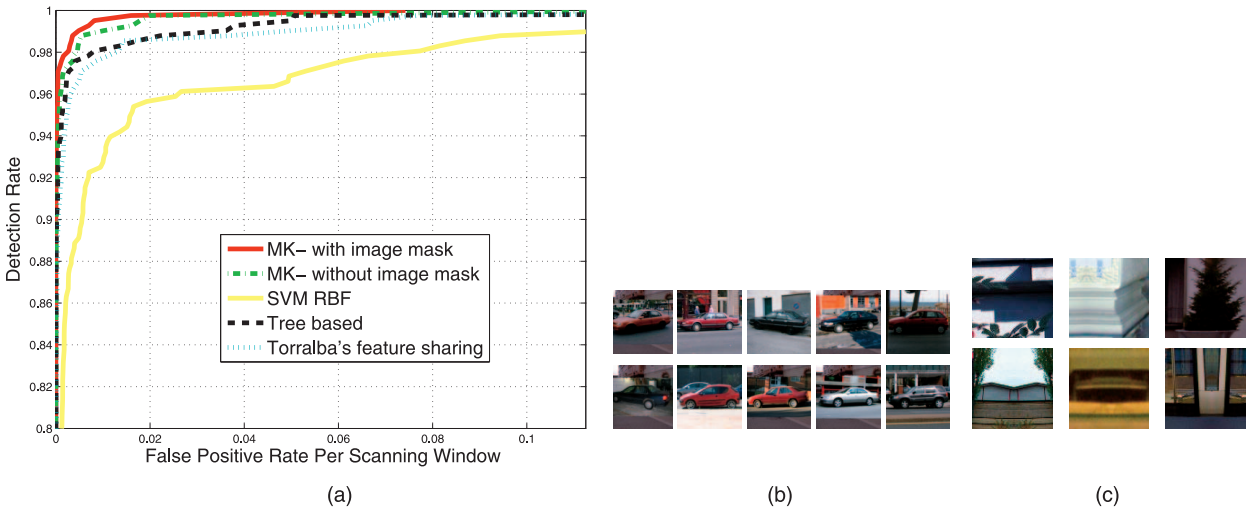


Fig. 8. Vehicle detection result on data set [25]: (a) ROC curves. (b) False-negative examples of our method. (c) False-positive examples of our method.

from 96.7 to 99.0 percent at the false-positive rate of 5×10^{-3}. At the detection rate of 99.5 percent, our method reduces the false-positive rate from 5 to 0.8 percent. On average, it takes 1.85×10^{-4} seconds for our method to evaluate a test example, and 4.40×10^{-4} seconds and 2.46×10^{-4} seconds for [55] and [60], respectively. However, the speed difference between all methods is not significant and we do not claim classification speed as an advantage of our approach.

In Figs. 8b and 8c, we show some false-negative examples and false-positive examples of our approach on the first task. The false negatives are collected at a fixed false-positive rate of 10^{-3}. The false positives are collected at a fixed detection rate 95 percent. Because the HOG features are based on gradient orientations and vehicles have symmetric shapes when viewed from certain angles, the false-positive examples also show symmetric patterns and strong edges.

Highway Scenario. For this experiment, we test our method on detecting vehicles on a highway. Differently from the first task where the vehicles are mostly captured in urban scenes, test sequence 5 of PETS 2001 vehicle data set is captured by two moving cameras on a highway, one facing the front and the other one facing the back of the vehicle. Example frames from the two cameras are shown in Fig. 9. In total, there are 2,867 frames for each camera. The frame size is 768×576.

In these two test sequences, the vehicles that are moving in the same direction with the cameras (i.e., vehicles bound in same direction) tend to be close to the cameras, and they are imaged at good pixel resolution. It is more challenging to detect the vehicles that are moving in the opposite direction, on the other side of the highway. These vehicles appear at smaller pixel resolutions and are partially occluded by the highway guard rail. For evaluation purposes, we manually annotated vehicles of sizes no

Fig. 9. Example frames and ground truth annotations of two cameras in test sequence 5 of the PETS 2001 data set. Although vehicles running close to the cameras have good resolutions, the actual challenges come from the vehicles running in the opposite direction across the fence. They usually have small resolutions and are partially occluded. Detection accuracy of these vehicles is a decisive factor in the ROC curves. (a) Forward view camera. (b) Rear view camera.

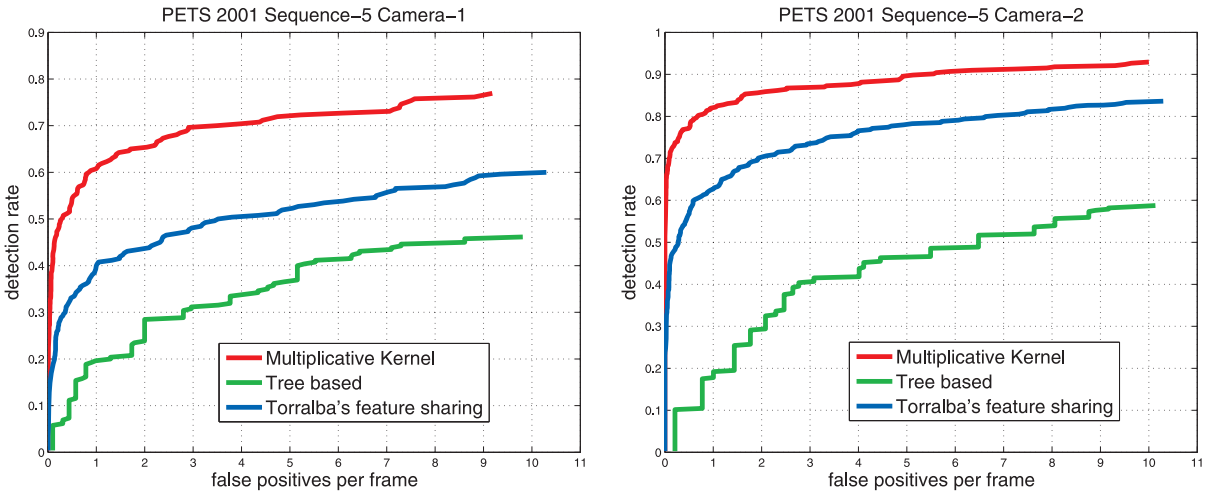


Fig. 10. Vehicle detection rate versus false-positive rate on sequence 5 of PETS 2001 data set. The proposed approach (Multiplicative Kernel) is compared with Wu-Nevatia's tree-based detector [60] and Torralba's feature sharing method [55].

smaller than 45×45 and occluded by less than one-third, in every 10th frame of each camera sequence.

As before, comparison is conducted between our approach and [55], [60]. For the purpose of fair comparison, we compared these methods with our detector without tracking. All methods detect vehicles frame by frame without temporal information. All 1,297 vehicle images from [25] with their horizontally flipped images are used as training samples. The settings of our method are the same as in the first task. For the tree-based method [60], the best splitting threshold is found again at 0.95, which produces a tree that has 19 leaf nodes. For the feature sharing method [55], view categories are provided and 4,000 weak classifiers are collected in training.

For evaluation, we consider a detection window as correct if it overlaps with the ground-truth annotation by more than 50 percent using the intersection-over-union criterion [30]. The detection performance of the three methods is summarized in Fig. 10. Compared with Torralba's method [55], our approach improves the detection rate from 40 to 60 percent for Camera 1 and 63 to 82 percent for Camera 2, both at the false-positive rate of one per frame. The tree-based method [60] yielded consistently inferior performance to both [55] and our approach.

With our approach, most of the misdetections are due to small object scales and occlusions. False positives happen in textured regions, e.g., along the highway guard rail.

7.4 Vehicle Tracking and View Angle Estimation

In this experiment, we measure the vehicle orientation estimation accuracy in tracking. For evaluation, eight test vehicle sequences were downloaded from Google video, and available from [64]. The test sequences are of low frame rate—about 5-10 frames per second, with a pixel resolution of 320×240. These sequences exhibit strong motion blur artifacts and fast changes in object scale. There are eight distinct vehicles in the sequences, and each vehicle has at least 90 degrees view angle change. Most of the vehicles run on dirt roads and three of them are race cars. The vehicles in the test sequences are annotated with view angles at 5 degrees apart by having a user compare vehicles in the video sequences with images of a synthetic car model rotated at different angles.

In this experiment, our multiplicative kernel detector is the same detector trained for the city scenario in the previous section. However, the original view angle partition is too coarse for view angle tracking. Thus, we use the mode-finding approaches in Section 5 and annotate the 280 detector modes (cluster centers) with angles that are quantized to 5-degree steps. Note, in total there are 866 vehicle training examples, but,with our approach, only fewer than one-third must be annotated for the purpose of angle estimation.

We apply the tracking process explained in Section 6 on this data set. In this test, we assume that there is at most one vehicle in a sequence. Thus, an extensive search with detectors is triggered at the first frame and then triggered again when the tracker loses the target. The view angle estimate in a no-detection frame is linearly interpolated from previous and later view angle estimates. The number of particles is 3,000 in this experiment.

Example tracking results in four test sequences are shown in Fig. 11. The resulting angle estimation accuracy during tracking is summarized in Table 1. The median of absolute error is in the range from 5 to 15 degrees for the eight test sequences. The mean of absolute errors is in the range from 7.62 to 26.29 degrees. Two main causes of errors are motion blur and view angles that are not covered in the training examples. Although the tracker may lose the target due to these two reasons, the detectors can recover the target location and view angle automatically in later frames when observations are better presented. The tracking speed is about 2 seconds per frame, including the HOG feature extraction and detector evaluation, which were both implemented in Matlab without optimization. The HOG feature extraction takes about 70 percent of the CPU time.

7.5 Multiview Face Angle Estimation and Tracking

Multiview face detection is challenging due to the variation of face appearances at different view angles, in addition to other variations due to changes in illumination and facial expressions. A commonly used approach to detecting

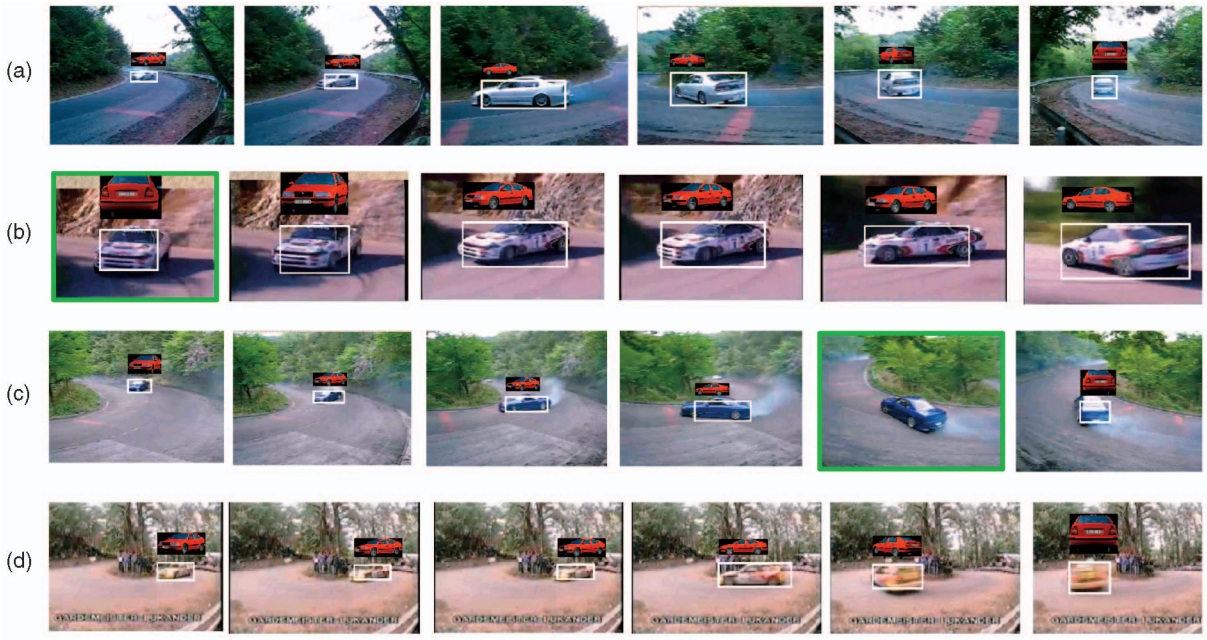


Fig. 11. Four example sequences of car tracking. Sequences (a), (b), (c), and (d) correspond to sequence IDs 3, 8, 1, and 7, respectively, in Table 1. Synthesized views of tracked cars are displayed on the top of a car. Green boxes highlight the errors in these sequences. In sequence (b), the initial detection in the first frame assigns the detected car a rear view, due to the ambiguity between front view and rear view. The error is corrected at subsequent frames when more frames are evaluated during temporal propagation. In sequence (c), the car is missed at frame 25 because the viewpoint elevation is much higher than those in training images.

multiview faces is to divide the view space into partitions and train a different detector for each partition. In previous work [19], multiview face detection is achieved via partitioning the face class into subclasses according to face view angles. In [38], a face manifold is learned by encoding the face view angles to detect multiview faces. Both approaches, however, require a huge amount of training images (30,000 in [38] and 75,000 in [19]). Manual annotation of such a large amount of data is expensive and both face training sets in [19], [38] are not publicly available. In contrast, our multiview face detectors can be trained with much fewer training examples because of implicit feature sharing.

In this experiment, we train and test our approach with a subset of the recently released CMU Multi-PIE data set [17]. The complete Multi-PIE data set contains face images from 337 subjects, imaged under 15 viewpoints and 19 illumination conditions in up to four recording sessions. We use a subset of 13 views and 10 illuminations of the first 32 subjects. In total, there are 8,320 face images in the subset. The 13 viewpoints are 30 degrees apart. Face regions are manually annotated by us. Background training samples are collected from 1,000 background images, containing indoor and outdoor images.

Our multiplicative kernel detector is trained with a nonparametric RBF kernel k_θ. The RBF is defined over the euclidean distance of HOG feature vectors, with $\eta = 0.1$. For comparison, subclass detectors for 13 view angle subclasses are trained by Torralba's feature sharing method [55], with 2,000 boosting iterations.

We evaluate the performance of face view angle estimation by four-fold cross-validation on 32 subjects. That is, every time we train on 24 subjects and test on the remaining eight subjects. Mean absolute error (MAE) is used as an evaluation metric for angle estimation accuracy. The comparison result is shown in Fig. 12. The overall MAE of our approach is 2.1 degrees, in contrast to 3.0 degrees of Torralba's feature sharing method. With our approach, 0.2 percent of the test samples have errors greater than or equal to 15 degrees. In contrast, 0.6 percent of the test samples have errors greater than or equal to 15 degrees with Torralba's feature sharing method.

To demonstrate our tracking approach in this setting, we collected two video sequences with multiple human subjects in a lab environment. There are 117 frames in the first sequence and 179 frames in the second sequence. The frame size is 480×360 pixels for the first sequence and 648×488 pixels for the second sequence. In each sequence, there are up

TABLE 1
View Angle Estimation Accuracy in Tracking Vehicles in Eight Different Test Sequences

Seq. ID	1	2	3	4	5	6	7	8
# frames	32	31	39	64	40	16	59	43
Mean-AE	24.38	26.29	15.71	17.10	7.62	11.25	18.73	19.42
Median-AE	10	10	15	15	5	10	10	15

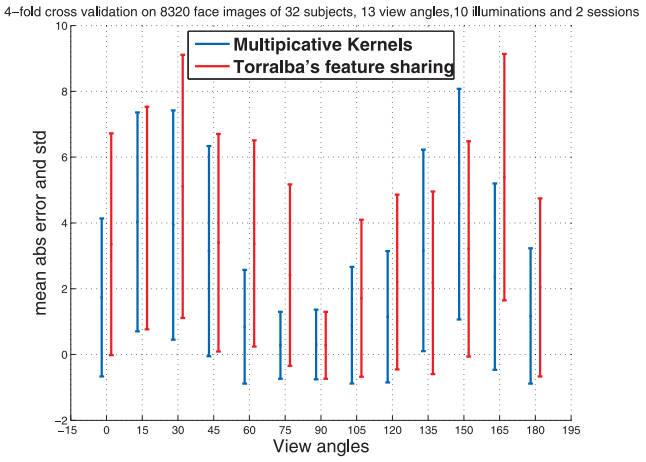


Fig. 12. Face view angle estimation result on Multi-PIE data set. For each view angle subclass, we plot the mean and standard deviation of the errors on test samples. The overall mean absolute errors are 2.1 degrees and 3.0 degrees for our method and Torralba's feature sharing method [55], respectively.

to three faces in a frame. The faces make left-right out-of-plane rotations and slight in-plane rotations. For evaluation purposes, we manually annotated all face locations and their left-right rotation angles in every other frame of the test sequences. During annotation, the faces in the test sequences are compared with face images from Multi-PIE training set to find matching face rotation angles.

We apply the tracking algorithm of Section 6 with multiplicative detectors on the two sequences. The training set for detectors that are used in tracking is the 4,160 face images of first 32 subjects and the first five illumination variations in Multi-PIE data set. During tracking, the frames are rotated up to 15 degrees at 5-degree increments to compensate for in-plane rotations. The number of particles is 6,000 in our approach. The tracking process is fully automatic. An exhaustive search with all detectors is triggered at every five frames in the first sequence and every 10 frames in the second sequence to reset the tracker. The reset rate was determined to roughl matchy the entrance rate of faces. The number of faces is determined by exhaustive search. Example frames of the tracking result are shown in Fig. 13. Most faces are detected correctly when their angle is within the range $[-90, 90]$ degrees. Most of the missed detections are due to large in-plane orientations or yaw angles.

At a false positive rate of 0.1 false positives per frame, our method achieves a detection rate of 77 percent on the first sequence and 90 percent on the second sequence. This difference might be attributed to the fact that faces are rotated outside the range $[-90, 90]$ in pitch angle more frequently in the first sequence. The MAEs of view angle estimation on detected faces are 3.08 degrees on the first sequence and 3.68 degrees on the second sequence.

The online tracking speed is about 10 seconds per frame on the first sequence and 17 seconds per frame on the sequence sequence. Extensive search takes about 5 minutes per frame on the first sequence and 14 minutes per frame on the second sequence, using unoptimized Matlab code. About 74 percent of the total time is spent in HOG feature extraction.

8 Discussion and Future Work

An observation in our experiments is that knowledge of variations within the foreground class helps the detection task. Both partition-based approaches [55], [60], [37] and our method are doing better than a single foreground-background classifier. Compared with partition-based approaches, our method can model more detailed variations like continuous rotation angles and distances/similarities between individual training examples. At the same time, feature sharing is strong during joint optimization of multiplicative kernels. These two factors may explain why our model does better than previous approaches in detection tasks.

The proposed approach is efficient given the fact that it uses a whole bank of detectors. Furthermore, our formulation does not preclude the use of a multilevel cascade structure. One possible extension is to add early detection stages to reject trivial background patches quickly. The detection speed may be further improved by a quick online process to determine a small subset of detectors to apply for a given input.

Acknowledgments

This paper reports work that was supported in part by the US National Science Foundation (NSF) under grants IIS-0705749 and IIS-0713168.

Fig. 13. Example tracking result in two test sequences. The first row is from sequence 1. The second row is from sequence 2. On top of each tracked face, a training example with the same face orientation is displayed. The tracker stops tracking when the left-right rotation of a face is larger than 90 degrees from a frontal face. A face is missed in the fifth example frame of the first sequence and the third example frame of the second sequence.

REFERENCES

[1] A. Agarwal and B. Triggs, "3D Human Pose from Silhouettes by Relevance Vector Regression," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2004.

[2] M. Andriluka, S. Roth, and B. Schiele, "People-Tracking-by-Detection and People-Detection-by-Tracking," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[3] V. Athitsos and S. Sclaroff, "Estimating 3D Hand Pose from a Cluttered Image," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2003.

[4] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 4, pp. 509-522, Apr. 2002.

[5] A. Bissacco, M. Yang, and S. Soatto, "Detecting Humans via Their Pose," *Advances in Neural Information Processing Systems,* MIT Press, 2006.

[6] A. Bissacco, M. Yang, and S. Soatto, "Fast Human Pose Estimation Using Appearance and Motion via Multi-Dimensional Boosting Regression," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.

[7] M.B. Blaschko and C.H. Lampert, "Learning to Localize Objects with Structured Output Regression," *Proc. European Conf. Computer Vision,* 2008.

[8] E. Borenstein and S. Ullman, "Class-Specific, Top-Down Segmentation," *Proc. European Conf. Computer Vision,* 2002.

[9] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning,* vol. 20, pp. 273-297, 1995.

[10] O. Crasborn, E. van der Kooij, A. Nonhebel, and W. Emmerik, "ECHO Data Set for Sign Language of the Netherlands," technical report, Dept. of Linguistics, Univ. of Nijmegen, Netherlands, 2004.

[11] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[12] T. Damoulas and M.A. Girolami, "Pattern Recognition with a Bayesian Kernel Combination Machine," *Pattern Recognition Letters,* vol. 30, no. 1, pp. 46-54, 2008.

[13] M. Enzweiler and D.M. Gavrila, "A Mixed Generative-Discriminative Framework for Pedestrian Classification," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[14] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 32, no. 9, pp. 1627-1645, Sept. 2010.

[15] P.F. Felzenszwalb and D.P. Huttenlocher, "Pictorial Structures for Object Recognition," *Int'l J. Computer Vision,* vol. 61, pp. 55-79, 2005.

[16] D.M. Gavrila, "Pedestrian Detection from a Moving Vehicle," *Proc. European Conf. Computer Vision,* 2000.

[17] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-PIE," *Proc. IEEE Int'l Conf. Face and Gesture Recognition,* 2008.

[18] D. Hoiem, A.A. Efros, and M. Hebert, "Putting Objects in Perspective," *Int'l J. Computer Vision,* vol. 80, no. 1, pp. 3-15, 2008.

[19] C. Huang, H. Ai, Y. Li, and S. Lao, "High-Performance Rotation Invariant Multiview Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 4, pp. 671-686, Apr. 2007.

[20] C. Ioffe and D. Forsyth, "Probabilistic Methods for Finding People," *Int'l J. Computer Vision,* vol. 43, no. 1, pp. 45-68, 2001.

[21] C. Ionescu, L. Bo, and C. Sminchisescu, "Structural SVM for Visual Localization and Continuous State Estimation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2009.

[22] M. Isard and A. Blake, "CONDENSATION: Conditional Density Propagation for Visual Tracking," *Int'l J. Computer Vision,* vol. 29, no. 1, pp. 5-28, 1998.

[23] T. Joachims, "Making Large-Scale SVM Learning Practical," *Advances in Kernel Methods—Support Vector Learning,* B. Scholkopf, C. Burges, and A. Smola, eds., MIT Press, 1999.

[24] M.P. Kumar, P.H.S. Torr, and A. Zisserman, "Obj Cut," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[25] B. Leibe, N. Cornelis, K. Cornelis, and L.V. Gool, "Dynamic 3D Scene Analysis from a Moving Vehicle," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.

[26] B. Leibe, A. Leonardis, and B. Schiele, "Robust Object Detection with Interleaved Categorization and Segmentation," *Int'l J. Computer Vision,* vol. 77, no. 1, pp. 259-289, 2007.

[27] S. Li, Q. Fu, L. Gu, B. Scholkopf, Y. Cheng, and H. Zhang, "Kernel Machine Based Learning for Multi-View Face Detection and Pose Estimation," *Proc. IEEE Int'l Conf. Computer Vision,* 2001.

[28] S. Li and Z. Zhang, "Floatboost Learning and Statistical Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 9, pp. 1112-1123, Sept. 2004.

[29] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, "Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Life Spans," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 10, pp. 1728-1740, Oct. 2008.

[30] M. Everingham et al. "The 2005 PASCAL Visual Object Class Challenge," *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment,* Springer, 2006.

[31] M. Marszalek, C. Schmid, H. Harzallah, and J. van de Weijer, "Learning Object Representations for Visual Object Class Recognition," *Proc. Visual Recognition Challange Workshop, in Conjunction with IEEE Int'l Conf. Computer Vision,* 2007.

[32] H. Murase and S.K. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance," *Int'l J. Computer Vision,* vol. 14, no. 1, pp. 5-24, 1995.

[33] C. Neidle, "SLLRP Signstream Databases," Boston Univ., http://ling.bu.edu/asllrpdata/queryPages, 2003.

[34] J. Nocedal and S.J. Wright, *Numerical Optimization.* Springer-Verlag, 2006.

[35] A. Oikonomopoulos, I. Patras, and M. Pantic, "Kernel-Based Recognition of Human Actions Using Spatiotemporal Salient Points," *Proc. Workshop Vision for Human Computer Interaction,* 2006.

[36] K. Okuma, A. Taleghani, N.D. Freitas, J. Little, and D. Lowe, "A Boosted Particle Filter: Multitarget Detection and Tracking," *Proc. European Conf. Computer Vision,* 2004.

[37] E. Ong and R. Bowden, "A Boosted Classifier Tree for Hand Shape Detection," *Proc. IEEE Int'l Conf. Face and Gesture Recognition,* 2004.

[38] R. Osadchy, M. Miller, and Y. LeCun, "Synergistic Face Detection and Pose Estimation with Energy-Based Model," *Advances in Neural Information Processing Systems,* MIT Press, 2004.

[39] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *Int'l J. Computer Vision,* vol. 38, no. 1 pp. 15-33, 2000.

[40] A. Pentland, B. Moghaddam, and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 1994.

[41] J. Platt, "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods," *Advances in Large Margin Classifiers,* A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, eds., MIT Press, 1999.

[42] D. Ramanan, D.A. Forsyth, and A. Zisserman, "Strike a Pose: Tracking People by Finding Stylized Poses," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[43] R. Rifkin and A. Klautau, "In Defense of One-vs-All Classification," *J. Machine Learning Research,* pp. 101-141, 2004.

[44] R. Rosales and S. Sclaroff, "Learning Body Pose via Specialized Maps," *Advances in Neural Information Processing Systems,* MIT Press, 2002.

[45] S. Roweis and L. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 5500, pp. 2323-2326, 2000.

[46] H.A. Rowley, S. Baluja, and T. Kanade, "Neural Network-Based Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 1, pp. 23-38, Jan. 1998.

[47] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman,, "LabelMe: A Database and Web-Based Tool for Image Annotation," technical report, Massachusetts Inst. of Technology, 2005.

[48] E. Seemann, B. Leibe, and B. Schiele, "Multi-Aspect Detection of Articulated Objects," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[49] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast Pose Estimation with Parameter-Sensitive Hashing," *Proc. IEEE Int'l Conf. Computer Vision,* 2003.

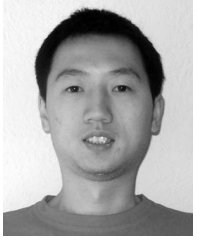
[50] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 1997.

[51] H. Sidenbladh, M.J. Black, and D.J. Fleet, "Stochastic Tracking of 3D Human Figures Using 2D Image Motion," *Proc. European Conf. Computer Vision,* 2000.

[52] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard, "Tracking Loose-Limbed People," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2004.

[53] C. Sminchisescu, A. Kanaujia, and D. Metaxas, "Learning Joint Top-Down and Bottom-Up Processes for 3D Visual Inference," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[54] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla, "Filtering Using a Tree-Based Estimator," *Proc. IEEE Int'l Conf. Computer Vision,* 2003.
[55] A. Torralba, K. Murphy, and W. Freeman, "Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2004.
[56] M. Varma and D. Ray, "Learning the Discriminative Power-Invariance Trade-Off," *Proc. IEEE Int'l Conf. Computer Vision,* 2007.
[57] P. Viola and M. Jones, "Fast Multi-View Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2003.
[58] P. Viola and M. Jones, "Robust Real Time Object Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.
[59] L. Wang, J. Shi, G. Song, and I. Shen, "Object Detection Combining Recognition and Segmentation," *Proc. Asian Conf. Computer Vision,* 2007.
[60] B. Wu and R. Nevatia, "Cluster Boosted Tree Classifier for Multi-View Multi-Pose Object Detection," *Proc. IEEE Int'l Conf. Computer Vision,* 2007.
[61] B. Wu and R. Nevatia, "Simultaneous Object Detection and Segmentation by Boosting Local Shape Feature Based Classifier," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.
[62] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff, "Parameter Sensitive Detectors," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.
[63] L. Zhu, Y. Chen, C. Lin, and A.L. Yuille, "Rapid Inference on a Novel and/or Graph: Detection, Segmentation and Parsing of Articulated Deformable Objects in Cluttered Backgrounds," *Advances in Neural Information Processing Systems,* MIT Press, 2007.
[64] http://www.cs.bu.edu/groups/ivc/data/MultiplicativeKernels/ 2010.



Quan Yuan received the BS and MS degrees in computer science and engineering from the Harbin Institute of Technology in 2001 and 2003, respectively, and the PhD degree in computer science from Boston University in 2009. He joined the Sony US Research Center in July 2009. His research interests include computer vision and machine learning. His recent work has focused on object recognition, gesture recognition, and medical image analysis. He is a member of the IEEE.

Ashwin Thangali received the BE degree from the National Institute of Technology, Surathkal, and the ME degree in electrical and computer engineering from the Indian Institute of Science, Bangaluru. He is currently working toward the PhD degree in computer science at Boston University as a member of the Image and Video Computing Group. He was a member of the research staff at the IBM Research Lab, New Delhi, where his work focused on content-based image and video retrieval with relevance feedback. His recent research interests are robust image similarity measures and linguistic constraints to aid handshape recognition in sign language video. He is a student member of the IEEE.

Vitaly Ablavsky received the BA degree in mathematics with departmental honors from Brandeis University, and the MS degree in computer science from the University of Massachusetts Amherst. He is currently working toward the PhD degree in computer science at Boston University. He was a software/research engineer at Amerinex Applied Imaging, Inc., Cognex Corporation, and Charles River Analytics, Inc. His research interests are object detection and tracking. He is a student member of the IEEE and the IEEE Computer Society.

Stan Sclaroff received the PhD degree from the Massachusetts Institute of Technology in 1995. He is currently a professor of computer science and the chair of the Department of Computer Science at Boston University. He founded the Image and Video Computing Research Group at Boston University in 1995. In 1996, he received a US Office of Naval Research (ONR) Young Investigator Award and a US National Science Foundation (NSF) Faculty Early Career Development Award. Since then, he has coauthored numerous scholarly publications in the areas of tracking, video-based analysis of human motion and gesture, deformable shape matching and recognition, as well as image/video database indexing, retrieval, and data mining methods. He has served on the technical program committees of more than 90 computer vision conferences and workshops. He has served as an associate editor for the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000-2004 and 2006-present. He is a senior member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.