Multiplicative Kernels: Object Detection, Segmentation and Pose Estimation

Quan Yuan, Ashwin Thangali, Vitaly Ablavsky, and Stan Sclaroff*
Computer Science Department, Boston University,USA

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. One kernel measures similarity for foreground-background classification. The other kernel accounts for latent factors that control within-class variation and implicitly enables feature sharing among foreground training samples. Detector training can be accomplished via standard SVM learning. The resulting detectors are tuned to specific variations in the foreground class. They also serve to evaluate hypotheses of the foreground state. When the foreground parameters are provided in training, the detectors can also produce parameter estimate. When the foreground object masks are provided in training, the detectors can also produce object segmentation. The advantages of our method over past methods are demonstrated on data sets of human hands and vehicles.

1. Introduction

For object classes that exhibit large within-class variations, detection (segmentation) and pose (parameter) estimation can be chicken-egg problems. Assuming the object is detected and segmented from background, pose estimation can be done as in [1, 28]. Assuming specific variations of the object class, detection (segmentation) can be achieved as in [23]. However, if both pose and detection (segmentation) are unknown, then challenges arise.

One common solution is to use a divide-and-conquer strategy, where the space of possible within-class variations is partitioned, and different detectors are trained for different partitions. Recent work in multi-view face detection builds detectors for different head pose ranges [12, 17, 33]. In pedestrian and human hand detection, sets or hierarchies of detectors tuned to different object variations are proposed to handle large appearance variations [10, 20, 27, 31, 35]. However, discrete partitions have limited power when there are too few training samples in each subclass. Explicit feature sharing has to be employed to boost the performance, but it also makes training more expensive.

In such a context, hybrid methods that unify detection and parameter estimation are of great interest, especially for human body detection and pose recognition. Some approaches combine bottom-up part-based detectors with top-down geometric constraints [9, 13, 21, 29]. Other approaches employ a recognition-verification strategy, where discriminative models are used to produce estimates of body pose (bottom-up), and then generative models are used to verify pose estimates (top-down) [16, 24, 30]. However, bottom-up recognition from images with background clutter remains difficult, and the verification step cannot correct an error when the recognition is already wrong.

Our work is related to recent work [37] in which a set of detectors is learned jointly. However, that work did not exploit the link to kernels. In our work, one kernel measures within-class similarity of the foreground class and the other one handles foreground-background classification. Nonlinear approximation is achieved with a nonlinear within-class kernel. The resulting detectors are tuned to specific variations of the foreground class. In the experiments, our detectors give improved detection accuracy over [32, 35, 37] on data sets of human hands and vehicles.

The main contributions are as follows. First, the multiplicative kernel formulation enables feature sharing implicitly; the solution for the optimal sharing is a byproduct of SVM learning. This is in contrast with [32, 37] where finding the optimum is intractable, and greedy strategies must be employed. Second, when object masks are included with the training data, detectors can produce a mask for segmentation of the foreground object. This mask also helps to reduce the influence of background clutter inside a detection window. Third, for applications where parameterization of foreground within-class variations is unavailable, a nonparametric form is proposed that can be used within the same multiplicative kernel model. This is in contrast with [37] where a separate Adaboost-based mechanism had to be employed. Finally, the proposed mode finding approach for the nonparametric case substantially reduces the amount of training data that must be labelled.

[1] This research was funded in part by NSF grants 0713168 and 0705749.

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

2. Our Approach

We start our derivation with the same problem definition as [37]. Given a feature vector $\mathbf{x} \in \mathbb{R}^n$ computed for an image patch[2], our goal is to decide whether or not the corresponding 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 poses, view angles, or latent factors that can be obtained via unsupervised learning. We aim to learn a function $C(\mathbf{x}, \boldsymbol{\theta})$ that tells whether $\mathbf{x}$ is an instance of the object with parameters $\boldsymbol{\theta}$,

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

The function $C(\mathbf{x}, \boldsymbol{\theta})$ is different from a generative model $P(\mathbf{x}, \boldsymbol{\theta})$ which has been explored in different contexts. Instead of estimating the distribution, $C(\mathbf{x}, \boldsymbol{\theta})$ only makes a binary decision as in Eq. (1). The magnitude of $C(\mathbf{x}, \boldsymbol{\theta})$ can be interpreted as confidence of the decision.

2.1. Multiplicative Kernel Construction

We now depart from [37] to introduce a nonlinear form of $C(\mathbf{x}, \boldsymbol{\theta})$, which leads naturally to a kernel representation. Assume $C(\mathbf{x}, \boldsymbol{\theta})$ can be factorized into the product of a feature space mapping $\boldsymbol{\phi}_x(\mathbf{x})$ and a weight vector $\mathbf{w}(\boldsymbol{\theta})$, which is a common form in a classification problem,

$$C(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\phi}_x(\mathbf{x})^T \mathbf{w}(\boldsymbol{\theta}) \tag{2}$$

where $\boldsymbol{\phi}_x(\mathbf{x}) = [\phi_x^0(\mathbf{x}), \phi_x^1(\mathbf{x}), \ldots, \phi_x^N(\mathbf{x})]^T$. Similarly $\mathbf{w}(\boldsymbol{\theta})$ is also in the form of basis function expansion, but is a combination of vectors,

$$\mathbf{w}(\boldsymbol{\theta}) = \sum_{i=0}^{M} \mathbf{v}_i \phi_\theta^i(\boldsymbol{\theta}) = \mathbf{V} \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \tag{3}$$

where vectors $\mathbf{v}_i \in \mathbb{R}^{N+1}$ are unknowns to be learned, and

$$\begin{aligned} \mathbf{V} &= [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_M] \\ \boldsymbol{\phi}_{\boldsymbol{\theta}}(\boldsymbol{\theta}) &= [\phi_\theta^0(\boldsymbol{\theta}), \phi_\theta^1(\boldsymbol{\theta}), \ldots, \phi_\theta^M(\boldsymbol{\theta})]^T. \end{aligned}$$

We assume that $\boldsymbol{\phi}_x$ and $\boldsymbol{\phi}_\theta$ are defined. If we plug Eq. 3 into Eq. 2,

$$\begin{aligned} C(\mathbf{x}, \boldsymbol{\theta}) &= \boldsymbol{\phi}_x(\mathbf{x})^T \mathbf{V} \boldsymbol{\phi}_\theta(\boldsymbol{\theta}) \\ &= \begin{bmatrix} \phi_\theta^0(\boldsymbol{\theta}) \boldsymbol{\phi}_x(\mathbf{x}) \\ \phi_\theta^1(\boldsymbol{\theta}) \boldsymbol{\phi}_x(\mathbf{x}) \\ \vdots \\ \phi_\theta^M(\boldsymbol{\theta}) \boldsymbol{\phi}_x(\mathbf{x}) \end{bmatrix}^T \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_M \end{bmatrix} \qquad (4) \\ &= \boldsymbol{\phi}_{\mathbf{x},\boldsymbol{\theta}}^T \mathbf{v}, \qquad (5) \end{aligned}$$

where $\boldsymbol{\phi}_{\mathbf{x},\boldsymbol{\theta}}$ and $\mathbf{v}$ are the left and right vectors respectively in Eq. 4. Note Eq. 5 is a standard binary classification problem, in which $\boldsymbol{\phi}_{\mathbf{x},\boldsymbol{\theta}}$ is the data term and $\mathbf{v}$ are unknown weights. It can be solved by a standard kernel based learning method, in our case, the SVM. The kernel function $k_c(\cdot, \cdot)$ is

$$\begin{aligned} k_c(\mathbf{y}, \mathbf{y}') &= \boldsymbol{\phi}_{\mathbf{x},\boldsymbol{\theta}}^T \boldsymbol{\phi}_{\mathbf{x}',\boldsymbol{\theta}'} \\ &= [\boldsymbol{\phi}_\theta(\boldsymbol{\theta})^T \boldsymbol{\phi}_\theta(\boldsymbol{\theta}')][\boldsymbol{\phi}_x(\mathbf{x})^T \boldsymbol{\phi}_x(\mathbf{x}')] \\ &= k_\theta(\boldsymbol{\theta}, \boldsymbol{\theta}') k_x(\mathbf{x}, \mathbf{x}'), \end{aligned} \tag{6}$$

where $\mathbf{y} = [\mathbf{x}^T, \boldsymbol{\theta}^T]^T$. Learning of $\mathbf{V}$ is implicit with this kernel representation. If learned by SVM, the classification function becomes,

$$\begin{aligned} C(\mathbf{x}, \boldsymbol{\theta}) &= \sum_{i \in SV} \alpha_i k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta}) k_x(\mathbf{x}_i, \mathbf{x}) \\ &= \sum_{i \in SV} \alpha_i'(\boldsymbol{\theta}) k_x(\mathbf{x}_i, \mathbf{x}), \end{aligned} \tag{7}$$

where α_i is the weight of the i^{th} support vector, and

$$\alpha_i'(\boldsymbol{\theta}) = \alpha_i k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta}). \tag{8}$$

Feature sharing among different $\boldsymbol{\theta}$ is implicitly achieved by sharing support vectors. When $k_\theta(\cdot, \cdot)$ is strictly nonnegative, *e.g.*, a radial basis function (RBF) kernel, Eq. 7 can be interpreted as re-weighting the support vectors so that only those having parameters similar to $\boldsymbol{\theta}$ keep high weights. Fewer support vectors have to be taken into account in a local subregion in $\boldsymbol{\theta}$ space, as shown in Fig. 1.

Once we obtain all support vectors and corresponding weights α_i after SVM learning, we are able to evaluate a tuple $(\mathbf{x}, \boldsymbol{\theta})$ as defined in Eq. 1. If we fix $\boldsymbol{\theta}$, $C(\cdot, \boldsymbol{\theta})$ is a detector for a specific $\boldsymbol{\theta}$. On the other hand, given a $\mathbf{x}$ from the foreground class, we can find the $\boldsymbol{\theta}$ that gives highest score via $C(\mathbf{x}, \boldsymbol{\theta})$ as the parameter estimate of $\mathbf{x}$.

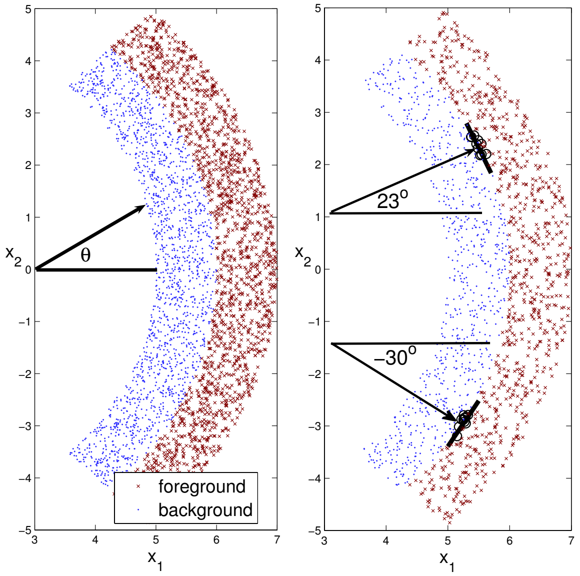
2.2. Nonparametric k_θ

In some problems, parametric forms of foreground within-class variations may not be readily available. For such cases, we propose a formulation that can utilize a nonparametric k_θ.

To understand the usage of nonparametric k_θ, we need to explain the role of parametric k_θ in feature sharing. When k_θ is defined on a continuous $\boldsymbol{\theta}$ space, two training samples with close $\boldsymbol{\theta}$ values 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. We aim to preserve this similarity behavior in designing a nonparametric kernel.

A straightforward design of a nonparametric kernel k_θ employs a nonparametric similarity/distance measure, *e.g.*,

[2] In this paper, all vector variables are column vectors.



(a) The original data (b) Two example detectors

Figure 1. An experiment on synthetic data. A family of multiplicative kernel classifiers is learned, where k_θ is a 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 $C(\mathbf{x}, 23^o)$ and $C(\mathbf{x}, -30^o)$ are shown on the right. The circle points are the reweighted support vectors (Eq. 8). These synthetic "foreground" and "background" classes were chosen to illustrate the idea that local discriminants can be learned jointly via multiplicative kernels, and then reconstructed at a given $\boldsymbol{\theta}$.

bidirectional chamfer edge distance [10, 2] or shape context distance [4]. Based on a distance measure D, a kernel function can be defined [19],

$$k_\theta(i, j) = exp\left(-\eta D(\mathbf{z}_i, \mathbf{z}_j)\right), \tag{9}$$

where $\mathbf{z}_i$ and $\mathbf{z}_j$ are representations of the foreground training samples indexed by i and j to calculate distance D.

After training, in contrast to obtaining a detector of a parameter $\boldsymbol{\theta}$ as in Eq. 7, we can obtain a detector for a particular training sample indexed by i:

$$\begin{aligned} C(\mathbf{x}, i) &= \sum_{j \in SV} \alpha_j k_\theta(i, j) k_x(\mathbf{x}_j, \mathbf{x}) \\ &= \sum_{j \in SV} \alpha'_j(i) k_x(\mathbf{x}_j, \mathbf{x}). \end{aligned} \tag{10}$$

The kernel $k_\theta(i, j)$ gives high weights to support vectors that are similar to i. Intuitively, those support vectors that can provide more evidence for i should be weighted higher.

It is possible that a distance measure does not guarantee a valid Mercer kernel, which should always have a positive semi-definite Gram Matrix. However, when the Gram Matrix based on k_θ is positive semi-definite for the training set, we can still apply it, since in detection only k_x is evaluated as in Eq. 7 and Eq. 10. If the Gram Matrix has small negative eigenvalues, we can either adjust η in Eq. 9, or employ a method to replace the negative eigenvalues with zeros [22].

Alternatively, we can use a prototype-based embedding method [3, 8, 11]. A set of prototype objects $z_1, \ldots, z_k$ can be selected from the foreground class. The embedding for each instance z is obtained by evaluating the distance D to the prototype objects:

$$\boldsymbol{\theta} = (D(z, z_1), \ldots, D(z, z_k)). \tag{11}$$

Since $\boldsymbol{\theta}$ is in a Euclidean space after embedding, we are guaranteed that k_θ in Eq. 9 is a valid Mercer kernel.

3. Detector Training

In this section, we give details on how to train the model defined in the previous section. A bootstrap training process is proposed first. Then, we will talk about how to incorporate image masks in training, to help reduce the influence of background clutter and to produce foreground object segmentation during detection.

3.1. Bootstrap Training

For training we are given a set of foreground and background training samples. The training samples take the form of tuples - $(\mathbf{x}, \boldsymbol{\theta})$ or $(\mathbf{x}, i)$. Each foreground sample $\mathbf{x}$ is associated with its corresponding groundtruth $\boldsymbol{\theta}$ (parametric case) or its sample index i (nonparametric case). A background training sample $\mathbf{x}$ can be associated with any foreground parameter or index of a foreground training sample to form a valid tuple. The number of such combinations can be huge. We therefore employ an iterative process of bootstrap training to avoid combinatorial complexity while maintaining the desired detection accuracy.

The training process starts with assigning each background feature vector $\mathbf{x}$ a foreground parameter $\boldsymbol{\theta}$ or index i at random. Then in each iteration false positives are collected. A false positive is also in the tuple form $(\mathbf{x}, \boldsymbol{\theta})$ or $(\mathbf{x}, i)$, where $\boldsymbol{\theta}$ or i is the corresponding parameter or sample index associated with the detector that accepts $\mathbf{x}$. A fixed subset uniformly sampled from the false positive tuples is added to the background training tuples for SVM training in the next iteration. The process ends when the number of false positives falls below a threshold or the maximum number of iterations has been reached. The pseudo code for the this process is shown in Fig. 2.

3.2. 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

```
Given
Foreground training samples $X_f = \{\mathbf{x}_{f_1}, \dots, \mathbf{x}_{f_n}\}$,
Background training samples $X_b = \{\mathbf{x}_{b_1}, \dots, \mathbf{x}_{b_n}\}$,
Parameters of foreground training samples $\Theta = \{\boldsymbol{\theta}_{f_1}, \dots, \boldsymbol{\theta}_{f_n}\}$.

  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 $\Theta$ to form tuples $(\mathbf{x}, \boldsymbol{\theta})$ for
      training. If object masks are available in foreground
      training data, the mask of $\boldsymbol{\theta}$ is applied on $\mathbf{x}$.

  While
          1. Compute Gram matrix $K_g$ of all training samples,
             where $k_g(i,j) = k_\theta(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) k_x(\mathbf{x}_i, \mathbf{x}_j)$.
          2. Carry out SVM training using $K_g$, and obtain individual
             detectors for each $\boldsymbol{\theta} \in \Theta$ according to Eq.7.
          3. Apply all the detectors to background training
             samples $X_b$. The false positive samples are in
             the form $(\mathbf{x}_{p_i}, \boldsymbol{\theta}_{p_i})$, where $\boldsymbol{\theta}_{p_i}$ is the parameter
             of the individual detector that accepts $\mathbf{x}_{p_i}$.
          4. If the number of $(\mathbf{x}_{p_i}, \boldsymbol{\theta}_{p_i}) <$ Threshold or the
             max number of iterations reached, break.
          5. Sample from all $(\mathbf{x}_{p_i}, \boldsymbol{\theta}_{p_i})$ to obtain a subset of
             a fixed size $N_s$. Expand original training tuples
             with this subset of false positive samples as background
             training tuples.

  end
```

Figure 2. Pseudo code 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.

be used to reduce the influence of background regions inside the detection window during both training and testing.

When each training sample has a 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 should be considered. When the features have local supports and are ordered according to their spatial arrangement, *e.g*., Histogram of Oriented Gradients (HOG) [7] or Haar wavelet features [34], applying object masks in feature extraction means that the feature components that have supports from outside the object masks have zero values.

To be consistent, background training samples also need to be applied with masks 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.

Once a detector is associated with a mask, segmentation can be produced by superimposing the detector's image mask onto an accepted image patch during detection. 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.

Masks do not have to be explicitly applied in testing when both of following two conditions are met:

1. The features have local supports like Histogram of Gradients (HOG) [7] and Haar wavelet features [34].
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 kernel and polynomial kernels.

This is true because during detection, when $\mathbf{x}_1$ is a support vector and $\mathbf{x}_2$ is the input feature vector, calculating $\mathbf{x}_1^T\mathbf{x}_2$ automatically zeros out the features of $\mathbf{x}_2$ from outside of $\mathbf{x}_1$'s mask. It has the same effect as excluding features of $\mathbf{x}_2$ from outside $\mathbf{x}_1$'s mask during detection.

We should also mention that object masks were also used in previous work [5, 36] where image segmentation and detection can be achieved jointly. However, in our method no decomposition of the image mask into local edgelets or image patches is needed, which reduces the effort in training.

4. Detector Set Acquisition and Mode Finding

In the detection process a set of detectors is applied on the input. An input is classified as from the foreground class if and only if at least one of the detectors accepts it, in the same manner as partition based methods [12, 17, 33, 35].

After training as described in Sec. 3, we are able to construct a detector for a parameter $\boldsymbol{\theta}$ or a foreground sample index i. The set of online detectors is sampled from all possible detectors. In the parametric case, a parameter sample set is uniformly sampled from the parameters of foreground training samples. The detectors of those sampled parameters form the online detector set. In our experiments, this works well, but other sampling methods could possibly be employed.

In the nonparametric case, first the detectors associated with each foreground training sample are constructed. Then via agglomerative clustering among detectors, the medoids of clusters are collected as the online detector set. The similarity measure S_α is defined on the support vector weights,

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

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 Eq.10.

Each cluster is regarded as a mode that represents a variation of the foreground class. During agglomerative clustering, the number of modes decreases as agglomerative clustering goes further. The proper number of modes is decided as the minimum number of modes that can achieve acceptable detection accuracy on a validation set.

Interestingly, in the nonparametric case, foreground states can still be recovered during detection by labelling the mode medoids (which are foreground training samples associated with online detectors). For example, in a pedestrian detection application, if the user wants to know whether a detected person is in a crouching pose, a walking pose or a standing pose, then the user can label mode medoids with these three labels. Once a pedestrian is detected, the label of the mode medoid associated with the detector of the highest score indicates the pose of this person. Parameter estimation can also be achieved by labelling the ground truth parameters of mode medoids. An obvious advantage of this strategy is that only a small portion of the foreground training data must be labelled. This can save a significant amount of effort that might be needed to label all training samples for the same purpose. We demonstrate the advantages of this approach in a vehicle view angle estimation experiment in the next section.

5. Experiments

In this section we evaluate the proposed method in two applications: hand detection and vehicle detection. For the purpose of these experiments, HOG [7] 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, and human hands and vehicles show strong edges in their appearances. The detectors of our method are trained by a modified version of SVMlight [14] with multiplicative kernels. The between-class kernel k_x is always a linear kernel, and the within-class kernel k_θ is parametric or nonparametric depending on the data set. In training both k_x and k_θ have to be calculated. In detection k_θ does not have to be calculated since the online detectors are sampled as a pre-computation as described earlier, and therefore only depend on $\mathbf{x}$. Our results are compared with results obtained via a partition based method [35] and methods that explicitly share features [37, 32].

5.1. Hand Shape Detection with Parametric k_θ

In the hand shape data set of [37], two angle parameters (within the range [0,90]) are labelled on all hand images. There are 1696 hand images for training and 1018 for testing. There are also 5500 background training samples and 50000 background test samples, cropped from real background images or hand images of other hand shapes. In the implementation of our method, the within-class kernel k_θ is a RBF kernel in the two-dimensional parameter space. After SVM training, 200 parameter values $\boldsymbol{\theta}$ with corresponding detectors are uniformly sampled from the 1696 parameter values associated with foreground training examples. These 200 detectors are then used at the online stage.

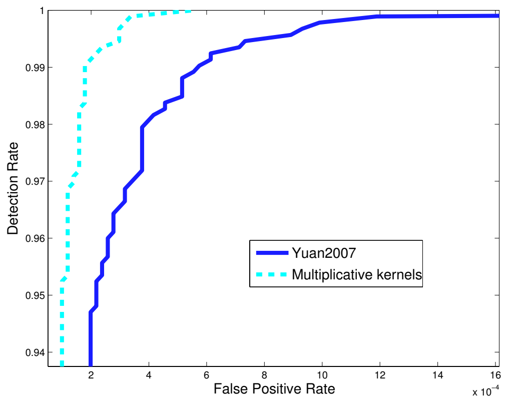


Figure 3. Comparison of ROC curves on the hand shape data set with two-dimensional parameters [37]. The detection rate is in the range between 94% and 100%.

The ROC curves of the detection result (hand vs. background) are shown in Fig. 3. As can be seen from the ROC curves, our method consistently outperforms [37]. At a false positive rate of 2×10^{-4}, it improves the true positive rate from 94% to 99%. Both methods apply 200 detectors at the online stage so the online speeds are very close. However, the training of the multiplicative kernel based method is about 10 times faster than the boosting based method [37].

Parameter estimation is achieved by assigning the parameter associated with the detector of the highest score. The mean absolute errors on the two finger parameters are 6.7 and 4.6 degrees respectively, in contrast to 9.0 and 5.3 degrees in [37].

5.2. Hand Detection and Segmentation with Nonparametric k_θ

In this data set the hand images are obtained without explicit parameter labelling. Hand images are collected from two sources of sign language video sequences: Flemish Sign Language data [6] and American Sign Language data [18]. Example frames are shown in Fig. 4.

Figure 4. Example sign language sequences from which the training and test hand images are obtained.

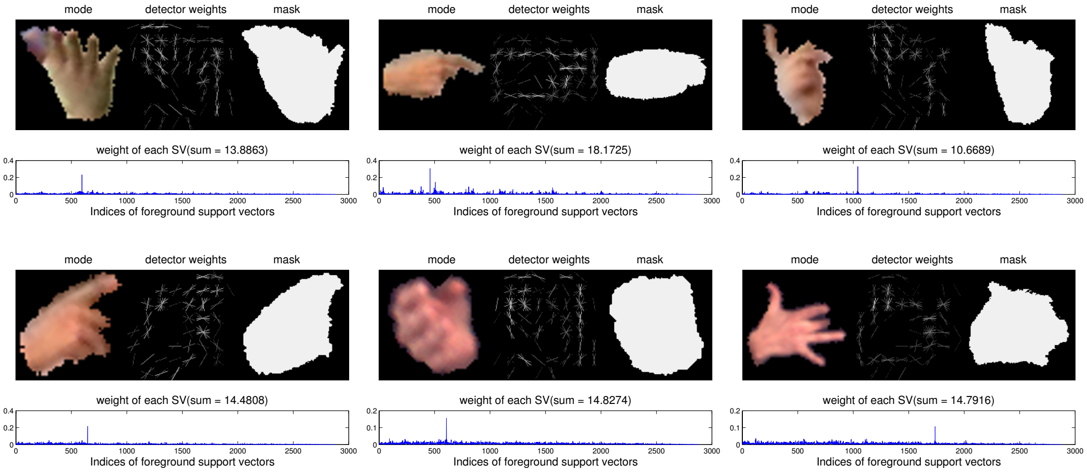


Figure 5. Six hand modes are displayed with their mode medoids, positive detector weights and mode masks. For each mode the weights of foreground support vectors are displayed at bottom.

In total there are 17 signers. The training set contains 3000 hand images and the test set has 2270 hand images. The test set and training set are disjoint. For the training hand images, corresponding hand silhouettes are also obtained. About 70% of the hand silhouettes are automatically segmented by skin color models or simple background models. The rest are obtained manually.

The background training set contains images of outdoor and indoor scenes. These are separated into disjoint training and test sets, which contain 300 images each. From each background image set 5000 image patches are collected as samples for the background class.

In the implementation of our method, the within-class kernel k_θ is in the nonparametric form of Eq. 9. The distance measure D is the bidirectional chamfer edge distance [10] between hand images. With $\eta = 1$, the Gram matrix of k_θ is positive semi-definite on the training set. 1242 hand modes are obtained after agglomerative clustering as described in Sec. 4. The total training time is about 30 minutes on a single 2.6GHz AMD Opteron 852 processor.

Six out of the 1242 hand modes obtained after training are shown in Fig. 5. For each mode the figure shows three images: the mode medoid, the positive weights of the detector associated with the medoid, and the mask. The positive weights of a mode detector are shown in the same way as in [7] to demonstrate how local edge orientations are weighted. A mode image mask 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 obtained masks have appropriate sizes and shapes for this setting. The resulting masks are applied at the online stage to yield segmentation results.

For each mode shown in Fig. 5, 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. Intuitively, this broad feature sharing ability may help to prevent overfitting on training sets that are of limited sizes.

Examples of the combined detection and segmentation results obtained with our method are shown in Fig. 6. The segmentation obtained in this way is only approximate; nonetheless, the shapes are matched well and the segmentation is obtained with no 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 as initialization to a method that requires segmented input.

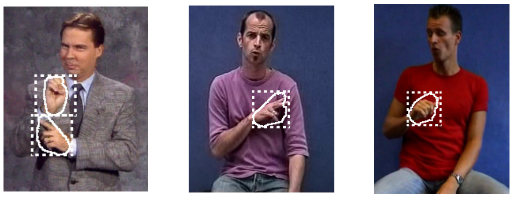


Figure 6. Examples of detection + segmentation.

For experimental comparison, a partitioning-based method is formulated and trained as follows: first clustering of hand subclasses is obtained via k-means with Euclidean distance of HOG features, then the detector for each subclass is trained using SVM with a RBF kernel. The γ 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

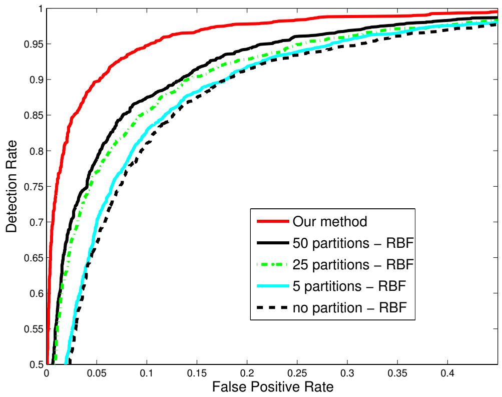


Figure 7. ROC curves of different detectors for hand detection.

are ignored during training and testing of the subclass detector. The accuracy of the partition based method improves as the number of partitions increases to around 50 partitions. Further increases of the number of partitions do not yield significant improvement.

The detection accuracy of the different methods is shown in the ROC curves of Fig. 7. Our method outperforms other methods by a clear margin on this data set. Compared to the best competing method (50 partitions), our method improves detection accuracy from 80% to 90% at a false positive rate of 5%. At the detection rate of 80% our method reduces the false positive rate from 5.3% to 1.7%.

5.3. Vehicle Detection and View Angle Category Classification

The vehicle detection data set consists of 2809 vehicle images from the LabelMe data set [25]. The vehicle images are put into 12 view angle categories as shown in Fig. 8. The background data set contains 778 street scene images without any vehicles. Both the vehicle and the background image sets are split half-half into disjoint training and test sets by random sampling. 5000 background training samples and 5000 background test samples are obtained from corresponding image sets.

In our method we use vehicle bounding boxes as masks. The nonparametric within-class kernel k_θ is a RBF kernel on HOG features with $\gamma = 0.5$. After training, 560 mode detectors obtained via agglomerative clustering are used for detection. We compare performance with three other detection methods [32, 37, 35]. For [32] the view angle labels are provided since it is a multi-class detection method. In [35] the tree structure is mainly controlled by a splitting threshold θ_z. The best θ_z is found at 0.95 in this experiment, which produces a tree of 13 leaf nodes.

The detection accuracy of all methods is shown in the ROC curves of Fig. 9. Compared with [35], our method improves the detection rate from 92.4% to 94.2% at the false positive rate of 1%. At the detection rate of 95%, our method reduces the false positive rate from 2.4% to 1.2%.

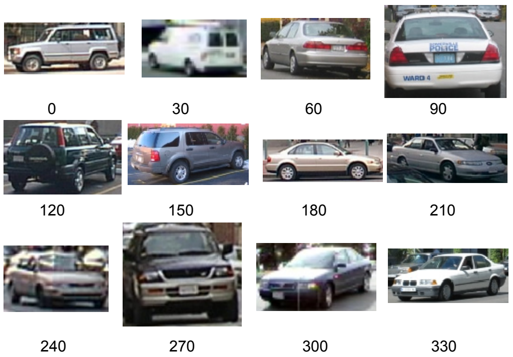


Figure 8. Examples of view angle variations of vehicles in the LabelMe data set [25].

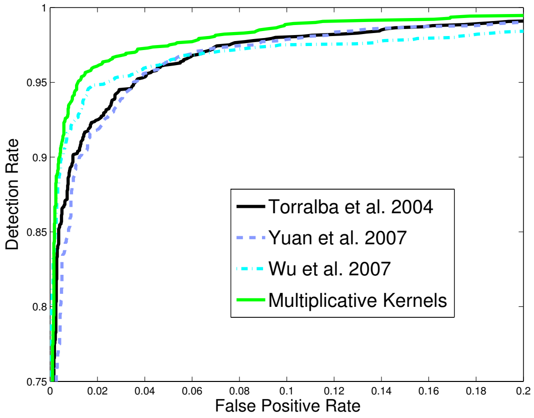


Figure 9. ROC curves for vehicle detection experiment.

Besides detection, we are also interested in estimating the view angle category of an input. The view angle category of a test input can be estimated if the mode medoids associated with detectors are labelled with true view categories. As described in Sec. 4, a foreground test example is assigned the view category of the mode medoids associated with the detector that has the highest score.

We compare our method with [32] and a baseline method on a view angle category estimation experiment. Each training and test vehicle example has a ground truth view angle category label, as is shown in Fig. 8. The method of [32] gives a fixed accuracy of 65.0% on this data set. A baseline method that learns 12 category detectors via Adaboost [26] gives an accuracy of 62.4%. The accuracy is relatively low, mainly due to 180 degree symmetry in cars. In the baseline method, each of the 12 category detectors is a combination of 2000 weak classifiers. In both [32] and the baseline method, view angle labels of all 1405 foreground training examples are given.

Our method achieves an accuracy of 65.6% with 400 modes, which means only 400 view angle category labels (out of 1405) are needed during training. With 600 and

800 modes, our method achieves accuracy of 68.0% and 71.7% respectively. Compared with [32] and the baseline method, a clear advantage of our method is that parameter labelling is only required on a portion of foreground training data to achieve comparable or better estimation accuracy on this vehicle data set.

6. Conclusion and Future Work

A multiplicative kernel gives a general form for learning a family of detectors that are tuned to within-class variations of the foreground class. With this formulation, feature sharing is implicitly achieved. The learned detectors also evaluate the foreground state. Moreover, object masks can be employed to reduce the background influence inside a foreground bounding box.

Currently the kernel functions employed in this work are limited, in contrast with the numerous interesting kernels proposed in the machine learning literature in recent years. As our formulation is general, we are enthusiastic about investigating the use of other kernels in our method. At the same time, exploiting implicit parameterizations like GPLVM [15] is also a direction for future investigation.

References

[1] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *CVPR*, 2004.

[2] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. In *CVPR*, 2003.

[3] V. Athitsos and S. Sclaroff. Boostmap: A method for efficient approximate similarity rankings. In *CVPR*, 2004.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(24):509–522, 2002.

[5] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV*, 2002.

[6] 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, U. Nijmegen, Netherlands, 2004.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[8] C. Faloutsos and K. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *SIGMOD*, 1995.

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61:55–79, 2005.

[10] D. M. Gavrila. Pedestrian detection from a moving vehicle. In *ECCV*, 2000.

[11] G. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *PAMI*, 25(5):530 – 549, 2003.

[12] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *ICCV*, 2005.

[13] C. Ioffe and D. Forsyth. Probabilistic methods for finding people. *IJCV*, 43(1):45–68, 2001.

[14] T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[15] N. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *NIPS*, 2003.

[16] 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. In *ICCV*, 2001.

[17] S. Li, L. Zhu, Z. Zhang, A. Blake, and H. Shum. Statistical learning of multi-view face detection. In *ECCV*, 2002.

[18] C. Neidle. ASLLRP signstream databases, http://ling.bu.edu/asllrpdata/queryPages, Boston University, 2003.

[19] A. Oikonomopoulos, I. Patras, and M. Pantic. Kernel-based recognition of human actions using spatiotemporal salient points. In *Workshop on Vision for Human Computer Interaction*, 2006.

[20] E. Ong and R. Bowden. Detection and segmentation of hand shapes using boosted classifiers. In *FGR*, 2004.

[21] C. Papageorgiou and T. Poggio. A trainable system for object detection. *IJCV*, 38(1):15–33, 2000.

[22] E. Pekalska, P. Paclik, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *JMLR*, 2:175–211, 2001.

[23] D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose : Tracking people by finding stylized poses. In *CVPR*, 2005.

[24] R. Rosales and S. Sclaroff. Learning body pose via specialized maps. In *NIPS*, 2002.

[25] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. Technical report, MIT, 2005.

[26] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[27] E. Seemann, B. Leibe, and B. Schiele. Multi-aspect detection of articulated objects. In *CVPR*, 2006.

[28] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, 2003.

[29] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard. Tracking loose-limbed people. In *CVPR*, 2004.

[30] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning joint top-down and bottom-up processes for 3D visual inference. In *CVPR*, 2006.

[31] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *ICCV*, 2003.

[32] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004.

[33] P. Viola and M. Jones. Fast multi-view face detection. In *CVPR*, 2003.

[34] P. Viola and M. Jones. Robust real time object detection. *IJCV*, 57(2):137 – 154, 2004.

[35] B. Wu and R. Nevatia. Cluster boosted tree classifier for multi-view multi-pose object detection. In *ICCV*, 2007.

[36] B. Wu and R. Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. In *CVPR*, 2007.

[37] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff. Parameter sensitive detectors. In *CVPR*, 2007.