Optimizing Discrimination-Efficiency Tradeoff in Integrating Heterogeneous Local Features for Object Detection

Bo Wu and Ram Nevatia
University of Southern California
Institute for Robotics and Intelligent Systems
Los Angeles, CA 90089-0273
$\{bowu|nevatia\}@usc.edu$

Abstract

A large variety of image features has been invented for detection of objects of a known class. We propose a framework to optimize the discrimination-efficiency tradeoff in integrating multiple, heterogeneous features for object detection. Cascade structured detectors are learned by boosting local feature based weak classifiers. Each weak classifier corresponds to a local image region, from which several different types of features are extracted. The weak classifier makes predictions by examining the features one by one; this classifier goes to the next feature only when the prediction from the already examined features is not confident enough. The order in which the features are evaluated is determined based on their computational cost normalized classification powers. We apply our approach to two object classes, pedestrians and cars. The experimental results show that our approach outperforms the state-of-the-art methods.

1. Introduction

Detection of objects of a class, such as humans or cars, is a fundamental problem of computer vision. It is difficult because the object appearance may vary due to many factors, including viewpoint, occlusion, illumination, texture, and articulation. This has motivated invention of different image features that capture different characteristic properties. Some existing methods for object detection base their detectors on a single type of feature. This enables a direct comparison of the detection performance of different features. Some others try to integrate multiple feature types to improve performance. Intuitively, more information should result in a better decision.

There are two main issues in feature integration. First, evaluating all the features before making a prediction is not efficient, because some features could be computationally expensive but not bring a significant boost in classification power. Second, different types of features could lie in different spaces, linear or nonlinear, which may require different classification techniques. For example, some features may lie on a nonlinear manifold embedded in a linear space. Directly applying the traditional classification techniques based on Euclidean distance to such a feature space is not appropriate as two points close in the linear space may be far from each other on the manifold. Hence, direct Cartesian product of different types of features before classification is not always feasible.

In this paper, we propose a novel method for integration of heterogeneous features for object detection. Our approach balances two criteria: accuracy and efficiency. It has better accuracy than the single-feature based methods and yet maintains a relatively fast speed.

1.1. Related work

The problem of object detection has been worked on since the beginning of computer vision research. A large variety of image features has been developed. Some are spatially global features, *e.g.* the edge template [24, 6], but most recent methods use local features, because local features are less sensitive to occlusions and other types of partial observation missing. Some examples are the wavelet descriptor [25], the Haar like feature [23], the sparse rectangle feature [12], the SIFT like orientation feature [17], the Histogram of Oriented Gradients (HOG) descriptor [14], the code-book of local appearance [15, 18], the edgelet feature [16], the boundary fragment [9], the biologically-motivated sparse, localized feature [7], the shapelet feature [4], the covariance descriptor [5], the motion enhanced Haar feature [20], and the Internal Motion Histogram (IMH) [10]. These features have been applied successfully to detection of several object classes, including faces [25, 23, 12, 27], pedestrians [24, 6, 17, 14, 15, 16, 4, 5, 20, 10], and cars [25, 18, 7, 2].

After the features or descriptors are computed, they are fed into a classifier. The classifier could be an SVM [14, 27], a boosted cascade [23], or based on a graphical

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

model [3, 11, 9, 15]. For the graphical models, different types of features are naturally integrated by the observation model of each node, like in [11]. However, estimating the joint probability distribution in a high dimensional feature space is not feasible. In practice, it is usually assumed that the different feature types are independent, so the joint probability is equal to the multiplication of the probabilities of individual feature types. For SVM classifiers, concatenated feature vectors are commonly used as input, but this is feasible only when the features are homogeneous, as the combination of two histogram features (HOG and IMH) in [10]. Linear combination of multiple non-linear kernels, each of which is based on one feature type, is a more general way to integrate heterogeneous features, *e.g.* [1]. However, both the vector concatenation and kernel combination based methods require evaluation of all features. For cascade classifiers, different types of features can be included in one big feature pool from which an ensemble classifier is learned by a boosting algorithm, as in [20]. However, if there are big differences of computational complexity between different feature types, the speed of the cascade classifier learned in this way will be dominated by the most complex feature type.

1.2. Outline of our approach

We choose the cascade structured classifier [23] for our classifier model, as it has proven to have both high accuracy and fast speed for several detection tasks [12, 16]. In the previous cascade classifier methods, each weak classifier corresponds to one image feature. In our approach each weak classifier corresponds to one sub-region of the image and different types of features are extracted from the sub-region, see Fig. 1 for an illustration. The classification function for each type of feature is learned in its own feature space. The multi-feature weak classifier makes a prediction by examining the different types of features from the sub-region one by one. Only when the prediction based on the already examined features is not of high confidence, does the weak classifier look at the next feature type. The order in which the features are evaluated is determined based on a measure of the classification power that includes the cost of computational time. A number of such weak classifiers are selected and combined by a boosting algorithm to form a cascade structured classifier.

The main advantages of our approach compared to the previous methods are: 1) the speed normalized classification power is used as the criterion for feature selection. Thus, the optimization goal is not only classification accuracy but also efficiency. 2) The classification functions of different types of features are learned in their own spaces, not in the Cartesian space, so that different classification techniques can be used to achieve better accuracy for different features. 3) Complex features, which may be more powerful, are evaluated only when necessary, *i.e.* when the decision can not be made confidently from the relatively cheap features. We use three types of features, the edgelet feature [16], the HOG descriptor [14], and the covariance descriptor [5], and two object classes, pedestrians and cars, to demonstrate and validate our approach. The experimental results show that our method achieves better accuracy with a relatively fast speed compared to the state-of-the-art single-feature based methods.

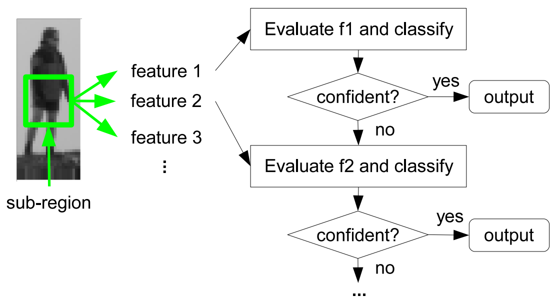


Figure 1. Schematic diagram of our feature integration method.

The rest of this paper is organized as follows: section 2 presents the general framework of our approach; section 3 gives the implementation details; section 4 shows the experimental results; and some conclusions and discussions are given in the last section.

2. Learning Algorithm

Our learning algorithm uses a boosting approach. A number of weak classifiers, each of which corresponds to one image region, are selected and combined to form a strong classifier. Assume that from one image region R, we can extract m types of local features, $\{f_1, \dots, f_m\}$. A feature f is a mapping from the image space $\mathcal{X}$ to a real valued d-dimensional space $\mathbb{R}^d$.

2.1. General weak classifier

Denote the weak classifier based on the image region R by $h_R : \mathcal{X} \to \mathbb{R}$. (The sign of h_R's output indicates the predicted class, $+$ for object and $-$ for non-object, and the magnitude represents the classification confidence.) If accuracy is the only objective, h_R should use all the features extracted from R for classification. However, for real applications, speed is another important criterion. We allow the weak classifier to use a variable subset of features to make the decision. Denote the power set of $\{f_1, \dots, f_m\}$ by $\mathcal{F}$, and a classifier based on a subset of features by h_F, $F \in \mathcal{F}$. We formalize the classifier by

$$h_R(\mathbf{x}) = h_{\Phi(\mathbf{x})}(\mathbf{x}) \tag{1}$$

where $\Phi : \mathcal{X} \to \mathcal{F}$ is a feature type selector. We define a *computational Cost Normalized classification Margin (CNM)* to measure the classification efficiencies of different subsets of features.

For a sample $\mathbf{x}$, denote its true class label by y $(= \pm 1)$. The classification margin of h on $\mathbf{x}$ is defined by $y \times h(\mathbf{x})$

(assuming h has been normalized to $[-1,1]$). The classification margin represents the discriminative power of the classifier. Larger margins imply lower generalization error [28]. The CNM of h on $\mathbf{x}$ is defined by

$$\dot{M}(h,\mathbf{x}) \triangleq \frac{yh(\mathbf{x})}{t(h)} \tag{2}$$

where $t(h)$ is the computational cost of h. We want to use the subset of features with the highest CNM measure:

$$\Phi^\star(\mathbf{x}) = \arg\max_{F\in\mathcal{F}} \left\{ \dot{M}(h_F,\mathbf{x}) \right\} \tag{3}$$

If h_F is approximated by a linear combination of several base classifiers $\sum_{f_i\in F} h_{f_i}$, each of which is based on one feature type, the computations of different features are independent, and the computational cost of Φ is ignorable compared to those of h_{f_i}, Equ.3 can be reduced to

$$\Phi^\star(\mathbf{x}) = \arg\max_{\{f_k\}\in\mathcal{F}} \left\{ \dot{M}(h_{f_k},\mathbf{x}) \right\} \tag{4}$$

The expected CNM of $h_{\Phi^\star}$ on $\mathcal{X}$ is

$$E\left(\dot{M}(h_{\Phi^\star})\right) = \sum\nolimits_k \alpha_k E\left(\dot{M}(h_{f_k})|\Phi^\star = \{f_k\}\right) \tag{5}$$

where $\alpha_k = P(\Phi^\star = \{f_k\})$.

2.2. Hierarchical weak classifier

In practice, it is unlikely to be able to compute $\Phi^\star$ before evaluating any features. (This requires choosing the best feature types before seeing the image.) In this work, we propose a hierarchical classification function to approximate $h_{\Phi^\star}$. The basic idea is to evaluate the features one by one, and after each evaluation decide whether it is necessary to look at more features. Assume we evaluate the features in the order $f'_1,\dots,f'_m$. We define $h_{\{f'_1..f'_m\}}$ in a recursive way

$$\begin{aligned} &h_{\{f'_1\}}(\mathbf{x}) = h_{f'_1}(\mathbf{x}) \\ &h_{\{f'_1..f'_k\}}(\mathbf{x}) = \begin{cases} h_{\{f'_1..f'_{k-1}\}}(\mathbf{x}), \text{if} \, \mathbf{x} \in V_{\{f'_1..f'_{k-1}\}}(\mathcal{X}) \\ h_{f'_k}(\mathbf{x}), \text{otherwise} \end{cases} \end{aligned} \tag{6}$$

where h_f is a single feature based weak classifier, whose details will be given later in section 2.3, and $V_{\{f'_1..f'_k\}}(\mathcal{X})$ is defined by

$$\begin{aligned} &V_{\{f'_1\}}(\mathcal{X}) = \left\{ \mathbf{x} | \left| h_{f'_1}(\mathbf{x}) \right| > \theta_1 \right\} \\ &V_{\{f'_1..f'_k\}}(\mathcal{X}) = V_{\{f'_1..f'_{k-1}\}}(\mathcal{X}) \cup \left\{ \mathbf{x} | \left| h_{f'_k}(\mathbf{x}) \right| > \theta_k \right\} \end{aligned} \tag{7}$$

where θ_k is a threshold of confidence. It is chosen adaptively for each feature

$$\theta_k = \arg\min_\theta P\left(\left| h_{f'_k}(\mathbf{x}) \right| > \theta \right) \le \frac{\alpha_k}{1-\sum_{i=1}^{k-1}\alpha_i} \tag{8}$$

$V_{\{f'_1..f'_k\}}$ represents the part of $\mathcal{X}$ where the prediction of $h_{\{f'_1..f'_k\}}$ is confident. Finally we have: if $\mathbf{x} \in V_{f'_k} = V_{\{f'_1..f'_k\}} - V_{\{f'_1..f'_{k-1}\}}$, then $\Phi = \{f'_1,\dots,f'_k\}$, and $h_\Phi = h_{\{f'_1..f'_k\}}$. The expected CNM of our hierarchical weak classifier is

$$E\left(\dot{M}(h_\Phi)\right) = \sum\nolimits_k \lambda_k \beta_k E\left(\dot{M}(h_{f'_k})|\mathbf{x}\in V_{f'_k}\right) \tag{9}$$

where $\lambda_k = \frac{t(h_{f'_k})}{\sum_{i=1}^k t(h_{f'_i})}$, and $\beta_k = P(\mathbf{x}\in V_{f'_k})$. If the feature types used are fully independent, β is equal to α.

To determine the order in which the features are evaluated, we sort the features according to their expected CNMs. The feature with higher expected CNM is evaluated earlier. This is an approximation of the optimal order. We have not found an algorithm that computes the optimal order in polynomial time w.r.t. the number of feature types. To rank the heterogeneous features, the classifiers should be defined in a comparable way. In our work, we use probability ratio based classifiers (details are given in the next section). For arbitrary classifier models, some normalization techniques, such as described in [22], should be applied before ranking.

2.3. Learning weak classifier

Following the real AdaBoost algorithm [26], we define the single-feature weak classifier h_f as a piecewise function based on a partition of the sample space $\mathcal{X}$ into disjoint subsets $\{X_j | X_j \subset \mathcal{X}\}_{j=1}^n$, which cover all of $\mathcal{X}$. For each subset of the partition, the output of h_f is defined by

$$\forall \mathbf{x}\in X_j, h_f(\mathbf{x}) = \frac{1}{2}\ln\left(\frac{W_+^j+\epsilon}{W_-^j+\epsilon}\right) \tag{10}$$

where ϵ is a smoothing factor [26], and $W_\pm$ is the object/non-object sample distribution on the partition $\{X_j\}$, *i.e.* $W_\pm^j = P\left(\mathbf{x}\in X_j \wedge y = \pm 1\right)$.

In our algorithm, for each feature f_k, first we find a projection p_k to map f_k to $[0,1)$. This projection separates the two classes as much as possible in the 1-D space. For different feature types, the projections can be different, either linear or non-linear. (Later in section 3, we give the implementation details of the projections for the features we use.) Then we do a uniform partition in the 1-D projection space:

$$X_{k,j} = \left\{ \mathbf{x}\in\mathcal{X} | p_k\left(f_k(\mathbf{x})\right) \in \left[\frac{j-1}{n},\frac{j}{n}\right) \right\} \tag{11}$$

and compute the classification function h_{f_k} by Equ.10. (In our experiments, we set $n = 32$.) Because the outputs of our classifiers are defined by probability ratio and the features are used only for partition, their margins are directly comparable.

One approximation in classification function learning is that the sample distributions W used to compute $h_{f'_k}$ are learned independently. For the hierarchical classification function in Equ.6, ideally we should learn the conditional probability distribution given that the samples lie in $\mathcal{X} - V_{\{f'_1..f'_{k-1}\}}$. However, this imposes an exponentially increasing demand of training data w.r.t. the number of feature types.

The hierarchical weak classifier corresponds to a hierarchical partition of the sample space. Fig.2 gives an illustration. Most of the sample space is divided only along the first dimension, while some difficult part is further divided along the second dimension, and so on. The classification power of the hierarchical weak classifier with multiple features is measured by its expected CNM in Equ.9. At each boosting round, we evaluate several sub-regions. For each of them, we find the best feature f_k for each feature type and combine them to form h_R. The h_R with the largest expected CNM is added to the current cascade classifier.

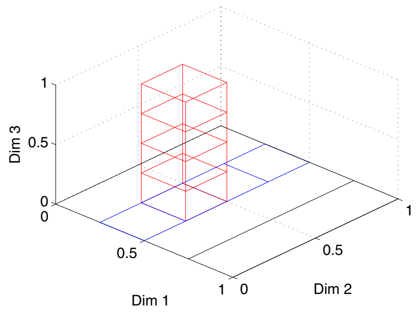


Figure 2. An illustration of hierarchical partition of sample space.

3. Implementation

The features we use are the edgelet feature [16], the HOG descriptor [14], and the covariance descriptor [5]. They are all state-of-the-art shape oriented features and have been applied to object detection problems successfully. There are many other candidates, however, these three types of features are sufficient to demonstrate the different aspects of our approach.

3.1. Feature dependent projection functions

For different feature types, we design different projection functions in order to get the best classification result. One edgelet feature can be seen as a short edge template. The feature response is the matching score between the template and the input image, *i.e.* $f_{edge} : \mathcal{X} \mapsto [0, 1)$. Hence, we just use the identity function as the projection function for edgelet, denoted by p_1.

For HOG descriptor, we do not use the dense sampling version in [14], instead we use the variable-sized block version in [8]. Given a rectangular sub-region, it is divided into 2×2 equal-sized cells. Within each cell, the edge intensities at nine orientations are summed. The output is a 36-D histogram vector, *i.e.* $f_{HOG} : \mathcal{X} \mapsto \mathbb{R}^{36}$. We use Linear Discriminative Analysis (LDA) to find a linear projection p_2 best separating the object and non-object classes:

$$p_2(f_{HOG}(\mathbf{x})) \triangleq a_2 \langle \mathbf{v}_2, f_{HOG}(\mathbf{x}) \rangle + b_2 \qquad (12)$$

where $\mathbf{v}_2 \in \mathbb{R}^{36}$, and a_2 and b_2 are normalizing factors learned from the training set.

Our covariance descriptor is extracted from a 6-D raw feature vector: $\begin{bmatrix} x & y & |I_x| & |I_y| & |I_{xx}| & |I_{yy}| \end{bmatrix}$, where x and y are the pixel location, and I_x, I_y, I_{xx}, I_{yy} are the first/second order intensity derivatives. The covariance matrices lie in a connected Riemannian manifold [21]. Formally $f_{cov} : \mathcal{X} \mapsto \mathcal{M}_{6\times6}$, where $\mathcal{M}_{6\times6}$ is a manifold embedded in $\mathbb{R}^{6\times6}$. Because the covariance matrix is symmetric, the real dimension is $6 \times (6+1)/2 = 21$. As the manifold is not a linear space, it is not appropriate to apply LDA directly. Following the method in [5], we first map the covariance matrices to a linear tangent space of the manifold, and then perform LDA in the tangent space. Denote the mapping to the tangent space by $\psi : \mathcal{M}_{6\times6} \mapsto \mathbb{R}^{21}$, whose definition is

$$\psi(\mathbf{X}) \triangleq \mathrm{vec}_\mu(\log_\mu(\mathbf{X})) \qquad (13)$$

where $\mathbf{X}$ and μ are two positive definite symmetric matrices, $\log_\mu$ is a matrix logarithm operator that maps a matrix in the manifold to a matrix in the tangent space attached to μ, and vec_μ is a coordinate mapping operator that converts the Riemannian metric on the tangent space to the canonical metric in the vector space. More details of ψ and its learning can be found in [5]. The projection function of the covariance descriptor is defined by

$$p_3(f_{cov}(\mathbf{x})) \triangleq a_3 \langle \mathbf{v}_3, \psi(f_{cov}(\mathbf{x})) \rangle + b_3 \qquad (14)$$

where $\mathbf{v}_3 \in \mathbb{R}^{21}$, and a_3 and b_3 are normalizing factors learned from the training set.

3.2. Computational costs of features

The computational costs of the three types of features are very different. Computing an edgelet response, which is basically edge template matching, requires mainly 16-bit integer operations; computing the histograms of HOG through integral images requires mainly 32-bit integer operations; computing a covariance matrix through integral images requires 32-bit integer and 64-bit floating point operations. The projection p_2 is an inner product operation of two floating point vectors; the complexity of p_3 is dominated by the matrix logarithm operator in ψ, which requires singular value decomposition (SVD). At first, we used the OpenCV SVD function and evaluated the speed of $p(f(\mathbf{x}))$ for the three features. The ratio of their average speeds is about $t_{edge} : t_{HOG} : t_{cov} = 1 : 10 : 30$. This order is consistent with those reported in the original papers [16, 14, 5]. In order to speedup, we replaced the OpenCV SVD with an implementation in the Intel IPP library [30]. This results in a speed ratio about 1:10:12. We have done this evaluation on several versions of Intel CPUs. The ratio is stable. Besides $p(f(\mathbf{x}))$, computing the edge intensity images of different orientations and the integral images brings some overhead. However, this overhead, which is partially shared among different types of features, is relatively small.

3.3. Selecting the best weak classifiers

Similar to [5, 8], we randomly sample 200 sub-regions at each boosting round, and search for the locally best edgelet, HOG, and covariance features. For each region R, the local search is done by randomly evaluating 40 edgelets, 5 HOG features, and 5 covariance features whose supporting regions $R(f)$ have large cover of R. For an edgelet feature, $R(f)$ is the bounding box of the edge template; for the HOG and covariance descriptors $R(f)$ is the rectangular region from which the histograms are computed. We sample more edgelet features than the other two types, because the edgelet feature pool is bigger than those of the other two.

During training, only for a part of the training set, the intermediate representation (including the edge intensity images and the integral images) is computed and stored in memory. At each boosting round, the samples with buffered intermediate representation are used to select the good features. After features are fixed, all the training samples are used to refine the classification functions.

4. Experimental Results

We apply our approach to two classes of objects, pedestrians and cars. These two classes are important for many applications, such as visual surveillance.

4.1. Performance on pedestrians

For pedestrians we use the INRIA data set [14][1], which contains 2,478 positive samples and 1,218 negative images for training, and 1,128 positive samples and 453 negative images for testing. The pedestrian sample size is 64 × 128 pixels. This set covers multiple viewpoints, and a large variety of poses.

For this set, we learn a cascade structured classifier that consists of 800 weak classifiers. Fig.3 shows the ROC curves of our method and some previous ones. (The ROC curve of our classifier is generated by changing the number of layers used.) From Fig.3, it can be seen that compared to the edgelet-only [2], HOG-only [8], and covariance-only [5] cascades, our hybrid-feature cascade achieves better performance. On average, our cascade classifier searches around 24,000 sub-windows per second on a 3.0GHz Intel CPU. Fig.8(a) shows some example results of pedestrian detection.

4.2. Feature statistics

Fig.4 shows the first weak classifier learned for pedestrian detection with its selected features and the corresponding classification functions. It can be seen that the covariance descriptor has the best discriminative power. However, due to its high computational cost, it is only the second feature of the weak classifier after an edgelet, before a HOG.

For the cascade pedestrian detector learned from the INRIA set, first we count the frequencies of different types of features selected as the first/second/third feature in the weak classifiers, shown in Fig.5. It can be seen that though HOG and covariance descriptors are stronger than edgelet for classification, they are much more computationally expensive so that they are used as the second or third feature most of the time.

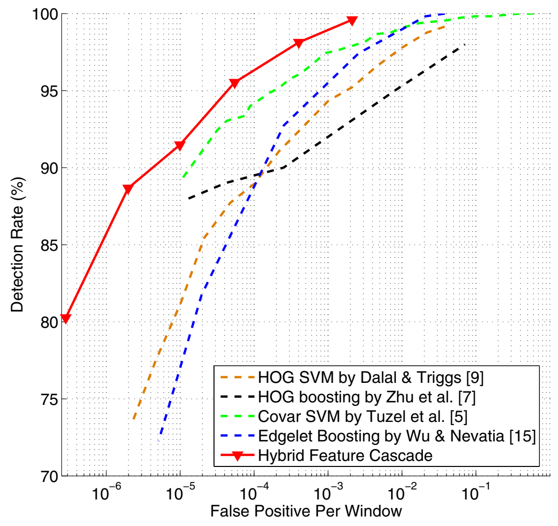


Figure 3. Pedestrian detection performance on the INRIA set. (For detection tasks, precision-recall curves are better to demonstrate the performance. However, in order to compare with previous methods, here we use detection rate and false positive per window for pedestrians.)

Next, we count the frequencies of different types of features that are evaluated per sub-window. This is a good hardware-independent metric to compare the speeds of different methods; Table 1 shows the results. Tuzel *et al.* [5] report that on average the HOG-only cascade requires evaluating 15.62 HOG features per sub-window and the covariance-only cascade needs 8.45 covariance descriptors per sub-window. For edgelets we do the evaluation ourselves, as there are no such results reported in the original paper. The edgelet-only cascade requires about 28 edgelets per sub-window. Based on the speed ratio of the three types of features, it can be seen that our hybrid-feature detector is faster than the HOG-only and covariance-only detectors, but slower than the edgelet-only detector.

Last, we count the evaluation frequencies for the first, second, and third features, shown in Table.2. It can be seen that the third features are rarely used. The first features are mostly edgelets, which are designed to encode the local silhouette explicitly but are relatively sensitive to small transformations, such as translation and rotation. The second and third features are mostly HOG and covariance descriptors, which encode the statistics of a sub-region and are robust to small transforms, but do not encode which pixels actually contribute to the histogram bins; very different shapes

[1]http://pascal.inrialpes.fr/data/human/

could result in the same histogram. Their complementarity is natural.

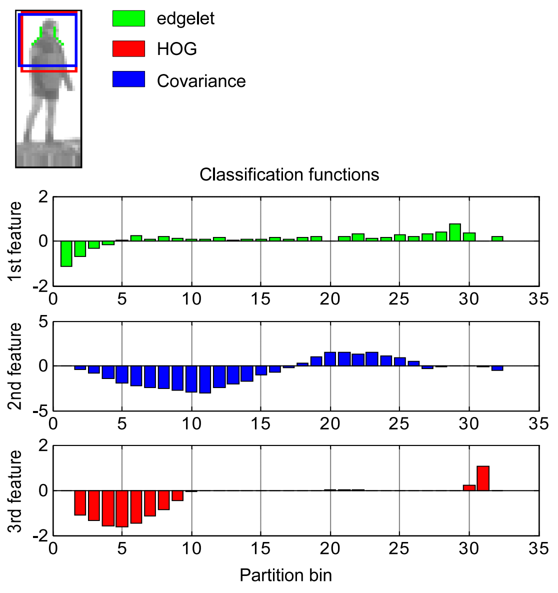


Figure 4. The first weak classifier learned for pedestrians and its selected features. The first feature evaluated is an edgelet corresponding to the head-shoulder contour of human body; the second feature is a covariance descriptor whose supporting region surrounds the head-should part; the third feature is a HOG descriptor. The x-axis is the index of the histogram bins, *i.e.* the partition along the projection direction. The y-axis is the classifier output.

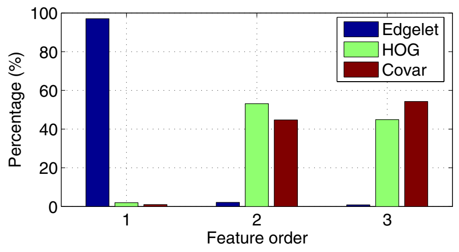


Figure 5. Frequencies of different feature types as the first, second, third feature in the hierarchical weak classifiers.

Feature type	Edgelet	HOG	Covar
Evaluation frequency per window	15.25	2.6	2.05

Table 1. Evaluation frequencies of different feature types.

4.3. Hierarchical *vs.* sequential weak classifier

We compare the performance of our hierarchical feature combination method with other two combination strategies: sequential summation and sequential maximum. Sequential summation is defined by

$$h_{\{f'_1..f'_m\}}(\mathbf{x}) = \sum_{k=1}^{m} h_{f'_k}(\mathbf{x}) \tag{15}$$

and sequential maximum is defined by

$$h_{\{f'_1..f'_m\}}(\mathbf{x}) = h_{f^M}(\mathbf{x}) \tag{16}$$

where $f^M = \underset{f'_k}{\arg\max} \left\{ \left| h_{f'_k}(\mathbf{x}) \right| \right\}$. Both of these strategies require evaluating all the features before making a prediction. They can be considered the accuracy upper bound of our hierarchical strategy. We take the first 10 features of the single-threshold pedestrian classifier in section 4.1, apply these three combination strategies and evaluate the classification performance on the test set. Fig.6 shows the ROC curves. It can be seen that the performance of the two sequential strategies is almost the same, and slightly better than that of our hierarchical strategy, but they are about five times slower than our method.

Feature order	First	Second	Third
Evaluation frequency per window	16.33	2.66	0.91

Table 2. Evaluation frequencies of the first, second, third features.

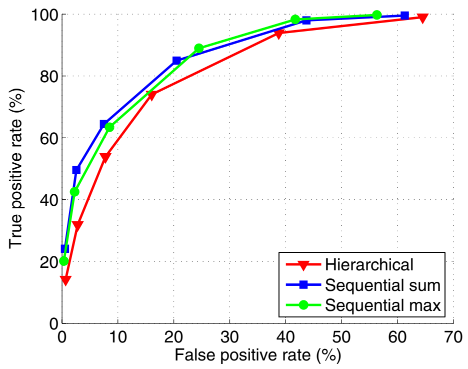


Figure 6. Comparison of different feature combination strategies.

4.4. Performance on cars

For car detection, we manually labeled 4,000 car samples of various models and different viewpoints from the MIT street scene images[2] [19], and collected 7,000 background images from the Internet as our training set. The car samples are normalized to 64 × 32 pixels. As the inner-class variation of multi-view cars is large, we train a tree structured detector with four leaves by the Cluster Boosting Tree (CBT) method in [2]. This is is an enhanced version of cascade.

For testing, we collected 390 car images from the PASCAL 2006 challenge data set [13]. This set includes multi-view cars of different models. For evaluation, we only consider the cars higher than 32 pixels. There are overall 481 counted cars in this set. The data set contains two different types of images, close shots and mid/long-distant shots. In the close shot images, we detect cars from 250 to 500 pixels high; in the mid/long-distant shot images, we detect cars from 32 to 250 pixels high. Following the PASCAL challenge, buses are not included in the car class. Positive responses on buses are counted as false alarms. Fig.7 shows the precision-recall curve of our method on this set. The

[2]`http://cbcl.mit.edu/software-datasets/streetscenes/`

equal precision-recall rate is about 81%. Hoiem *et al*. [3] use 150 car images from the PASCAL 2006 data for testing and their method achieves an equal precision-recall rate of about 61%. The highest reported results in the PASCAL 2006 and 2007 challenges have the equal precision-recall rates of about 45% and 55% respectively [29]. However, these rates are for the whole test set, which is much more difficult. Fig.8(b) shows some example results of car detection.

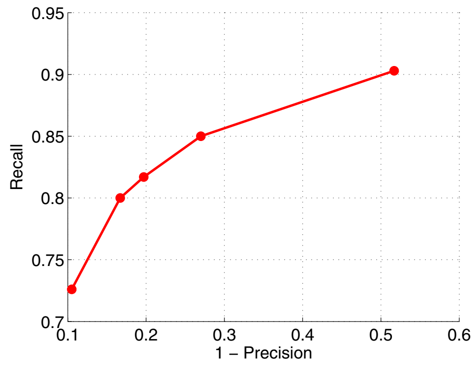


Figure 7. Performance of multi-view car detection.

5. Conclusion and Discussion

We described a framework to integrate different types of features for object detection. We learn strong object classifiers by boosting weak classifiers. Each weak classifier is based on several different types of features, which are ranked according to their speed normalized classification margins. The weak classifier makes the prediction by examining the feature types one by one to optimize the discrimination-efficiency tradeoff.

As long as the features used are not highly correlated, combination of them should result in better accuracy than by using any single one of them. However, if one feature truly dominates another one on all samples but is slower, it is possible that the accuracy of the combination is higher than the weaker one but lower than the stronger one. We demonstrated our approach on two object classes, pedestrians and cars, with three feature types, edgelet, HOG and covariance descriptors. However, our method is not limited to these types of features; new features could be readily integrated into the framework.

Acknowledgements: This research was funded, in part, by the U.S. Government VACE program.

References

[1] M. Varma, and D. Ray. Learning the Discriminative Power-Invariance Trade-off. ICCV 2007. 2

[2] B. Wu, and R. Nevatia. Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. ICCV 2007. 1, 5, 6

[3] D. Hoiem, C. Rother, and J. Winn. 3D LayoutCRF for Multi-View Object Class Recognition and Segmentation. CVPR 2007. 1, 7

[4] P. Sabzmeydani and G. Mori. Detecting Pedestrians by Learning Shapelet Features. CVPR 2007. 1

[5] O. Tuzel, F. Porikli, and Peter Meer. Human Detection via Classification on Riemannian Manifolds. CVPR 2007. 1, 2, 4, 5

[6] D. M. Gavrila. A Bayesian, Exemplar-based Approach to Hierarchical Shape Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(8): 1408-1421, 2007. 1

[7] J. Mutch, and D. Lowe. Multiclass Object Recognition with Sparse, Localized Features. CVPR 2006. 1

[8] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng. Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. CVPR 2006. 4, 5

[9] A. Opelt, A. Pinz, and A. Zisserman. A Boundary-Fragment-Model for Object Detection. ECCV 2006. 1

[10] N. Dalal, B. Triggs, and C. Schmid. Human Detection Using Oriented Histograms of Flow and Appearance. ECCV 2006. 1, 2

[11] J. Shotton, J. Winn, C. Rother, and A.Criminisi. TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation. ECCV 2006. 1, 2

[12] C. Huang, H. Ai, Y. Li, and S. Lao. Learning Sparse Features in Granular Space for Multi-View Face Detection. FG 2006. 1, 2

[13] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. Technical report, 2006. 6

[14] N. Dalal, and B. Triggs. Histograms of Oriented Gradients for Human Detection. CVPR 2005. 1, 2, 4, 5

[15] B. Leibe, E. Seemann, and B. Schiele. Pedestrian Detection in Crowded Scenes. CVPR 2005. 1

[16] B. Wu, and R. Nevatia. Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors. ICCV 2005. 1, 2, 4

[17] C. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. ECCV 2004. 1

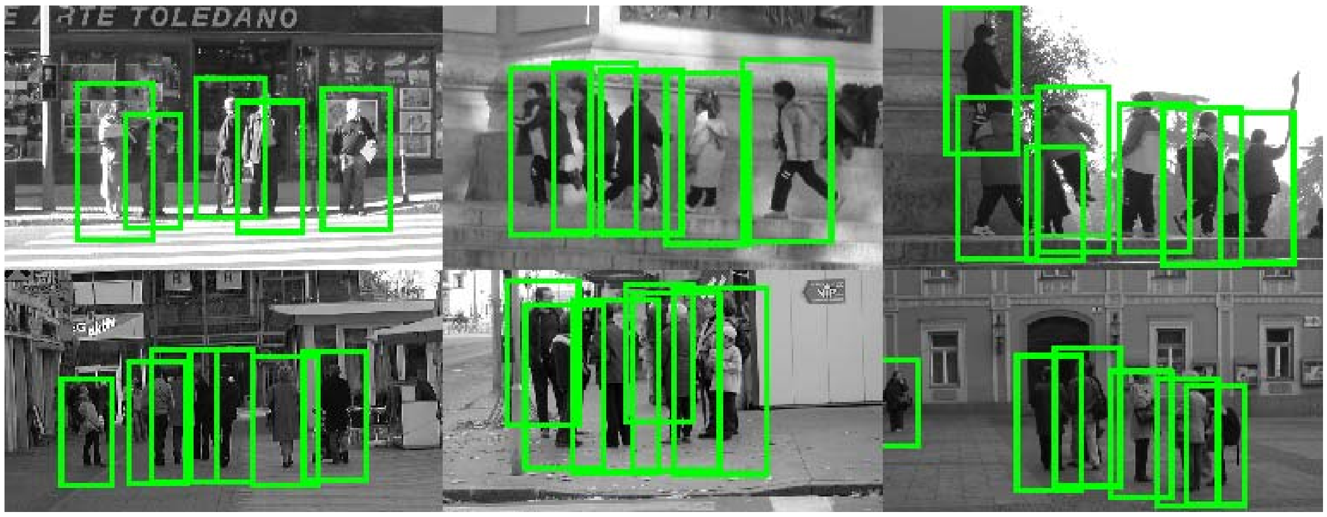
[18] B. Leibe, A. Leonardis, and B. Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. Workshop on Statistical Learning in Computer Vision, in conjunction with ECCV 2004. 1

[19] B. Leung. Component-based Car Detection in Street Scene Images. Master's Thesis, EECS, MIT, 2004. 6

[20] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. ICCV 2003. 1, 2

[21] W. M. Boothby. An Introduction to Differentiable Manifolds and Riemannian Geometry. Academic Press, 2002. 4

[22] H. Altincay, and M. Demirekler. Post-processing of Classifier Outputs in Multiple Classifier Systems. Lecture Notes in Computer Science, LNCS 2364, Springer Verlag, pp. 159-168, 2002. 3


(a) Example results of pedestrian detection

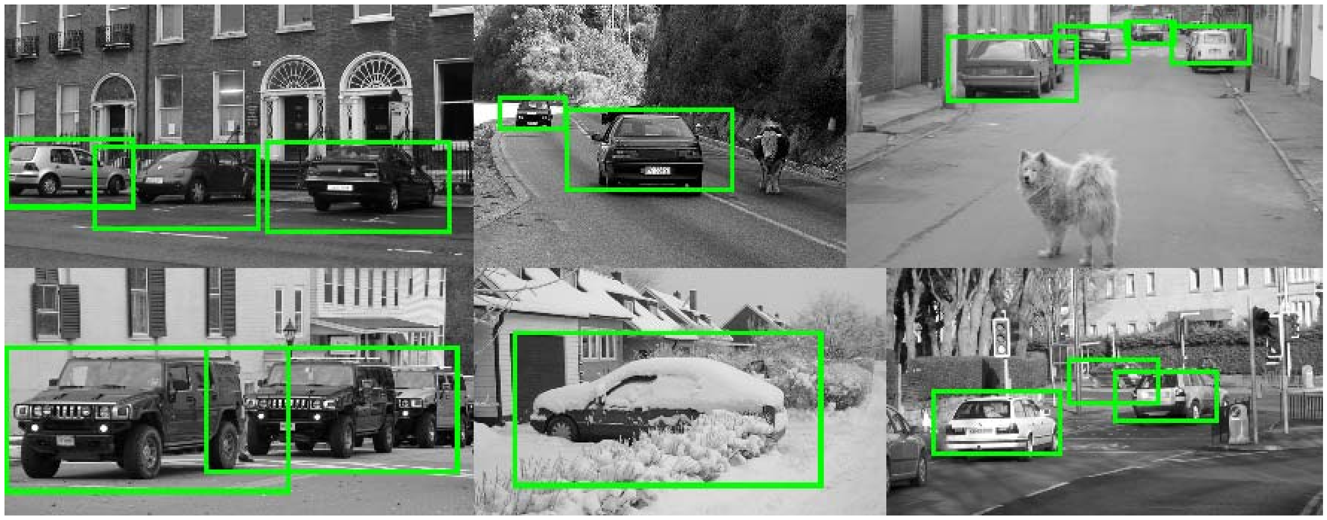

(b) Example results of car detection

Figure 8. Example detection results.

[23] P. Viola, and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. CVPR 2001. 1, 2

[24] D. Gavrila. Pedestrian detection from a moving vehicle. ECCV 2000. 1

[25] H. Schneiderman, and T. Kanade. A Statistical Method for 3D Object Detection Applied to Faces and Cars. CVPR 2000. 1

[26] R. E. Schapire and Y. Singer. Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning, 37: 297-336, 1999. 3

[27] C. Papageorgiou, T. Evgeniou, and T. Poggio. A Trainable Pedestrian Detection System. In: Proc. of Intelligent Vehicles 1998. pp. 241-246 1

[28] R. E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. The Annals of Statistics, 26(5): 1651-1686, 1998 3

[29] `http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html` 7

[30] `http://www.intel.com/cd/software/products/asmo-na/eng/302910.htm` 4