2007

Parameter Sensitive Detectors

Boston U. Computer Science Tech. Report No. BUCS-TR-2007-003, June 2007.
To appear in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

Parameter Sensitive Detectors

Quan Yuan, Ashwin Thangali, Vitaly Ablavsky, and Stan Sclaroff
Computer Science Department
Boston University
111 Cummington Street, Boston, MA 02215
{yq,tvashwin,ablavsky,sclaroff}@cs.bu.edu

Abstract

Object detection can be challenging when the object class exhibits large variations. One commonly-used strategy is to first partition the space of possible object variations and then train separate classifiers for each portion. However, with continuous spaces the partitions tend to be arbitrary since there are no natural boundaries (for example, consider the continuous range of human body poses). In this paper, a new formulation is proposed, where the detectors themselves are associated with continuous parameters, and reside in a parameterized function space. There are two advantages of this strategy. First, a-priori partitioning of the parameter space is not needed; the detectors themselves are in a parameterized space. Second, the underlying parameters for object variations can be learned from training data in an unsupervised manner. In profile face detection experiments, at a fixed false alarm number of 90, our method attains a detection rate of 75% vs. 70% for the method of Viola-Jones. In hand shape detection, at a false positive rate of 0.1%, our method achieves a detection rate of 99.5% vs. 98% for partition based methods. In pedestrian detection, our method reduces the miss detection rate by a factor of three at a false positive rate of 1%, compared with the method of Dalal-Triggs.

1. Introduction

There are two typical classification problems in object recognition tasks:

1. Determine whether or not an image chip depicts a view of the foreground object class. This is commonly called *detection*, for instance, face detection, pedestrian detection, vehicle detection, etc.

2. Given an image chip that depicts a view of the foreground class, determine the object parameters. This is commonly called *pose estimation* or *parameter estimation*, for instance, body pose estimation.

Solutions of these two problems are typically treated as separate stages in an overall system, e.g., [1, 9]. On the other hand, there has been recent work in multi-view face detection that unifies detection and pose estimation [3, 5, 6, 11]; as a result, knowledge of the variations in the foreground class can be used to improve the detection accuracy. In [3, 5] the space of face view angles is hierarchically partitioned into sub-classes; thus, the pose class label is linked to the classifier at the finest level in the hierarchy. Finally, in [6] an input is mapped to a parameterized face manifold for simultaneous detection and pose estimation.

For many objects like human hands, human bodies and vehicles, the appearance can be quite different when the view angle or parameter settings (e.g., joint angles of human body) vary. The advantage of providing different detectors for different parameter variations is two-fold. First, each individual detector can be simpler as the variation of the object appearance for a fixed parameter setting is much smaller. Second, the parameter estimate or pose class for the object can be obtained almost for free.

However, despite the demonstrated benefits of such a divide-and-conquer strategy, there are still two key problems that are not addressed in previous work: (a) The partitioning is arbitrary in a continuous space where there are no natural boundaries (for example, consider the continuous range of human body poses). (b) Labelling the parameters of the training samples can be tedious for some applications. For instance, there can be more than 20 degrees of freedom for the human body or human hands.

One observation we have is that the detectors tuned to different parameter settings are still correlated; in other words, they are likely to share features, especially for those whose parameters are close to each other. Therefore, we propose to learn the detectors by modelling them in a parameterized function space and learn them jointly. In our derivation, the classification functions of different parameter settings boil down to a single "meta" binary classification function that can be solved via standard learning methods like Support Vector Machines (SVMs) or Adaboost.

In our work the detectors themselves are associated with continuous parameters. Thus, partitioning the parameter space is unnecessary. Because each detector has a coordinate in the continuous parameter space, they can be used for parameter estimation. Furthermore, the individual detectors share features implicitly or explicitly and thus improve their performance. As has been shown in [10], with feature sharing the multi-class detectors achieve higher detection accuracy and keep the structure compact.

In our formulation the parameters can explicitly model the state, like joint angles or view angles for objects like the human body, faces, hands, vehicles, etc. They can also be intrinsic parameters obtained via dimensionality reduction methods like Principal Component Analysis (PCA), or Gaussian Process Latent Variable Models (GPLVMs) [4]. In the case of intrinsic parameters, the effort of parameter labelling is saved. Note in this case the detection result does not directly recover explicit parameters of the original objects. Nonetheless, association of classifiers with the intrinsic parameters still makes individual classifiers tuned to different variations of the object appearance and therefore improves detection accuracy. Furthermore, the technique can be extended to non-metric spaces where only a similarity measure exists.

Although multiple detectors of different parameters are applied in the classification process, the complexity can be reduced significantly via the cascade strategy used in the Viola-Jones face detector [12], as is demonstrated in our experiments. Experiments in profile face detection, hand shape detection, and pedestrian detection demonstrate the advantages of this new approach over past techniques.

2. Our Approach

Given a feature vector[1] $\mathbf{x} \in \mathbb{R}^m$ computed for an image patch, our goal is to decide whether or not the corresponding image patch depicts an instance of the object. Let the variations of the object class be parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$, e.g., object poses, view angles. We aim to learn a function $C(\mathbf{x}, \boldsymbol{\theta})$ which tells us whether $\mathbf{x}$ is an instance of the object with parameters $\boldsymbol{\theta}$,

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

We define

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = C(\mathbf{x}, \boldsymbol{\theta}),$$

where $f_{\boldsymbol{\theta}}(\cdot)$ has $\mathbf{x}$ as a variable but $\boldsymbol{\theta}$ is fixed. Intuitively, learning an $f_{\boldsymbol{\theta}}$ that can identify object instances for a specific parameter $\boldsymbol{\theta}$ is easier than learning a detector that works for all possible parameter variations. For example, in human hand detection, if the finger angles and view angles are fixed, then a simple linear classifier may suffice.

The learning of $C(\mathbf{x}, \boldsymbol{\theta})$ implies the learning of a family Ω of functions $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. The underlying mapping is from parameters $\boldsymbol{\theta}$ to this family of functions,

$$\mathbf{w} : \mathbb{R}^n \rightarrow \Omega.$$

We propose two alternative ways to learn the mapping. One is by SVM, and the other one is by Adaboost.

2.1. Learning the Mapping $\mathbf{w}(\boldsymbol{\theta})$ by SVM

Assuming Ω is a family of linear functions, i.e.

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = C(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}^T \mathbf{w}(\boldsymbol{\theta}) \tag{2}$$

where $\mathbf{w}(\boldsymbol{\theta})$ is a vector of weights in the linear classifier, we can use a linear function to approximate $\mathbf{w}(\boldsymbol{\theta})$ given training data. Suppose $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^T$. The linear approximation of $\mathbf{w}(\boldsymbol{\theta})$ is

$$\mathbf{w}(\boldsymbol{\theta}) \quad = \quad \sum_{i=1}^{n} \mathbf{v}_i \theta_i + \mathbf{v}_0 \tag{3}$$

where the vectors $\mathbf{v}_i \in \mathbb{R}^{m+1}$ are unknowns to be learned via supervised learning. If we plug Eq.(3) into Eq.(2), the classification function $f_{\boldsymbol{\theta}}$ of Eq. (2) becomes

$$\begin{aligned} f_{\boldsymbol{\theta}}(\mathbf{x}) \quad &= \quad \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}^T [\mathbf{v}_n \theta_n + \ldots + \mathbf{v}_1 \theta_1 + \mathbf{v}_0] \\ &= \quad \begin{bmatrix} 1 \\ \mathbf{x} \\ \theta_1 \\ \theta_1 \mathbf{x} \\ \vdots \\ \theta_n \\ \theta_n \mathbf{x} \end{bmatrix}^T \begin{bmatrix} \mathbf{v}_0 \\ \\ \mathbf{v}_1 \\ \\ \vdots \\ \mathbf{v}_n \\ \\ \end{bmatrix}. \end{aligned} \tag{4}$$

During training, the goal is to find $\mathbf{v}_0, \mathbf{v}_1 \ldots \mathbf{v}_n$, given a training set of positive and negative samples of the form $(\mathbf{x}, \boldsymbol{\theta})$. The optimality of the learned vectors $\mathbf{v}_i$ is measured by the classification performance of all the $f_{\boldsymbol{\theta}}$ over the instances of $\mathbf{x}$. In Eq. (4) the knowns (from data) and unknowns are separated into two vectors. The problem of learning the unknown $\mathbf{v}_0, \mathbf{v}_1 \ldots \mathbf{v}_n$ reduces to learning a binary linear classifier in this augmented feature space. Structural risk minimization strategies like those used in the SVM can be applied. Also note that if only $\mathbf{v}_0$ is used (i.e., $n = 0$), then Eq. (4) reduces to a linear classifier – without knowledge of $\boldsymbol{\theta}$. The vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ can be regarded as correction terms that adjust the classifier vector linearly with respect to $\boldsymbol{\theta}$. In training, each negative sample

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

is a background patch associated with a random parameter or a foreground patch with a parameter far from the true value.

During classification, we sample in the parameter space to obtain detectors. Each sample comes with a linear classifier $f_{\boldsymbol{\theta}}$. If any of these classifiers accepts the input $\mathbf{x}$, then $\mathbf{x}$ is classified as an instance of the object class, otherwise it is not. Since we have the knowledge of variations in the object class, sampling follows the prior distribution of $\boldsymbol{\theta}$ to make classification more efficient.

2.2. Learning by Adaboost

There is an alternative way to learn the mapping $\mathbf{w}(\boldsymbol{\theta})$ by Adaboost. The advantage of Adaboost training is that the augmented feature space is not needed because of explicit feature sharing. Moreover, it works in cases when parameters are not available.

We can denote a parameter-sensitive classifier in terms of a combination of weak classifiers:

$$C(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} h_1(\mathbf{x}) \\ \dots \\ h_M(\mathbf{x}) \end{bmatrix}^T \mathbf{w}(\boldsymbol{\theta}) = \mathbf{h}^T \mathbf{w}(\boldsymbol{\theta}), \tag{5}$$

where h_j is a weak classifier, usually a decision stump that involves a single feature and a threshold. For each weak classifier, we need to know: (a) what positive samples or range of parameters would benefit from this weak classifier for detection, and (b) what the weight of the weak classifier should be.

The weights $\mathbf{w}(\boldsymbol{\theta})$ are defined by weight vectors $\mathbf{v}_i$ and binary functions $\phi_i(\boldsymbol{\theta})$ which indicate what range of parameters are sharing features.

$$\mathbf{w}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mathbf{v}_k \phi_k(\boldsymbol{\theta}) = \mathbf{V}\boldsymbol{\phi}, \tag{6}$$

where $\mathbf{V} = [\mathbf{v}_1 | \mathbf{v}_2 | \dots | \mathbf{v}_K] \in \mathbb{R}^{M \times K}$ and $\boldsymbol{\phi} = [\phi_1(\boldsymbol{\theta}), \phi_2(\boldsymbol{\theta}), \dots, \phi_K(\boldsymbol{\theta})]^T$,

$$\phi_k(\boldsymbol{\theta}) = \begin{cases} 1, \boldsymbol{\theta} \in \Theta^k \\ 0, \ otherwise. \end{cases} \tag{7}$$

Θ^k is a region of the parameter space, for instance, defined as a hyper-rectangle or hyper-sphere. We call $\phi_k(\boldsymbol{\theta})$ the *support* of weak classifiers, and the size of ϕ_k is defined as the size of the region of Θ^k in the parameter space. If we plug Eq.(6) into Eq.(5), we have

$$C(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{h}^T \mathbf{V} \boldsymbol{\phi} = \sum_{j=1}^{M} h_j(\mathbf{x}) \sum_{k=1}^{K} v_{kj} \phi_k(\boldsymbol{\theta}). \tag{8}$$

where v_{kj} is the (k, j)-th entry of $\mathbf{V}$. The learning of v_{kj} is achieved via AdaBoost, but in a way that allows sharing of weak classifiers, similar as [10]. However, unlike [10], in our work features are shared among samples of continuous parameters instead of a number of discrete classes.

Given: $\Phi = \{\phi_k | \phi_k(\boldsymbol{\theta}) = 0 \text{ or } 1\}$ are the supports,
$H = \{h_j | h_j(\mathbf{x}) = \pm 1\}$ are the weak classifiers,
$X = \{(\mathbf{x}_i, \boldsymbol{\theta}_i, l_i, w_i)\}$ are training samples, where $\mathbf{x}_i$ is a feature vector, $\boldsymbol{\theta}_i$ is a parameter value, $l_i = \pm 1$ is the class label, and w_i is the training sample weight.

- Initialize:
  1. $w_i = 1/N$,
  2. $\{\phi_1, \phi_2, \dots, \phi_K\}$ are partitioned into disjoint subsets Φ_i, such that all ϕ_k in each Φ_i are of the same size. $\Phi_1, \Phi_2, \dots, \Phi_T$ are sorted in decreasing order of sizes of $\phi_k \in \Phi_i$.
- For $t = 1$ to T
  1. $X_t = \{(\mathbf{x}_i, \boldsymbol{\theta}_i, l_i, w_i) | \phi_k(\boldsymbol{\theta}_i) = 1, \phi_k \in \Phi_t\} \subset X$.
  2. Run AdaBoost [7] on X_t with H and Φ_t. In each boosting iteration, we select a $h_j \in H$ and a $\phi_k \in \Phi_t$ that achieves the most reduction of training sample weights. The classifier weight v_{kj} is assigned by Adaboost. The sum of w_i is normalized to 1 after each boosting iteration. The boosting iterations stop when the termination condition is met.

Figure 1. Algorithm for parameter-sensitive detector training.

In training, a pool of supports ϕ_k with different Θ^k is provided, where Θ^k are overlapping and $\bigcup \Theta^k$ covers the parameter range of all training samples. Our goal is to select the supports and weak classifiers to minimize the training error via Adaboost. The joint optimization over all features and all supports can be expensive for practical applications. Instead our algorithm employs a greedy strategy. In each boosting iteration, it selects weak classifiers always under supports of a fixed size, i.e., only one size of ϕ_k is considered in each boosting iteration, starting from the largest size. The algorithm optimizes with respect to training samples within a selected support; normalization in each boosting iteration keeps the total weight of training samples constant. Once the termination criterion is met (validation accuracy does not increase or margin does not improve), the algorithm moves to the subset of supports of the second largest size, and so on. The heuristic of selecting large support first encourages feature sharing, which in turn, leads to more parsimonious strong classifiers. We employ the discrete AdaBoost algorithm [7] with predefined feature thresholds, which leads to much faster training. The

training algorithm is summarized in Fig. 1.

The main difference between our method and training of a regular binary Adaboost classifier is that a weak classifier may be selected because it reduces weighted error of a subset of training samples, instead of all training samples being taken into account in every iteration. After training with our algorithm, for a parameter value $\hat{\boldsymbol{\theta}}$, all the selected tuples (h_j, ϕ_k) can be partitioned into two non-overlapping sets A and B,

$$A = \{(h_j, \phi_k) | \phi_k(\hat{\boldsymbol{\theta}}) = 1\}, \tag{9}$$

$$B = \{(h_j, \phi_k) | \phi_k(\hat{\boldsymbol{\theta}}) = 0\}. \tag{10}$$

All weak classifiers in A (with their weights v_{kj}) are combined into a strong classifier as the detector for the sample $\hat{\boldsymbol{\theta}}$. During classification, samples in the $\boldsymbol{\theta}$ space are generated and corresponding strong classifiers are composed. Given an input $\mathbf{x}$, if one of the strong classifiers accepts it, then the input is classified as an instance of the object class.

The parameter-sensitive detector learned by Adaboost is a piece-wise approximation of the parameterized function space. It is different from partition-based methods [3, 11] in that supports in the piece-wise space are learned via optimizing classification accuracy, instead of simply partitioning the foreground class using within-class similarities without a direct link to resulting classification accuracy.

2.3. Extension to Non-Metric Spaces

The definition of binary support functions ϕ_k can be easily extended to non-metric spaces. In practice, there are applications where continuous parameterization of training samples is difficult to obtain. In this case the partition based methods and parameter sensitive detector using numerical parameters cannot be applied. However, the variations among foreground objects can still be captured by similarity measures, e.g., the Chamfer edge distance in the hand data set of [1]. With small modifications, the Adaboost-trained parameter sensitive classifier can still be applied. Note in the previous section the parameters define which samples are going to share features in training. A similarity measure is enough to define the sharing among the training samples, if the support of features is defined

$$\phi_k(\mathbf{x}) = \begin{cases} 1, D(\mathbf{x}_k, \mathbf{x}) \leq T_k \\ 0, \ otherwise \end{cases} \tag{11}$$

where $\mathbf{x}_k$ and $\mathbf{x}$ are two samples, e.g., two hand images, D is a similarity measure, e.g. Chamfer edge distance, and T_k is a threshold. Given a sample $\mathbf{x}_k$ as a center, the function ϕ_k specifies that those $\mathbf{x}$ within a certain distance from $\mathbf{x}_k$ share weak classifiers. Note the domain of ϕ_k is now the training sample space, instead of the parameter space. The size of ϕ_k is defined as the number of positive training samples $\mathbf{x}$ that satisfies $D(\mathbf{x}_k, \mathbf{x}) \leq T_k$. During training, each negative sample is associated with an index of positive samples, instead of a numerical parameter. During classification, the detectors are sampled from indices of the positive samples.

3. Implementation and Experiments

We demonstrate the parameter sensitive classifiers in three detection problems. The detection accuracies are compared with previous methods [2, 6, 11].

3.1. Profile Face Detection

Profile face detection tends to be more difficult than frontal face detection, due to the variations of head poses and fewer discriminative features. Detection rates reported in the literature are lower than that for frontal face detection [6, 11]. Our positive training samples and features are the same as [11]. The test set is CMU-PROFILE data set [8] which has 208 images with 353 profile faces.

Pose parameters are not provided with the training data. Instead, we learn a PCA space of the intensities from positive training samples. The intensity variations of face images are normalized to reduce illumination effects before applying PCA. The first two PCA components account for 70% of the variance and are used as parameters for the parameter sensitive classifier, i.e., $\boldsymbol{\theta} \in \mathbb{R}^2$.

A nine-level cascade detector is constructed via bootstrap, in a similar manner to [11]. The first eight levels are standard Adaboost classifiers which reject trivial background patches. The last level is trained by linear SVM parameter-sensitive classifier (Sec. 2.1) with 3300 binary features selected from sixth to eighth levels. The feature values are corresponding weak classifiers outputs, which are 1 or -1. The negative training samples are the false positive training samples passing through the first eight levels. During testing, a sample is first projected onto the PCA space to get the two PCA coordinates. No sampling in the parameter space is needed.

Table 1. Correct detection rates at two false alarms levels on the CMU profile face data set.

# False alarms	90	700
Our method	75%	86%
Viola-Jones[11]	70%	83%
Osadchy *et al.* [6]	67%	83%
Schneiderman-Kanade [8]	86%	93%

The comparison result is shown in Table. 1. The parameter-sensitive detector improves the detection rate over Viola-Jones detector by 5% and 3% at 90 and 700 false alarms respectively. The detection rate of [8] is the best among the four methods but with a large penalty in speed, as is reported in [11].

3.2. Detecting Hands with Bending Index Finger

In the second experiment, a parameter sensitive detector trained by Adaboost is used to detect a hand shape class that has the index finger extended, with variations of index finger angles and in-plane orientations as shown in Fig. 2. There are two degrees of freedom in the hand shapes: in-plane orientation, and angle of the index finger with respect to the palm. Each angle is within the range [0,90] degrees. In total 3394 real hand images from several subjects are labelled and separated into positive training (70%) and testing samples(30%). Negative training samples are cropped from real background images or hand images of other hand shapes, as shown in Fig. 3. There are 5500 negative training samples and 50000 negative testing samples.

Histogram of Gradients (HOG) [2] features are employed. The hand window is of size 48 by 48 pixels, which is divided into 64 cells of size 6 by 6. For each cell, nine edge orientation bins are evenly spaced between 0 to 180 degrees ("unsigned" gradient). The edges are detected by a Sobel edge detector, and each pixel votes for its orientation bin by edge magnitude. Bins in each cell are normalized with the surrounding 3 by 3 cells using the 2-norm as in [2]. There are 576 features extracted from each image patch.

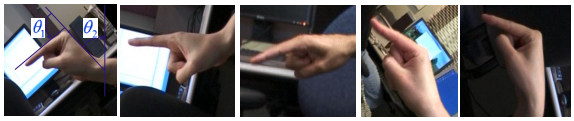

Figure 2. The hand shape with two degrees of freedom.

Figure 3. Example negative training and testing image patches.

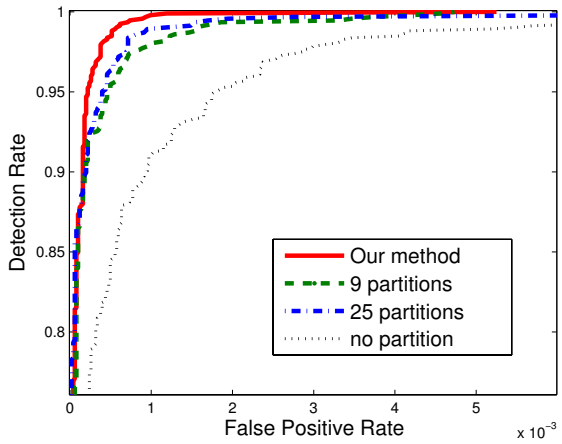

Figure 4. The ROC curves of parameter-sensitive detectors, regular Adaboost detector and partition-based detectors.

There are 600 supports for training of the parameter-sensitive detector. The smallest support covers a 15 degree by 15 degree region in the parameter space of (θ_1, θ_2). The detection accuracy is compared with that of an Adaboost classifier with no partition, and partition-based methods where each parameter dimension is evenly divided into three and five partitions. As shown in the ROC curves of Fig. 4, the parameter-sensitive detector outperforms the best comparing method by 1.5% at a false positive rate of 0.1%. Among the 4490 weak classifiers combined in the parameter-sensitive detector, 945 of them have supports that cover more than 80% of all the positive training samples, and 1830 of them have supports that cover fewer than 50% of all the positive training samples. Clearly there exists strong feature sharing among different parameters, yet the classifiers with smaller supports make a difference.

A parameter estimator is learned in a similar way as the detector. We should note that following Eq. (1) and Eq. (5) detection and parameter estimation can be accomplished in one step. In practice, we found it advantageous to employ a two level cascade. The first level of our cascade is optimized for detection (foreground vs background classification) by selecting negative training patches from background images. The second level is optimized for parameter estimation by including hand shape images with random parameter values as negative training samples. Feature sharing between different hand shapes decreases in the second level. The cascade structure however yields better performance with little extra computation as described below.

During training of the parameter estimator, the positive and negative samples are the same set of hand images as in Fig. 2. Negative samples are assigned random parameters that differ by ≥ 15 degrees from the true values. In total, AdaBoost selects 227 weak classifiers for the parameter estimator. During classification, for a given test image, all positive parameter samples in training data are used to generate their corresponding classifiers. The parameter estimate is given by the classifier with the maximum score for the test input. In our experiments with the cascade, the mean absolute errors (MAE) for θ_1 and θ_2 (as in Fig. 2) are 9.045 and 5.303 degrees respectively. This is an improvement compared to the MAE of 10.946 and 6.793 degrees we obtain using the parameter values corresponding to the highest scoring detectors from first level of the cascade.

3.3. Pedestrian Detection

In the third experiment, we demonstrate the approach in a pedestrian detection application, using the data set of [2]. There are no explicit parameters given for the positive samples. Substantial background regions included in positive training samples make it difficult to learn a PCA space of foreground appearance variations; nonetheless, histogram distances can still be used to measure the similarity between

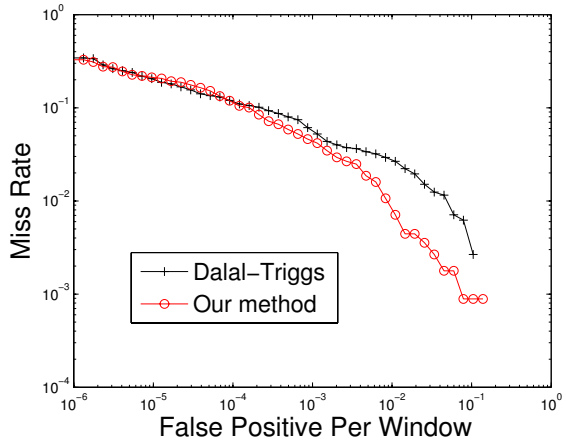

Figure 5. Comparing our method with Dalal-Triggs[2] method.

Figure 6. The matching between testing and training samples. For each pair, on the left is a test image, on the right is the training image corresponding to the detector of the highest score.

positive training samples. We train the parameter sensitive classifiers by Adaboost using the extension to non-metric spaces of Sec. 2.3. The similarity measure D is the Mahalonobis distance with diagonalized covariance matrix. In training, 300 supports ϕ_k are centered at 50 positive training samples. The centers are selected sequentially from all foreground training samples after removing the closest samples around previous centers. The sizes of supports are between 50 and 2416, which is the total number of foreground training samples.

Our detector is a two-level cascade. The first level is a regular binary classifier trained by Adaboost with 1000 weak classifiers using HOG features. The second level is a parameter sensitive detector trained by Adaboost (Sec. 2.3), which has 6000 weak classifiers in total. 200 individual detectors are sampled from the second level and used during classification. We implement the method of [2] and our implementation matches their result. Another method [13] reports an accuracy comparable to [2] but with a much faster speed. The comparison between our method and [2] is shown in Fig. 5. Compared with [2], our method reduces the miss detection rate from 2.7% to 0.8% at a false positive rate of 1%, and reduces the false positive rate from 5% to 0.9% at a miss detection rate of 1%. The speed of our method is 1.3 times slower than [2] over the test set.

During testing, the detector with the highest score corresponds to the training sample that can be interpreted as a "matching" sample for the test input. Twelve matching pairs are displayed in Fig. 6. For each pair, on the left is a test image, on the right is the training sample corresponding to the detector of the highest score. As can be seen, there is similarity between the images in each pair.

4. Conclusion and Future Work

We developed a framework for detection of objects with large within-class variations. The proposed approach learns a parameterized function space of detectors. Each sample in the function space corresponds to a detector for a particular variation of the object class. When explicit parameters are not available, the framework can make use of intrinsic parameters via unsupervised learning. Furthermore, the method extends to non-metric spaces.

In future work, parameter-sensitive detectors can be developed into hierarchical detectors by combining weak classifiers of the same support into one component, which may further improve classification speed.

References

[1] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR*, pages 432 – 439, 2003.

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

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

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

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

[6] R. Osadchy, M. Miller, and Y. LeCun. Synergistic face detection and pose estimation with energy-based model. In *NIPS*, 2004.

[7] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *ICML*, 1997.

[8] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *CVPR*, 2000.

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

[10] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, pages 1441–1448, 2004.

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

[12] P. Viola and M. Jones. Robust real time object detection. In *IJCV*, volume 57, pages 137 – 154, 2004.

[13] Q. Zhu, S. Avidan, M. Yeh1, and K. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, 2006.