Fast Object Detection with Occlusions

Yen-Yu Lin[1], Tyng-Luh Liu[1], and Chiou-Shann Fuh[2]

[1] Inst. of Information Science, Academia Sinica, Nankang, Taipei 115, Taiwan,
liutyng@iis.sinica.edu.tw
[2] Dept. of CSIE, National Taiwan University, Taipei 106, Taiwan

Abstract. We describe a new framework, based on *boosting algorithms* and *cascade structures*, to efficiently detect objects/faces with occlusions. While our approach is motivated by the work of Viola and Jones, several techniques have been developed for establishing a more general system, including (i) *a robust boosting scheme*, to select useful weak learners and to avoid overfitting; (ii) *reinforcement training*, to reduce false-positive rates via a more effective training procedure for boosted cascades; and (iii) *cascading with evidence*, to extend the system to handle occlusions, without compromising in detection speed. Experimental results on detecting faces under various situations are provided to demonstrate the performances of the proposed method.

1 Introduction

While object detection has long been an important and active area in vision research, most of its applications now demand not only *accuracy* but also (real-time) *efficiency*. Often, to address these two concerns satisfactorily, a typical detection system considers only a certain *regular class* of target objects even though the restriction may limit its practical use. In [17], Viola and Jones propose an effective scheme using *AdaBoost* to detect faces through a *boosted cascade*. Their framework has prompted considerable interest in further investigating the use of boosting algorithms and cascade structures for fast object detection, e.g., [1],[5],[6],[7]. Our detection method also relies on the two elements, but different from the foregoing works, we aim to develop a more general detection system by focusing on the issues of *overfitting* and *occlusion*.

Previous Work. The literature on object detection is quite extensive. We discuss only some of the recent works, especially those based on learning. Also, unless further specified, we focus hereafter on the subject of *face detection*.

Methods based on dimension reduction are often used in detecting faces. Moghaddam and Pentland [9] propose an approach for face detection by calculating several *eigenfaces* from training data. Each detected pattern is then projected into a feature space, formed by the eigenfaces. A testing pattern is classified as face or non-face, depending on its DIFS (Distance-In-Feature-Space) and DFFS (Distance-From-Feature-Space). In [18], Yang et al. develop a face detection system using FLD (Fisher Linear Discriminant), and consider the face classification in a feature space, spanned by the so-called *fisherfaces*.

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3021, pp. 402–413, 2004.

Sung and Poggio [16] establish a neural network approach that uses a set of face and non-face prototypes to build the hidden layer. The final output is decided by measuring the distances from the detected pattern to each of these prototypes. In [10], Osuna et al. describe an SVM-based method for face detection. They train a hyperplane in some high-dimension feature space for separating faces and non-faces. Each testing pattern is mapped to the feature space for face classification. Romdhani et al. [12] present another SVM-based face detection system by introducing the concept of *reduced set vectors*. Using a sequential evaluation strategy, they report that their face detector is about 15 times faster than the one of Osuna et al. [10]. The SNoW (Sparse Network of Winnows) face detection system by Roth et al. [13] is a sparse network of linear functions that utilizes winnows update rules. They show SNoW is computationally more efficient, and yields better results than those derived by [10].

The excellent work of Viola and Jones [17] has redefined what can be achieved by an efficient implementation of a face detection system. They formulate the detection task as a series of non-face rejection problems. In addition, they calculate an *integral image* to speed up *rectangle feature* computation, and apply AdaBoost to construct stage-wise face classifiers. Since then, a number of systems have been proposed to extend the idea of detecting faces through a boosted cascade. For example, Li et al. [7] develop a system to detect side-view faces by using a coarse-to-fine, simple-to-complex architecture. They divide side-view faces into nine classes, and report that the resulting detector requires about three times computation time than Viola and Jones's to detect the nine kinds of side-view faces. Yet another work by Lienhart and Maydt [5] focuses on extending the set of rectangle features. They rotate extended rectangle features by $\pm 45^{\circ}$ to obtain rotated rectangle features, and also calculate rotated integral images. In this way, the system can efficiently compute rotated rectangle features by array references. More recently, Liu and Shum [6] introduce a *Kullback-Leibler boosting* to derive weak learners by maximizing projected KL distances. In [1], face patterns in video streams are detected by a boosted cascade, and then classified into different classes of facial expressions. A novel combination of AdaBoost and SVMs (AdaSVMs) is employed so that features selected by AdaBoost are used to form the mapping to a reduced representation for training SVMs.

Our Approach. We generalize the work of Viola and Jones [17] to efficiently detect objects with occlusions. We also deal with the problem of overfitting in training boosted cascades, and thus derive a more robust system. Specifically, the proposed approach leverages its detection performances with three key components. First, we establish a new boosting algorithm that at each iteration, the selection of a weak learner and its coefficient can be determined simultaneously. Each classifier is then formed by a linear combination of the chosen weak learners using a soft-boosting scheme. Second, we propose a reinforcement training procedure to dynamically add difficult and representative training data in each stage. This makes the resulting classifier more general and discriminant. Third, we design a cascading-with-evidence scheme to handle occlusions. The resulting system can detect complete frontal faces and occluded faces at the same time.

2 Classification Using Boosting

Originated from Kearns and Valiant's question [4] to improve the performance of a *weak* learning scheme, boosting has now become one of the most important recent developments in classification methodology. It elegantly leads to a general approach to improve the accuracy of any given learning algorithm. In 1989, Schapire [15] introduces the first polynomial-time boosting procedure. However, it is the AdaBoost [3], proposed by Freund and Schapire, that stimulates the widespread research interest in boosting. A carefully implemented boosting algorithm often gives a compatible performance, with more efficiency, to those yielded by current best classification methods, e.g., SVMs, HMMs.

In this section, we discuss first the ideas of boosting, and then focus on an exponential-loss upper bound on training error. Motivated by [6], we also consider the selection of weak learners by analyzing the weighted projected data. Moreover, we show that an easier-to-implement boosting algorithm can be derived by directly analyzing the error bound, and by addressing overfitting.

2.1 The Ideas behind Boosting

The basic concepts of boosting can be best understood by illustrating with the AdaBoost. Consider now a training set, $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$, where the first component $\mathbf{x}$ of each sample is the feature value(s), and y is its label. For a two-class classification problem like face detection, $y = 1$ (face) or -1 (non-face), i.e., $D = D^+ \cup D^-$. To elevate the classification performance, AdaBoost uses data re-weighting w_t on D, at iteration t, to iteratively select a weak learner h_t and decide its coefficient α_t in the linear combination of weak learners. It is known that such an iterative process is indeed an attempt to minimize an upper bound of the training error [14]. More precisely, say after T iterations, the training error of a strong classifier $H(\mathbf{x}) = sign(f(\mathbf{x})) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$ can be bounded as follows.

$$\frac{1}{\ell}\sum_{i=1}^{\ell}\frac{1}{2}|y_i - H(\mathbf{x}_i)| \leq \frac{1}{\ell}\sum_{i=1}^{\ell}\exp(-y_i f(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t\,, \tag{1}$$

where $Z_t = \sum_{i=1}^{\ell} w_t(i)\exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right)$. At each iteration t, AdaBoost tries to minimize the error bound by reducing Z_t as much as possible via *steepest descent*. When weak learners h_ts are restricted to be binary, it leads to the choice of α_t in [17]. Nevertheless, the relation in (1) still holds for weak learners assuming real values—*a crucial property for selecting good weak learners*.

2.2 Boosting without Overfitting

The foregoing discussion simply points out the two main elements of a boosting algorithm: *weak-learner selection* and *data re-weighting*. For AdaBoost, the new data weight $w_{t+1}(i)$ can be explicitly computed from $w_t(i)$ and α_t. Furthermore,

recent studies suggest that AdaBoost may overfit when the training data contain highly noisy patterns [2], [11]. For face detection via learning, the problem of overfitting is especially delicate and must be handled appropriately in that there are quite a number of non-face patterns resembling faces.

Effective Weak Learner Selection. When efficiency is emphasized, it is preferable to have a classifier of fewer weak learners to achieve the required training accuracy. Meanwhile, the mechanism to select weak learners should take account of its implication on data re-weighting. For instance, the fast detection system of Viola and Jones [17] considers binary weak learners from thresholding on rectangle features. Though their scheme may choose weak learners that are too crude for effectively discriminating the face and non-face distributions, it does have the advantage of using a straightforward updating scheme on data weights w_t, through the analytic form of α_t. On the other hand, the KL boosting of Liu and Shum [6] computes weak learners by maximizing the relative entropy between two 1-D projected distributions of face and non-face samples. At each iteration t, all the coefficients $\alpha_1, \ldots, \alpha_t$ for combining the chosen weak learners are re-evaluated and optimized *in parallel*. As a result, the data weights are updated according to heuristic formulas (defined in (8) and (9) of [6]). Motivated by these observations, we describe a method to select useful real-valued weak learners of *positive unit coefficients*, and to conveniently perform data re-weighting iteratively by following the AdaBoost manner.

Assume that we have a set of 1-D mappings $\{\phi_i\}_{i=1}^n$ that each ϕ projects the training data D into real-valued scalars. In our approach, the mapping ϕ will be defined uniquely by a rectangle feature. Thus, we could further assume each ϕ has a compact support. This implies that it is possible to compute histogram distributions for the projected data with a pre-defined partition of m equal-size bins over a finite range of the real line, denoted as $\{b_k\}_{k=1}^m$. Now, focus on how to derive good weak learners with the projected data. Similar to the AdaBoost algorithm used in [17], we try to find, at each iteration t, a weak learner h_t by minimizing Z_t. The differences are: 1) h_t need not be binary, and 2) like [6], each h_t is defined by considering the two-class weighted histograms of projected training data. As our discussion below applies to all iterations, we shall drop the subscript t to simplify the notations.

For each projection ϕ, we define $\mathbf{i}_k(\phi) = \{i \,|\, \mathbf{x}_i \in D, \phi(\mathbf{x}_i) \in b_k\}$, the indexes of training data being projected by ϕ into bin b_k. Analogously, $\mathbf{i}_k^+(\phi)$ and $\mathbf{i}_k^-(\phi)$ are defined, respectively, for $\mathbf{x}_i \in D^+$ and D^-. With these notations, we are ready to evaluate the values of weighted positive histogram and negative histogram of b_k by

$$p_k^+(\phi) = \sum_{\mathbf{i}_k^+(\phi)} w(i) \quad \text{and} \quad p_k^-(\phi) = \sum_{\mathbf{i}_k^-(\phi)} w(i). \tag{2}$$

Notice that the two weighted histograms p^+ and p^- are not normalized into distributions. Nevertheless, defining them in this way will be more convenient for our analysis, and also without any bearings on the classification outcomes.

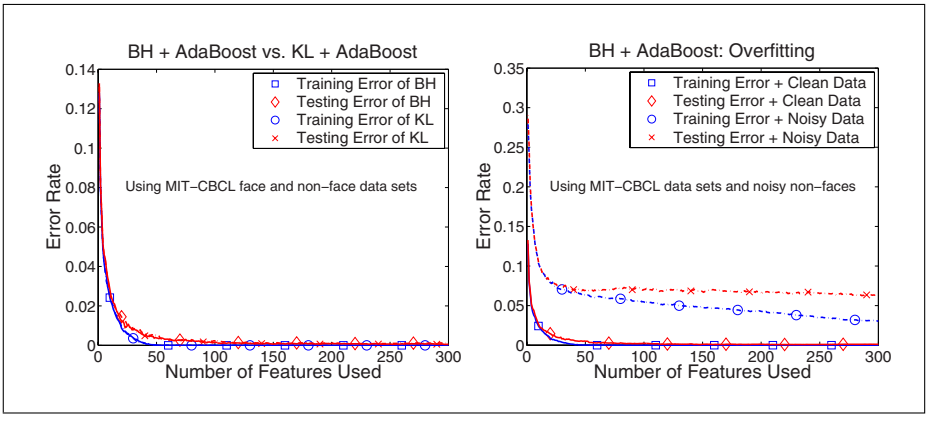

Fig. 1. (a) Using BH or KL weak learners gives a comparable boosting performance. (b) Overfitting is perceived by increasing gaps between training and testing errors.

To establish rules for selecting good weak learners, we consider a projection mapping ϕ and an arbitrary $\mathbf{x} \in D$ such that $\phi(\mathbf{x}) \in b_k$. Let h_ϕ be the weak learner arising from ϕ. Then, it is reasonable to establish the definition of h_ϕ by assuming $h_\phi(\mathbf{x})$ depends on some quantity related to bin b_k. In particular, we write $h_\phi(\mathbf{x}) = s_k$, where s_k is a real scalar. Applying this relation to the definition of Z, we have $Z = \sum_{k=1}^{m} \sum_{\mathbf{i}_k(\phi)} w(i) \exp(-\alpha y_i s_k)$. Following the strategy of AdaBoost [14], the best choices of s_k, for $k = 1, \ldots, m$, can be obtained by minimizing Z with respect to each s_k:

$$\frac{dZ}{ds_k} = 0 \Rightarrow s_k^* = \frac{1}{\alpha} \ln \sqrt{p_k^+(\phi)/p_k^-(\phi)} \quad \text{and} \quad Z^* = 2 \sum_{k=1}^{m} \sqrt{p_k^+(\phi) p_k^-(\phi)}. \tag{3}$$

The equations in (3) suggest that we can re-scale h_ϕ by multiplying with α such that the selection of weak learner and its coefficient can be decided simultaneously. In addition, the *strength* of a weak learner with respect to the weighted training data can now be measured explicitly with the Bhattacharyya coefficient. We summarize these observations into the following two criteria:

1. Each mapping ϕ implicitly defines a weak learner h_ϕ of coefficient 1 by

$$h_\phi(\mathbf{x}) = \ln \sqrt{p_k^+(\phi)/p_k^-(\phi)}\,, \quad \text{for all } \mathbf{x} \in D \text{ and } \phi(\mathbf{x}) \in b_k.$$

2. At each iteration, the best weak learner h_ϕ^* is the one that yields the minimal Bhattacharyya coefficient. This result somewhat supports the use of Kullback-Leibler distance in [6]. However, besides a much easier computation, using the Bhattacharyya coefficient has the advantage to skip the estimation of coefficient α, and most of all, it guarantees to optimally minimize Z. In Figure 1a, comparisons between using AdaBoost with the two criteria for selecting weak learners indicate a comparable classification performance.

Algorithm 1: Soft-Boosting with Bhattacharyya weak learners.

Input : A weak learning algorithm *BH-WeakLearn* derived from (3), the number of iterations T, and ℓ labeled training data D.

Output : A strong classifier H.

Initialize the weight vector $w_1(i) = 1/\ell$, *for* $i = 1, \ldots, \ell$.

for $t \leftarrow 1, 2, \ldots, T$ **do**

1. Call *BH-WeakLearn*, using distribution w_t on D, to derive $h_t : X \to \mathbb{R}$.
2. $w_{t+1}(i) \leftarrow w_t(i) \exp\left(-y_i h_t(\mathbf{x}_i)\right) / Z_t$, for $i = 1, 2, \ldots, \ell$. (Z_t is a normalization factor such that w_{t+1} is a distribution.)

Call LP_{REG}-AdaBoost, with inputs $\{y_i h_t(\mathbf{x}_i)\}_{t=1}^{T}$, to output $\{\beta_t\}_{t=1}^{T}$.

Output the final strong classifier: $H(\mathbf{x}) = sign(f(\mathbf{x})) = sign(\sum_{i=1}^{T} \beta_t h_t(\mathbf{x}))$.

Table 1. Error Rate: Soft-Boosting against different degree of noisy non-face data.

# of h_t	No Noise	25% Noise		50% Noise		75% Noise		100% Noise	
	BH	BH	BH+Soft	BH	BH+Soft	BH	BH+Soft	BH	BH+Soft
100	0.23	2.96	2.36	4.91	4.07	5.50	4.54	7.03	5.85
200	0.12	2.29	1.99	4.11	3.41	5.16	3.91	6.65	5.23
300	0.11	2.13	1.85	3.77	2.92	5.23	3.74	6.39	4.97

Soft-Boosting. Rätsch et al. [11], show that AdaBoost behaves asymptotically like a hard-margin classifier, and propose to use soft margins for AdaBoost (or *soft-boosting* for abbreviation) to avoid data overfitting (see Figure 1b). Extending AdaBoost to incorporate soft margins allows certain of difficult patterns to be misclassified within some ranges during the training stage. Such a strategy has been tested extensively and successfully in SVMs. We adopt the LP_{REG}-AdaBoost in [11] that uses linear programming with slack variables to achieve soft margins. The LP_{REG}-AdaBoost is paired nicely with our implementation in that it only needs the T weak learners and the margin distributions as inputs. The coefficients of weak learners are not used at all. Since the resulting weak learners have positive unit coefficients, all the available information from the training stage for selecting weak learners is passed to the regularized linear programming. In Table 1, we summarize experimental results of applying soft-boosting to deal with overfitting, using the MIT-CBCL face and non-face data sets. The noisy non-face data are generated from the false positive samples at different stages of a cascade structure, proposed by Viola and Jones [17]. When combining the Bhattacharyya weak learners with soft-boosting, consistent improvements in the detection rates are obtained in all experiments. A complete description of our method is listed in Algorithm 1.

3 Fast Detection with Occlusions

We construct a feature-based face detection system through a *simple-to-complex* cascade structure. Such a strategy reduces a face detection problem into the rejections of non-face patterns stage-wise. As it turns out, to detect occluded

faces via a boosted cascade is a nontrivial problem because they are very likely to be rejected at some intermediate stage. While adopting a more lenient policy in rejecting non-face patterns may be able to handle occlusions to some degree, it often causes an increase in the false-positive detection rate. In dealing with these issues, we propose a method that consists of two schemes: 1) *reinforcement training*, to reduce the false positive rates, and 2) *cascading with evidence*, to detect faces with occlusions.

3.1 Reinforcement Training

In a cascade structure $\mathcal{M}$, the face detector at stage k can be denoted as $H_k(\mathbf{x}) = sign(f_k(\mathbf{x})) = sign(\sum_{t=1}^{T_k} \beta_t h_t(\mathbf{x}))$, where the size of T_k increases as k becomes large. We also use $\mathcal{M}_1^k$ to represent the *sub-cascade* of the first k stages. To train a cascade of face detectors, we usually start with a balanced two-class data D, i.e., the same number of face and non-face data. During training, data arriving at the kth stage are those that pass the sub-cascade $\mathcal{M}_1^{k-1}$. Among them, most are face data and only a few are non-face. The situation can lead to a problem that there could be too few non-face data to be trained with, and consequently, a boosting method may yield unreliable decision boundary predictions. To alleviate this problem, we collect an additional set $\mathcal{N}$ of images that do not contain any face patterns, and then perform a two-step reinforcement training whenever there are too few non-face samples reaching some stage k.

1. The *Bootstrap* technique is applied to generate non-face patterns by testing all images in $\mathcal{N}$ with the sub-cascade $\mathcal{M}_1^{k-1}$. Those that survive are indeed false positives of $\mathcal{M}_1^{k-1}$. We denote them as $\mathcal{N}_k$.
2. In practice, when k is small, there are way too many samples of $\mathcal{N}_k$ to be considered. For each $\mathbf{x} \in \mathcal{N}_k$, the mapping $\mathbf{x} \mapsto (\tilde{f}_1(\mathbf{x}), \ldots, \tilde{f}_{k-1}(\mathbf{x}))$ is used to associate $\mathbf{x}$ with a $(k-1)$-dimension feature vector. (Note that each $\tilde{f}$ is normalized from the stage-wise f such that $\sum_{\mathbf{x} \in \mathcal{N}_k} \tilde{f}(\mathbf{x}) = 1$.) Then, *k-means clustering* is applied to divide $\mathcal{N}_k$ into six clusters [16] to model the empirical non-face distribution. The needed non-face samples, at each stage k, can now be selected uniformly from the six clusters.

The reinforcement strategy enables the system to consider meaningful and difficult non-face samples from $\mathcal{N}$. Though it is difficult to model the distribution of non-face patterns, we establish an effective way in selecting representative ones. With reinforcement training, the system is expected to have a lower false positive rate in each cascade stage of a testing procedure.

3.2 Cascading with Evidence

Mentioned briefly in Section 2.2, rectangle features used in [17] can be computed rather efficiently by referencing integral images. A rectangle feature simply computes the difference between sums of pixel intensity in adjacent regions, and hence a strong classifier formed by these rectangle features records intensity distribution in the faces. Apparently, detection systems using rectangle features are

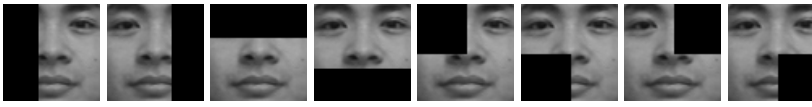

Fig. 2. From left to right: eight types of occluded faces (1/3 or 1/4 occlusions).

sensitive to occlusions because the intensity differences related to occluded regions are no longer reliable. We instead treat each rectangle feature as a mapping ϕ that projects $\mathbf{x} \in D$ to the resulting intensity difference, and then compute the Bhattacharyya coefficient between the weighted positive and negative histograms. Still, the derived classifiers only work for *regular* faces—to account for occlusions, other mechanisms are needed.

To distinguish an occluded face from non-face samples, the clues lie in the *evidence* left behind when a testing sample is in question. In Figure 2, there are eight types of occlusions that our detection system is designed to handel. The face data we choose to train our system are images of size 20×20. Totally, about $80,000$ rectangle features can be generated, and represented as a set Ψ. (We use the same three types of rectangle features proposed in [17].) Then, for each type $\mathcal{I}$ of occluded faces shown in Figure 2, we use $\mathcal{O}_\mathcal{I}$ to denote the occluded region, and define the largest subset of Ψ disjoint from $\mathcal{O}_\mathcal{I}$ by

$$\Psi_\mathcal{I} = \{\psi \,|\, \psi \in \Psi \text{ and } \psi \cap \mathcal{O}_\mathcal{I} = \emptyset\}, \quad \text{for } \mathcal{I} = \mathcal{A}, \mathcal{B}, \ldots, \mathcal{H}. \tag{4}$$

Now, in testing a sample $\mathbf{x}$ at some stage k of the cascade, besides calculating $H_k(\mathbf{x})$, we also compute an additional eight-dimensional feature vector, the evidence of $\mathbf{x}$ at stage k, defined as follows:

$$\mathcal{E}_k(\mathbf{x}) = (f_k^\mathcal{A}(\mathbf{x}), f_k^\mathcal{B}(\mathbf{x}), \ldots, f_k^\mathcal{H}(\mathbf{x})) \quad \text{and} \quad f_k^\mathcal{I}(\mathbf{x}) = \sum\nolimits_\mathcal{I} \beta_t h_t(\mathbf{x}), \tag{5}$$

where the summation $\sum_\mathcal{I}$ involves only those weak learners that their corresponding rectangle features do not intersect with $\mathcal{O}_\mathcal{I}$, for $\mathcal{I} = \mathcal{A}, \mathcal{B}, \ldots, \mathcal{H}$. With (5), we propose a *cascading with evidence* scheme to detect faces with the eight types of occlusions efficiently, where its advantages are summarized below.

- Since each $\beta_t h_t(\mathbf{x})$ has already been evaluated in the computation of $H_k(\mathbf{x})$, the evidence vector $\mathcal{E}_k(\mathbf{x})$ is easier to derive.
- To illustrate, let $\mathbf{x}$ be a face sample of type-$\mathcal{A}$ occlusion, and $\mathbf{x}$ is being considered to be rejected as a non-face pattern due to $H_k(\mathbf{x}) < 0$. Then, we can reference its evidence vectors from the k stages. In particular, the majority of $f_1^\mathcal{A}, \ldots, f_k^\mathcal{A}$ should be positive responses to indicate $\mathbf{x}$ is a type-$\mathcal{A}$ occluded face. Such a property is not shared by most true non-face samples. The details of cascading with evidence are given in Algorithm 2 and 3.

Algorithm 2: Cascading with Evidence: Training Procedure

Input : Rectangle feature sets, Ψ and $\Psi_{\mathcal{I}}$, for $\mathcal{I} = \mathcal{A}, \mathcal{B}, ..., \mathcal{H}$.
Output : A main cascade $\mathcal{M}$, and 8 occlusion cascades, $\mathcal{I} = \mathcal{A}, \mathcal{B}, ..., \mathcal{H}$.

1. Train a regular cascade $\mathcal{V}$ from Ψ, using the techniques in [17].
2. Train 8 occlusion cascades $\mathcal{I}$ from $\Psi_{\mathcal{I}}$, using $\mathcal{V}$ as a benchmark.
3. $T_k^{\mathcal{I}} \leftarrow$ Number of weak learners used at the kth stage of $\mathcal{I}$.
4. Train cascade $\mathcal{M}$ such that $|\{h_1, \ldots, h_{T_k}\} \cap \Psi_{\mathcal{I}}| \geq T_k^{\mathcal{I}}$, for each stage k.

Algorithm 3: Cascading with Evidence: Testing Procedure

Input : A testing pattern, $\mathbf{x}$.
Output : Face, Non-Face, or Type-$\mathcal{I}$ Occluded Face.

1. If $\mathbf{x}$ goes through $\mathcal{M}$ return Face.
2. If $\mathbf{x}$ is rejected at stage k and all $f_k^{\mathcal{I}} < 0$ return Non-Face.
3. Dispatch $\mathbf{x}$ to cascade $\mathcal{I}$ if $f_k^{\mathcal{I}} > 0$ and $\sum_{t=1}^{k} f_t^{\mathcal{I}}$ is the largest.

if $\mathbf{x}$ *goes through* $\mathcal{I}$ **then**
  return Type-$\mathcal{I}$ Occluded Face

else
  return Non-Face

4 Experimental Results

The face training data are obtained from MIT-CBCL database and AR [8] face database. They are pictured under different lighting, facial expressions, and poses. We rotate ($\pm 15°$) and mirror each face image, and crop the face region with slightly different scales. The non-face training data are collected from the Internet. Both face and non-face training data are resized at the resolution, 20 by 20 pixels. Totally, $10,000$ face images and $10,000$ non-face images are used as our initial training data. We also prepare about $16,000$ images that contain no faces for generating non-face training data in reinforcement training.

A rectangle feature is indeed a Haar wavelet filter that maps each training image into a real value. Thus, each rectangle feature gives rise to two different distributions for face and non-face data. When the number of training samples is fixed, the number of bins used to model the two distributions is decided on the tradeoff between accuracy and data overfitting. Empirically, we have used 10 bins to derive satisfactory results. At each stage, we aim to construct a face detector with a loose threshold by reducing the false positive rate, while detecting almost all positive samples. This is achieved by adding weak learners until the false positive rate is less than 40%, and also the detection rate is higher than 99.9%.

In our implementation, a regular cascade to detect frontal faces (without occlusion) has 21 stages, including 872 weak learners. The first three stages contain 2, 3, and 5 weak learners respectively. We test it on the benchmark testing set, i.e., CMU+MIT data set. It contains 130 images with 507 frontal faces. There are totally $81,519,506$ sub-windows scanned. The detecting results are represented as an ROC curve shown in Figure 3b. The performance is comparable to other

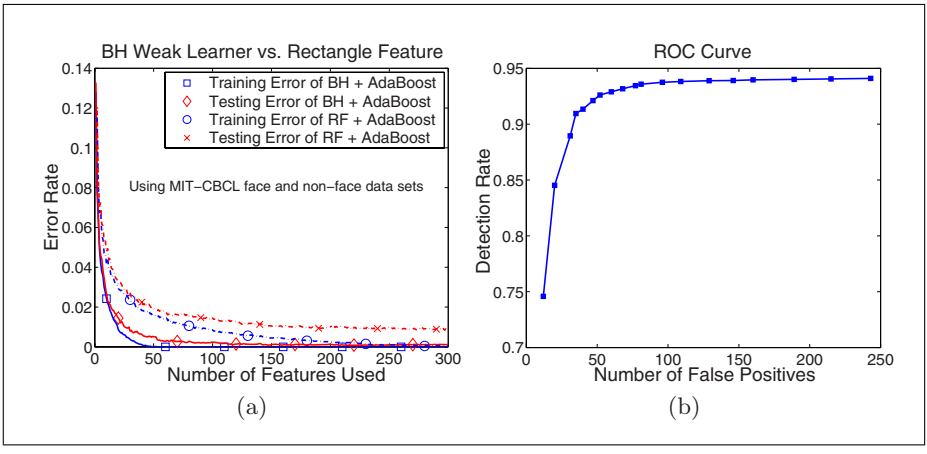

Fig. 3. (a) Using Bhattacharyya weak learners outperforms implementation with rectangle features in accuracy and convergence speed. (b) The ROC curve of our detector performed on CMU+MIT data set (81, 519, 506 sub-windows scanned).

Table 2. Stage Passing Rate of a Cascade Using CMU+MIT Data Set

Stage	Passing Rate	Stage	Passing Rate	Stage	Passing Rate	Stage	Passing Rate
1	0.35503	4	0.01254	7	0.00082	10	0.00018
2	0.10934	5	0.00383	8	0.00048	11	0.00013
3	0.04611	6	0.00165	9	0.00023	12	0.00009

existing face detectors, e.g., [16], [17]. The stage-wise classification efficiency is shown in Table 2. The first stage rejects about 64.5% sub-windows (almost non-face). Averagely, a sub-window is classified by only 4.59 weak learners. In addition, the advantages of using Bhattacharyya weak learners in improving accuracy and convergence speed are also illustrated in Figure 3a.

To detect frontal and occluded faces at the same time, we need a main cascade, $\mathcal{M}$, and eight occlusion cascades. In designing $\mathcal{M}$, besides satisfying the detection rate constraints mentioned above, at each stage, the number of weak learners used should satisfy the condition described in Algorithm 2-(**4**). The requirements are to ensure that every component of the evidence vector $\mathcal{E}_k$ is well-defined at each stage k. Since a weak learner may be associated with more than one type. The number of weak learners used in each stage of the main cascade $\mathcal{M}$ is still manageable to produce efficient detection. (The first three stages of $\mathcal{M}$ contain 7, 9, and 12 weak learners, respectively.) Regarding the eight occlusion cascades, each of them contains from 23 to 26 stages respectively. Applying cascading with evidence, we detect frontal and eight kinds of occluded faces in *three times* computing time used in detecting only frontal faces. It detects about 18 320x240 frames per second on a P4 3.06GHz PC. A number of experimental results are reported in Figure 4 to demonstrate the effectiveness of our system.

Fig. 4. (a)-(f) Detection results derived by applying our face detector to some of the CMU+MIT data set. (g)-(l) Detection results on several occluded faces. (m)-(o) Detection results for various exaggerated expressions.

References

1. Bartlett, M.S., Littlewort, G., Fasel, I., Movellan, J.R.: Real time face detection and facial expression recognition: Development and applications to human computer interaction. In: Computer Vision and Pattern Recognition HCI Workshop, Madison, Wisconsin, USA (2003)
2. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40** (2000) 139–157
3. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proc. European Conf. Computational Learning Theory, Barcelona, Spain (1995) 23–37
4. Kearns, M., Valiant, L.G.: Learning boolean formulae or finite automata is as hard as factoring. Technical report, Harvard University Aiken Computation Laboratory (1988)
5. Lienhart, R., Maydt, J.: An extended set of Haar-like features for rapid object detection. In: Proc. Int'l Conf. Image Processing. Volume 1., Rochester, NY, USA (2002) 900–903
6. Liu, C., Shum, H.: Kullback-Leibler boosting. In: Proc. Conf. Computer Vision and Pattern Recognition. Volume 1., Madison, Wisconsin, USA (2003) 587–594
7. Li, S., Zhu, L., Zhang, Z., Blake, A., Zhang, H., Shum, H.: Statistical learning of multi-view face detection. In: Proc. Seventh European Conf. Computer Vision. Volume 4., Copenhagen, Denmark (2002) 67–81
8. Martinez, A., Benavente, R.: The AR face database. Technical report, CVC Technical Report #24 (1998)
9. Moghaddam, B., Pentland, A.P.: Probabilistic visual learning for object representation. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 696–710
10. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: An application to face detection. In: Proc. Conf. Computer Vision and Pattern Recognition, San Jaun, Puerto Rico (1997) 130–136
11. Rätsch, G., Onoda, T., Müller, K.R.: Soft margins for AdaBoost. Machine Learning **42** (2001) 287–320
12. Romdhani, S., Torr, P., Schölkopf, B., Blake, A.: Computationally efficient face detection. In: Proc. Eighth IEEE Int'l Conf. Computer Vision. Volume 2., Vancouver, BC, Canada (2001) 695–700
13. Roth, D., Yang, M., Ahuja, N.: A SNoW-based face detector. In: Advances in Neural Information Processing Systems, Denver, CO, USA (2000) 855–861
14. Schapire, R.E., Singer, Y.: Improved boosting using confidence-rated predictions. Machine Learning **37** (1999) 297–336
15. Schapire, R.E.: The strength of weak learnability. Machine Learning **5** (1990) 197–227
16. Sung, K.K., Poggio, T.: Example-based learning for view-based human face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 39–51
17. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proc. Conf. Computer Vision and Pattern Recognition. Volume 1., Kauai, HI, USA (2001) 511–518
18. Yang, M.H., Abuja, N., Kriegman, D.: Face detection using mixtures of linear subspaces. In: Proc. Int'l Conf. Automatic Face and Gesture Recognition, Grenoble, France (2000) 70–76