Statistical Learning of Multi-view Face Detection

Stan Z. Li[1], Long Zhu[1], ZhenQiu Zhang[2]⋆, Andrew Blake[3]
HongJiang Zhang[1], Harry Shum[1]

[1] Microsoft Research Aisa, Beijing, China
[2] Institute of Automation, Chinese Academy Sinica, Beijing, China
[3] Microsoft Research Cambridge, Cambradge, UK
Contact: szli@microsoft.com, http://research.microsoft.com/∼szli

Abstract. A new boosting algorithm, called FloatBoost, is proposed to overcome the monotonicity problem of the sequential AdaBoost learning. AdaBoost [1, 2] is a sequential forward search procedure using the greedy selection strategy. The premise offered by the sequential procedure can be broken-down when the monotonicity assumption, *i.e.* that when adding a new feature to the current set, the value of the performance criterion does not decrease, is violated. FloatBoost incorporates the idea of Floating Search [3] into AdaBoost to solve the non-monotonicity problem encountered in the sequential search of AdaBoost.
We then present a system which learns to detect multi-view faces using FloatBoost. The system uses a coarse-to-fine, simple-to-complex architecture called detector-pyramid. FloatBoost learns the component detectors in the pyramid and yields similar or higher classification accuracy than AdaBoost with a smaller number of weak classifiers. This work leads to the first real-time multi-view face detection system in the world. It runs at 200 ms per image of size 320x240 pixels on a Pentium-III CPU of 700 MHz. A live demo will be shown at the conference.

1 Introduction

Pattern recognition problems has two essential issues: (i) feature selection, and (ii) classifier design based on selected features. Boosting is a method which attempts to boost the accuracy of an ensemble of weak classifiers to a strong one. The AdaBoost algorithm [1] solved many of the practical difficulties of earlier boosting algorithms. Each weak classifier is trained one stage-wise to minimize the empirical error in a given distribution re-weighted according classification errors of the previously trained classifier. It is shown that AdaBoost is a sequential forward search procedure using the greedy selection strategy to minimize a certain margin on the training set [4].

A crucial heuristic assumption made in such a sequential forward search procedure is the monotonicity, *i.e.* that when adding a new weak classifier to the

⋆ The work presented in this paper was carried out at Microsoft Research Asia.

current set, the value of the performance criterion does not decrease. The premise offered by the sequential procedure can be broken-down when the assumption is violated, *i.e.* when the performance criterion function is non-monotonic. This is the first topic to be dealt with in this paper.

Floating Search [3] is a sequential feature selection procedure with backtracking, aimed to deal with non-monotonic criterion functions for feature selection. A straight sequential selection method like sequential forward search (SFS) or sequential backward search (SBS) adds or deletes one feature at a time. To make this work well, the monotonicity property has to be satisfied by the performance criterion function. Feature selection with a non-monotonic criterion may be dealt with by using a more sophisticated technique, called plus-ℓ-minus-r, which adds or deletes ℓ features and then backtracks r steps [5, 6].

The Sequential Floating Search methods [3] allows the number of backtracking steps to be controlled instead of being fixed beforehand. Specifically, it adds or deletes $\ell = 1$ feature and then backtracks r steps where r depends on the current situation. It is such a flexibility that amends limitations due to the non-monotonicity problem. Improvement on the quality of selected features is gained with the cost of increased computation due to the extended search. The SFFS algorithm performs very well in several applications [3, 7]. The idea of Floating Search is further developed in [8] by allowing more flexibility for the determination of ℓ.

The second topic is an application of booting learning in face detection. Learning based methods have so far been the most effective ones for face detection, *e.g.* [9–12]. There, face detection is treated as an intrinsically two-dimensional (2-D) problem. Taking advantage of the fact that faces are highly correlated, it is assumed that human faces can be described by some low dimensional features which may be derived from a set of prototype face images. Large amount of variation and complexity brought about by changes in facial appearance, lighting and expression makes the face manifold highly complex [13, 14]. Changes in facial view (head pose) further complicate the situation. From pattern recognition viewpoint, two issues are essential in face detection: (i) feature selection, and (ii) classifier design based on selected features.

Applied to face detection [15], AdaBoost is adapted to solving the following three fundamental problems in one boosting procedure: (1) learning incrementally crucial features from a large feature set, (2) constructing weak classifiers each of which is based on one of the selected features, and (3) boosting the weak classifiers into a stronger classifier using a linear combination derived during the learning process. The work of Viola and Jones results in the first real-time frontal face detection system which runs at about 14 frame per second for a 320x240 image [15]. This work, like [9–12], deals with frontal faces only.

However, ability to deal with non-frontal faces is important for many real applications because statistics show that approximately 75% of the faces in home photos are non-frontal [16]. A reasonable treatment for multi-view is the view-based method [17], in which several face models are built, each describing faces in a certain view. This way, explicit 3D modeling is avoided. Feraud *et al.* [18]

adopt the view-based representation for face detection, and use an array of 5 detectors with each detector responsible for one view. Wiskott *et al.* [19] build elastic bunch graph templates for multi-view face detection and recognition. Gong and colleagues [20] study the trajectories of faces in linear PCA feature spaces as they rotate, and use kernel support vector machines (SVMs) for multi-pose face detection and pose estimation [21, 22]. Huang *et al.* [23] use SVM's to estimate facial poses.

The system of Schneiderman and Kanade [24] is claimed to be the first algorithm in the world for (non-real-time) multi-view face detection. Multi-resolution information is used for different levels of wavelet transform. The algorithm consists of an array of 5 face detectors in the view-based framework. Each is constructed using statistics of products of histograms computed from examples of the respective view. However, it is slow and takes 1 min to work on a 320x240 image over only 4 octaves of candidate size [24].

In this paper, we propose a new boosting algorithm, called FloatBoost, for effective statistical learning. In this work, face detection is posed as a problem of classifying each scanned sub-window as face or nonface and such a classifier is trained using an AdaBoost learning algorithm, following the earlier works of [24, 15]. AdaBoost [1, 2] is a sequential forward search procedure using the greedy selection strategy. Its heuristic assumption is the monotonicity. The premise offered by the sequential procedure can be broken-down when the assumption is violated. FloatBoost incorporates the idea of Floating Search [3] into AdaBoost (the real version of AdaBoost presented in [2, 25] is in this work) to solve the non-monotonicity problem encountered in the sequential algorithm of AdaBoost.

We then present an application of FLoatBoost in a learning-based system for real-time multi-view face detection. The system uses a coarse-to-fine, simple-to-complex detector-pyramid architecture. Coarse-to-fine [26, 27] refers to the strategy for view space partition in the pyramid hierarchy from the top (input) to the bottom (output), which deals with changes in facial view. Simple-to-complex refers to the complexities of face detectors, which enables the efficiency needed for detection of a small number of faces from a vast number of candidate sub-windows. These go beyond straightforward view-based methods. This work leads to the first real-time multi-view face detection system in the world. It runs at 200 ms per image of size 320x240 pixels on a Pentium-III CPU of 700 MHz.

The rest of the paper is organized as follows: Section 2 presents the FloatBoost learning methods. Section 3 describes the multi-view face detection system. Section 4 provides experimental results.

2 FloatBoost Learning

Multi-view face detection can be done in three steps: First, scan $\mathcal{I}$ exhaustively at all possible locations (u, v) and scales s, resulting in a large number of sub-windows $x = x(u, v, s \mid \mathcal{I})$. Second, test for each x if it is a face at pose θ

$$H^{\theta}(x) \begin{array}{l} \geq 0 \Rightarrow x \text{ is a pattern of face at pose } \theta \\ < 0 \Rightarrow x \text{ is a nonface pattern} \end{array} \tag{1}$$

Third, post-process to merge multiple detects.

In this section, we describe boost based learning methods for constructing face/nonface classifiers, and propose a new boosting algorithm which improves boosting learning. Here, we consider face-nonface classification only and drop the pose notation θ. Multi-view face detection will be tackled in the next section.

2.1 AdaBoost Learning

For two class problems, we are given a set of N labelled training examples $(x_1, y_1), \ldots, (x_N, y_N)$, where $y_i \in \{+1, -1\}$ is the class label associated with example x_i. For face detection, x_i is an image sub-window of a fixed size (*e.g.* 20x20) containing an instance of the face ($y_i = +1$) or nonface ($y_i = -1$) pattern. In the notion of RealBoost (a real version of AdaBoost [2, 25], see Fig.1, as opposed to the original discrete one [1]), a stronger classifier is a linear combination of M weak classifiers

$$H_M(x) = \sum_{m=1}^{M} h_m(x) \tag{2}$$

where $h_m(x) \in \mathbb{R}$ are weak classifiers. The class label for a test x is obtained as $H(x) = \text{sign}[H_M(x)]$ (an error occurs when $H(x) \neq y$) while the magnitude $|H_M(x)|$ indicates the confidence.

In boosting learning [1, 2, 25], each example x_i is associated with a weight w_i, and the weights are updated dynamically using a multiplicative rule according to the errors in previous learning so that more emphasis is placed on those examples which are erroneously classified by the weak classifiers learned previously. This way, the new weak classifiers will pay more attention to those examples. The stronger classifier is obtained as a proper linear combination of the weak classifiers.

The "margin" of an example (x, y) achieved by $H(x)$ (a single or a combination of weak classifiers) on the training examples can be defined as $yH(x)$ [4]. This can be considered as a measure of the confidence of the h's prediction. The following criterion measures the bound on classification error [2]

$$J(H(x)) = E_w(\mathrm{e}^{-yH(x)}) = \sum_i \mathrm{e}^{-y_i H(x_i)} \tag{3}$$

where $E_w()$ stands for the mathematical expectation with respect to w over the examples (x_i, y_i).

AdaBoost construct $h(x)$ by stage-wise minimization of Eq.(3). Given the current $H_{M-1}(x) = \sum_{m=1}^{M-1} h_m(x)$, the best $h_M(x)$ for the new strong classifier $H_M(x) = H_{M-1}(x) + h_M(x)$ is the one which leads to the minimum cost

$$h_M = \arg\min_{h^\dagger} J(H_{M-1}(x) + h^\dagger(x)) \tag{4}$$

It is shown in [2, 25] that the minimizer is

$$h_M(x) = \frac{1}{2} \log \frac{P(y = +1 \mid x, w^{(M-1)})}{P(y = -1 \mid x, w^{(M-1)})} \tag{5}$$

```
0. (Input)
    (1) Training examples $\mathcal{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
      where $N = a + b$; of which $a$ examples have $y_i = +1$
      and $b$ examples have $y_i = -1$;
    (2) The number $M$ of weak classifiers to be combined;
1. (Initialization)
    $w_i^{(0)} = \frac{1}{2a}$ for those examples with $y_i = +1$ or
    $w_i^{(0)} = \frac{1}{2b}$ for those examples with $y_i = -1$.
2. (Forward Inclusion)
  For $m = 1, \ldots, M$:
    (1) Choose $h_m$ according to Eq.5;
    (2) Update $w_i^{(m)} \leftarrow w_i^{(m)} \exp[-y_i h_m(x_i)]$, and
      normalize to $\sum_i w_i^{(m)} = 1$;
3. (Output)
    $H(x) = \text{sign}[\sum_{m=1}^{M} h_m(x)]$.
```

Fig. 1. RealBoost Algorithm.

where $w^{(M-1)}$ are the weights given at time M. Using $P(y \mid x, w) = P(x \mid y, w)P(y)$ and letting

$$L_M(x) = \frac{1}{2} \log \frac{P(x \mid y = +1, w)}{P(x \mid y = -1, w)} \tag{6}$$

$$T = \frac{1}{2} \left[\log \frac{P(y = +1)}{P(y = -1)} \right] \tag{7}$$

we arrive

$$h_M(x) = L_M(x) - T \tag{8}$$

The half log likelihood ratio $L(x)$ is learned from the training examples of the two classes, and the threshold T can be adjusted to control the balance between the detection and false alarm rates in the case when the prior probabilities are not known.

2.2 FloatBoost Learning

AdaBoost [1, 2] is a sequential forward search procedure using the greedy selection strategy. Its heuristic assumption is the monotonicity. The premise offered by the sequential procedure can be broken-down when the assumption is violated. FloatBoost incorporates the idea of Floating Search [3] into AdaBoost [1, 2, 25] to overcome the non-monotocity problems associated with AdaBoost. The Sequential Floating Search (SFS) method [3] allows the number of backtracking steps to be controlled instead of being fixed beforehand. Specifically, it adds or deletes $\ell = 1$ feature and then backtracks r steps where r depends on the current situation. It is such a flexibility that amends limitations due to the non-monotonicity problem. Improvement on the quality of selected features is gained with the cost of increased computation due to the extended search. The SFS algorithm performs very well in several applications [3, 7]. The idea of Floating Search is further developed in [8] by allowing more flexibility for the determination of ℓ.

These feature selection methods, however, do not address the problem of (sub-)optimal classifier design based on the selected features. FloatBoost combines them into AdaBoost for both effective feature selection and classifier design.

```
0. (Input)
    (1) Training examples $\mathcal{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
        where $N = a + b$; of which $a$ examples have
        $y_i = +1$ and $b$ examples have $y_i = -1$;
    (2) The maximum number $M_{\max}$ of weak classifiers;
    (3) The cost function $J(H_M)$ (e.g., error rate made by $H_M$), and
        the maximum acceptable cost $J^*$.
1. (Initialization)
    (1) $w_i^{(0)} = \frac{1}{2a}$ for those examples with $y_i = +1$ or
        $w_i^{(0)} = \frac{1}{2b}$ for those examples with $y_i = -1$;
    (2) $J_m^{\min}$ =max-value (for $m = 1, \ldots, M_{\max}$),
        $M = 0$, $\mathcal{H}_0 = \{\}$.
2. (Forward Inclusion)
    (1) $M \leftarrow M + 1$;
    (2) Choose $h_M$ according to Eq.4;
    (3) Update $w_i^{(M)} \leftarrow w_i^{(M-1)} \exp[-y_i h_M(x_i)]$,
        normalize to $\sum_i w_i^{(M)} = 1$;
    (4) $\mathcal{H}_M = \mathcal{H}_{M-1} \cup \{h_M\}$;
        If $J_M^{\min} > J(H_M)$, then $J_M^{\min} = J(H_M)$;
3. (Conditional Exclusion)
    (1) $h' = \arg\min_{h \in \mathcal{H}_M} J(H_M - h)$;
    (2) If $J(H_M - h') < J_{M-1}^{\min}$, then
        (a) $\mathcal{H}_{M-1} = \mathcal{H}_M - h'$;
            $J_{M-1}^{\min} = J(H_M - h')$; $M = M - 1$;
        (b) if $h' = h_{m'}$, then
            re-calculate $w_i^{(j)}$ and $h_j$ for $j = m', \ldots, M$;
        (c) goto 3.(1);
    (3) else
        (a) if $M = M_{\max}$ or $J(\mathcal{H}_M) < J^*$, then goto 4;
        (b) goto 2.(1);
4. (Output)
    $H(x) = \mathrm{sign}[\sum_{m=1}^{M} h_m(x)]$.
```

Fig. 2. FloatBoost Algorithm.

Let $\mathcal{H}_M = \{h_1, \ldots, h_M\}$ be the so-far-best set of M weak classifiers; $J(H_M)$ be the criterion which measures the overall cost of the classification function $H_M(x) = \sum_{m=1}^{M} h_m(x)$ build on $\mathcal{H}_M$; $J_m^{\min}$ be the minimum cost achieved so far with a linear combination of m weak classifiers for $m = 1, \ldots, M_{\max}$ (which are initially set to a large value before the iteration starts). As shown in Fig.2, the FloatBoost Learning procedure involves training inputs, initialization, forward inclusion, conditional exclusion and output.

In Step 2 (forward inclusion), the currently most significant weak classifier is added one at a time, which is the same as in AdaBoost. In Step 3 (conditional exclusion), FloatBoost removes the least significant weak classifier from $\mathcal{H}_M$, subject to the condition that the removal leads to a lower cost than $J_{M-1}^{\min}$. Supposing that the removed weak classifier was the m'-th in $\mathcal{H}_M$, then $h_{m'}, \ldots, h_M$ will be re-learned. These are repeated until no more removals can be done.

For face detection, the acceptable cost J^* is the maximum allowable risk, which can be defined as a weighted sum of missing rate and false alarm rate. The algorithm terminates when the cost is below J^* or the maximum number M of weak classifiers is reached.

FloatBoost usually needs fewer weak classifiers than AdaBoost to achieve a given objective J^*. One have two options with such a result: (1) Use the FloatBoost-trained strong classifier with its fewer weak classifiers to achieve similar performance as can be done by a AdaBoost-trained classifier with more weak classifiers. (2) Continue FloatBoost learning to add more weak classifiers even if the performance on the training data does not increase. The reason for (2) is that even if the performance does not improve on the training data, adding more weak classifiers may lead to improvements on test data [4].

3 Multi-view Face Detection System

The multi-view face detection task is the following: Given the input image, sub-windows at all locations and scales are scanned. Face detection is to classify each sub-window into face or nonface. Multi-view face detection should be able to detect non-frontal faces. Adopting a coarse-to-fine view-partition strategy, the detector-pyramid architecture consists of several levels from the coarse top level to the fine bottom level.

3.1 Dealing with Head Rotations

Our system deals with three types of head rotations which currently are in the following ranges: (1) out-of-plane rotations in the range of $\Theta = [-90°, +90°]$, (2) in-plane rotations in the range of $\Phi = [-45°, +45°]$, and (3) a moderate amount of up-and-down nodding rotations. We adopt the view-based approach. A detector-pyramid is constructed to detect the presence of up-right faces, subject to out-of-plane rotations in Θ and in-plane rotations in $[-15°, +15°]$. The design of such a detector-pyramid will be described shortly. In-plane rotations are handled as follows: (1) Divide Φ into three sub-ranges $\Phi_1 = [-45°, -15°]$,

$\Phi_2 = [-15°, +15°]$, and $\Phi_3 = [+15°, +45°]$ (*cf.* Fig.3). (2) Apply the detector-pyramid on two images in-plane-rotated by $\pm 30°$ as well on the original image. This will effectively cover in-plane-rotations in $[-45°, +45°]$. The up-and-down nodding rotations are dealt with by tolerances of the face detectors to them.

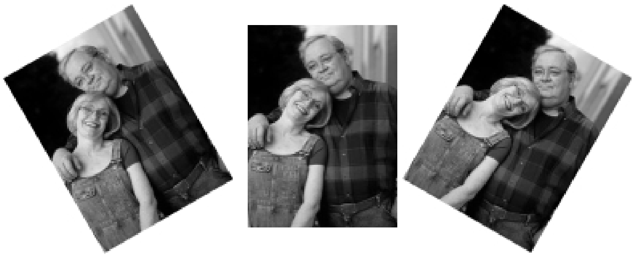

Fig. 3. Middle: An image containing frontal faces subject to in-plane rotations. Left and right: In-plane rotated by $\pm 30°$.

3.2 Detector-Pyramid

The design of the detector-pyramid adopts the coarse-to-fine and simple-to-complex (top-down in the pyramid) strategy [26, 27]. The architecture is illustrated in Fig.4. This architecture design is for the detection of faces subject to out-of-plane rotations in $\Theta = [-90°, +90°]$ and in-plane rotations in $\Phi_2 = [-15°, +15°]$. The full in-plane rotations in $\Phi = [-45°, +45°]$ is dealt with by applying the detector-pyramid on the images rotated $\pm 30°$, as mentioned earlier.

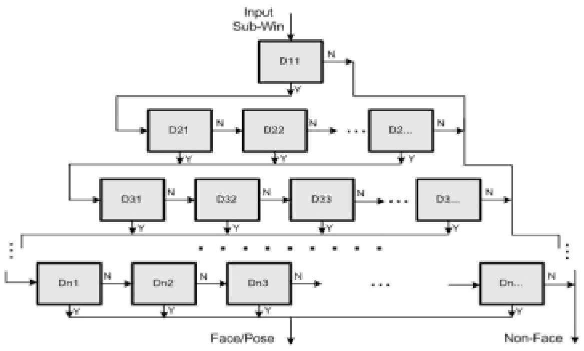

Fig. 4. Detector-pyramid for multi-view face detection.

Coarse-to-fine The full range Θ of out-of-plane rotations is partitioned into increasingly narrower ranges, and thereby the whole face space is divided into increasingly smaller subspaces. Our current implementation of the detector-pyramid consists of 3 levels. The partitions of the out-of-plane rotation for the 3 levels is illustrated in Fig.5. Faces detected by the 7 channels at the bottom level of the detector-pyramid need to be merged to give the final result.

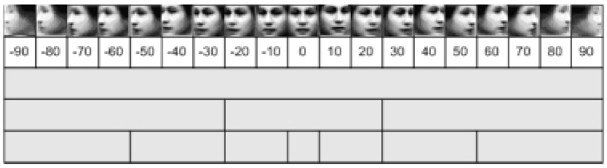

Fig. 5. Out-of-plane view partition. Out-of-plane head rotations (row 1), the facial view labels (row 2), and the coarse-to-fine view partitions at the three levels of the detector-pyramid (rows 3-5).

Although there are no overlaps between the partitioned view sub-ranges at each level, a face detector trained for one view may detect faces of its neighboring views. Therefore, faces detected by the 7 channels at the bottom level of the detector-pyramid need to be merged to give the final result. This is schematically illustrated in Fig.6.

Simple-to-complex A vast number of sub-windows result from the scan of the input image. For the purpose of efficiency, it is crucial to discard as many as possible nonface sub-windows at the earliest possible stage so that as few as possible sub-windows will be processed further by later stages. Therefore, the detectors in the early stages are simpler so as to reject a vast number of nonface sub-windows more quickly with little computation, whereas those in the later stage are more complex and spend more time.

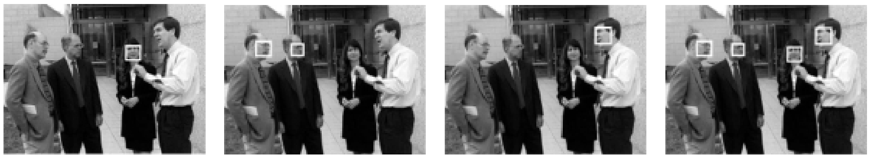

Fig. 6. Schematic illustration of merge from different channels. From left to right: Outputs of fontal, left, right view channels, and the final result after merge.

3.3 Learning Weak Classifiers for Face Detection

Each weak classifier is constructed based on a simple feature, denoted feature $x_{(j)}$ (note this notation differs from x_i, the latter being for training example i), derived from the sub-window x. Three basic types of simple features are used in this work as shown in Fig.7. These block differences are steerable filters similar to those used in [28, 15], but are more general in that these features can be non-symmetrical to cater to non-symmetrical characteristics of non-frontal faces. Each such feature has a scalar value which can be computed very efficiently [29] from the summed-area table [30] or integral image [15]. There are a total number of 102,979 two-block features for a sub-window of size 20x20 pixels. There are a total number of 188,366 three-block features (with some restrict to their freedom).

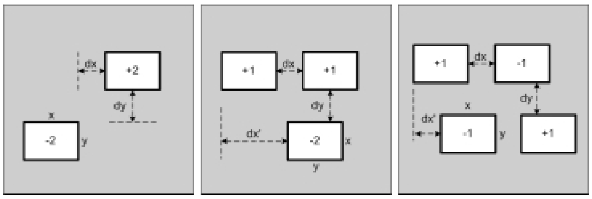

Fig. 7. The three types of simple Harr wavelet like features $x_{(j)}$ defined on a sub-window x. The rectangles are of size $x \times y$ and are at distances of (dx, dy) apart. Each feature takes a value calculated by the weighted $(\pm 1, 2)$ sum of the pixels in the rectangles.

Because we know the form of the optimal weak classifier as Eq.(6), we design a set of candidate weak classifiers as follows, given that $M-1$ features have been selected and the corresponding $M-1$ weak classifiers have been learned: First, we approximate $p(x \mid y, w)$ by using the distributions of the M features selected so far by

$$\begin{aligned} p(x \mid y) &\approx p(x_{(1)}, x_{(2)}, \ldots, x_{(M)} \mid y) && (9) \\ &= p(x_{(1)} \mid y)\, p(x_{(2)} \mid y, x_{(1)}) \cdots \\ &\quad p(x_{(M-1)} \mid y, x_{(1)}, \ldots, x_{(M-2)}) \\ &\quad p(x_{(M)} \mid y, x_{(1)}, \ldots, x_{(M-1)})) && (10) \end{aligned}$$

Note that $p(x_{(m)} \mid y, x_{(1)}, \ldots, x_{(m-1)})$ is actually $p(x_{(m)} \mid y, w^{(m-1)})$ because $w^{(m)}$ contains the information about entire history of w due to the multiplicative rule and accounts for the dependencies on $x_{(1)}, \ldots, x_{(m-1)}$. Therefore, we have

$$\begin{aligned} p(x \mid y) \approx\; & p(x_{(1)} \mid y, w^{(0)})\, p(x_{(2)} \mid y, w^{(1)}) \cdots && (11) \\ & p(x_{(M-1)} \mid y, w^{(M-2)}) p(x_{(M)} \mid y, w^{(M-1)}) \end{aligned}$$

While $p(x \mid y) \approx p(x_{(1)} \mid y) p(x_{(2)} \mid y), \ldots, p(x_{(M)} \mid y)$ is simply assumed in [24], our assumption of the conditional independence in the above equation may be more justifiable.

Then denote the conditional probability densities of feature $x_{(j)}$ of sub-window x by $p_j(x_{(j)} \mid y, w^{(M-1)})$ with $y = +1$ for the face pattern and $y = -1$ for the non-face pattern. The two densities are estimated using the histograms resulting from weighted voting of the training examples. Let $L_{(j)}^{(M-1)}(x) = \frac{p_j(x_{(j)} \mid y=+1, w^{(M-1)})}{p_j(x_{(j)} \mid y=-1, w^{(M-1)})}$, and $h_{(j)}^{(M-1)}(x) = L_{(j)}^{(M-1)}(x) - T$. The set of candidate weaker classifiers is designed as

$$\mathcal{H}^{(M-1)} = \{h_{(j)}^{(M-1)}(x) \mid \forall j\} \tag{12}$$

Now, the best $h_M(x)$ among all in $\mathcal{H}_{(M-1)}$ for the new strong classifier $H_M(x) = H_{M-1}(x) + h_M(x)$ is given by Eq.(4) among all $h^\dagger \in \mathcal{H}^{(M-1)}$.

3.4 Summary of the System

To summarize the above, the construction of the detector-pyramid is done as features $\rightarrow$ weak classifiers $\rightarrow$ strong classifier $\rightarrow$ detectors $\rightarrow$ pyramid level $\rightarrow$ pyramid as follows:

1. Simple *features* are designed. There are a number of candidate features.
2. A subset of them are selected and the corresponding *weak classifiers* are learned using FloatBoost.
3. The *strong classifier* is constructed as a linear combination of the weak ones, as the output of FloatBoost learning.
4. A *detector* is composed of one, or a cascade of strong classifiers.
5. At each *level* of the pyramid, the full range of out-of-plane rotation is partitioned into a number of sub-ranges, and the same number of detector are trained for face detection in that partition, each specialized for a certain view sub-range.
6. Finally, the *detector-pyramid* is composed of several levels from the coarsest view partition at the top to the finest partition at the bottom.

4 Experimental Results

4.1 Frontal Face Detection

About 3000 face examples are collected from various sources, covering the out-of-plane rotation in the range of $[-20°, +20°]$ of out-of-plane rotations. They are roughly aligned by eyes and mouth. For each aligned face example, a synthesized face example is generated by a random in-plane-rotation in the range of $[-15°, +15°]$. This creates a training set of 6,000 face examples. The 6,000 images are then cropped and re-scaled to the size of 20x20. Sufficient nonface examples are collected from 100,000 images containing no faces. The ROC curve for the training set is shown in Fig.8.

The MIT+CMU test set composed of 125 images containing 481 faces, which is used in [9], is used for test the performance. Our Floatboost (FB) algorithm is compared with AB (20) (AdaBoost of viola-Jones as implemented by ourselves using training examples of size 20x20), AB (24) (AdaBoost with training examples of size 24x24 [15]), and CMU-NN of Rowley *et al.* [9]. The results are

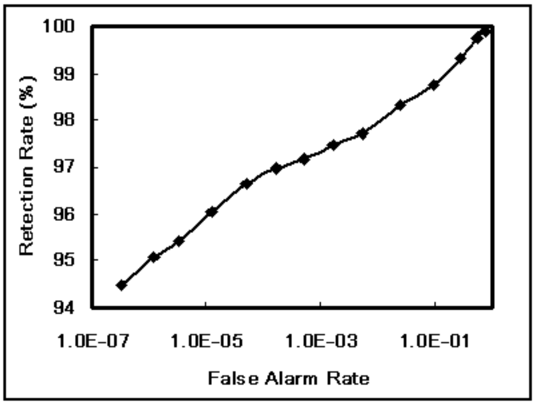

Fig. 8. ROC curve of FloatBoost method for the frontal face training set.

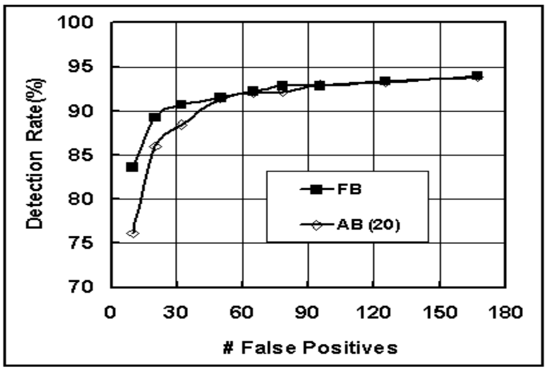

Fig. 9. Comparison of ROC curves of FloatBoost and Viola-Jones (20x20) methods on the MIT+CMU test set.

shown in Table 1, where "N.Fea" stands for number of features and "N.FA" for number of false alarms. Fig.9 shows ROC curves of FB and AB (20). Our algorithm using FloatBoost learned a total of 2546 features (weak classifiers) from the 20x20 training examples to achieve the performance. This is about 2/3 of 3872 computed the AdaBoost counterpart from the 20x20 training examples. The reader is also referred to [31] for a more comprehensive comparison with other systems.

4.2 Multi-view Face Detection

The training set for multi-view face detection is created in the similar way to that for the frontal faces, except that the out-of-plane rotation covers the full range of $[-90°, +90°]$. The CMU profile face set [24] is used to test the algorithm. This test set consists of 208 images with 441 faces of which 347 were

	FB	AB (20)	AB (24)	CMU-NN
N.Fea	2546	3872	6061	N/A
N.FA=10	83.6%	82.7%	76.1%	83.2%
N.FA=31	90.2%	89.2%	88.4%	86%

Table 1. Comparison of FloatBoost, Viola-Jones AdaBoost, and CMU-NN methods on the MIT+CMU test set.

profile views, which are not restricted in terms of subject matter or background scenery. They were collected from various news web sites. The database can be downloaded at http://vasc.ri.cmu.edu/idb/html/face/profile_images /index.html. Some results are shown in Fig.10. We also provide a video clip showing multi-view face detection at http://research.microsoft.com/∼szli/Demos/MV-FaceDet.html.

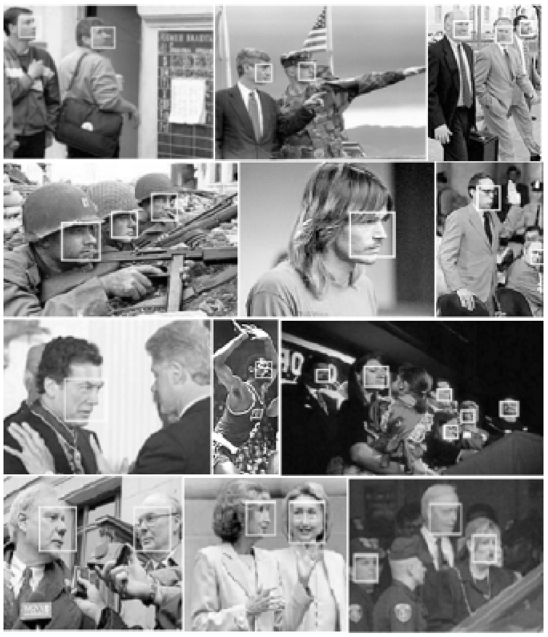

Fig. 10. Some multi-view face detection results.

5 Conclusion and Future Work

The coarse-to-fine and simple-to-complex detector-pyramid leads to the first real-time multi-view face detection system. By incorporating the idea of Floating Search [3] into AdaBoost [1, 2], FloatBoost effectively improves the learning results. It needs fewer weaker classifiers than AdaBoost to achieve similar or higher performance. Given that this framework demonstrates good performance in multi-view face detection, we stress that the underlying architecture is fairly general and can be applied to other appearance based object detection problems as well.

In the current version, the forward step is unconditional. This strategy may cause some problem due to the local minimum problem [8]. In Active Floating Search [8], the forward step is made conditional too and this may give a better solution.

The Boosting algorithm may need substantial computation for training. Several methods can be used to make the training more efficient with little drop in the training performance. Noticing that only examples with large weigh values are influential, Friedman *et al.* [25] propose to select examples with large weights, *i.e.* those which in the past have been wrongly classified by the learned weak classifiers, for the training weak classifier in the next round. Top examples within a fraction of $1 - \beta$ of the total weight mass are used, where $\beta \in [0.01, 0.1]$. Fan *et al.* [32] reduces samples size by random sampling of the training set problem. Two sampling schemes are adopted: r-sampling (uniform sampling each round), d-sampling (disjoint subsets).

References

1. Freund, Y., Schapire, R.: "A decision-theoretic generalization of on-line learning and an application to boosting". Journal of Computer and System Sciences **55** (1997) 119–139
2. Schapire, R.E., Singer, Y.: "Improved boosting algorithms using confidence-rated predictions". In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory. (1998) 80–91
3. Pudil, P., Novovicova, J., Kittler, J.: Floating search methods in feature selection. Pattern Recognition Letters **15** (1994) 1119–1125
4. Schapire, R., Freund, Y., Bartlett, P., Lee, W.S.: "Boosting the margin: A new explanation for the effectiveness of voting methods". The Annals of Statistics **26** (1998) 1651–1686
5. Stearns, S.D.: "On selecting features for pattern classifiers". In: Proceedings of International Conference Pattern Recognition. (1976) 71–75
6. Kittler, J.: "Feature set search algorithm". In Chen, C.H., ed.: Pattern Recognition in Practice. NorthHolland, Sijthoff and Noordhoof (1980) 41–60
7. Jain, A., Zongker, D.: Feature selection: evaluation, application, and samll sample performance. IEEE Trans. on PAMI **19** (1997) 153–158
8. Somol, P., Pudil, P., Novoviova, J., Paclik, P.: "Adaptive floating search methods in feature selection". Pattern Recognition Letters **20** (1999) 1157–1163
9. Rowley, H.A., Baluja, S., Kanade, T.: "Neural network-based face detection". IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998) 23–28
10. 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
11. Osuna, E., Freund, R., Girosi, F.: "Training support vector machines: An application to face detection". In: CVPR. (1997) 130–136
12. Roth, D., Yang, M., Ahuja, N.: "A snow-based face detector". In: Proceedings of Neural Information Processing Systems. (2000)

13. Bichsel, M., Pentland, A.P.: "Human face recognition and the face image set's topology". CVGIP: Image Understanding **59** (1994) 254–261
14. Simard, P.Y., Cun, Y.A.L., Denker, J.S., Victorri, B.: "Transformation invariance in pattern recognition - tangent distance and tangent propagation". In Orr, G.B., Muller, K.R., eds.: Neural Networks: Tricks of the Trade, Springer (1998)
15. Viola, P., Jones, M.: "Robust real time object detection". In: IEEE ICCV Workshop on Statistical and Computational Theories of Vision, Vancouver, Canada (2001)
16. Kuchinsky, A., Pering, C., Creech, M.L., Freeze, D., Serra, B., Gwizdka, J.: "FotoFile: A consumer multimedia organization and retrieval system". In: Proceedings of ACM SIG CHI'99 Conference, Pittsburg (1999)
17. Pentland, A.P., Moghaddam, B., Starner, T.: "View-based and modular eigenspaces for face recognition". In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (1994) 84–91
18. Feraud, J., Bernier, O., Collobert, M.: "A fast and accurate face detector for indexation of face images". In: Proc. Fourth IEEE Int. Conf on Automatic Face and Gesture Recognition, Grenoble (2000)
19. Wiskott, L., Fellous, J., Kruger, N., malsburg, C.V.: "face recognition by elastic bunch graph matching". IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 775–779
20. Gong, S., McKenna, S., Collins, J.: "An investigation into face pose distribution". In: Proc. IEEE International Conference on Face and Gesture Recognition, Vermont (1996)
21. Ng, J., Gong, S.: "performing multi-view face detection and pose estimation using a composite support vector machine across the view sphere". In: Proc. IEEE International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, Corfu, Greece (1999) 14–21
22. Li, Y.M., Gong, S.G., Liddell, H.: "support vector regression and classification based multi-view face detection and recognition". In: IEEE Int. Conf. Oo Face & Gesture Recognition, France (2000) 300–305
23. Huang, J., Shao, X., Wechsler, H.: "Face pose discrimination using support vector machines (SVM)". In: Proceedings of International Conference Pattern Recognition, Brisbane, Queensland, Australia (1998)
24. Schneiderman, H., Kanade, T.: "A statistical method for 3d object detection applied to faces and cars". In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2000)
25. Friedman, J., Hastie, T., Tibshirani, R.: "Additive logistic regression: a statistical view of boosting". Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity (1998)
26. Amit, Y., , Geman, D., Wilder, K.: "Joint induction of shape features and tree classifiers". IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 1300–1305
27. Fleuret, F., Geman, D.: "Coarse-to-fine face detection". International Journal of Computer Vision **20** (2001) 1157–1163
28. Papageorgiou, C.P., Oren, M., Poggio, T.: "A general framework for object detection". In: Proceedings of IEEE International Conference on Computer Vision, Bombay, India (1998) 555–562
29. Simard, P.Y., Bottou, L., Haffner, P., Cun, Y.L.: "Boxlets: a fast convolution algorithm for signal processing and neural networks". In Kearns, M., Solla, S., Cohn, D., eds.: Advances in Neural Information Processing Systems. Volume 11., MIT Press (1998) 571–577
30. Crow, F.: "Summed-area tables for texture mapping". In: SIGGGRAPH. Volume 18(3). (1984) 207–212
31. Erik Hjelmas, B.K.L.: "Face detection: A survey". Computer Vision and Image Understanding **3** (2001)
32. Fan, W., Stolfo, S., Zhang, J.: "The application of adaboost for distributed, scalable and on-line learning". In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, California (1999)