Chapter 11
Face Detection

Stan Z. Li and Jianxin Wu

11.1 Introduction

Face detection is the first step in automated face recognition. Its reliability has a major influence on the performance and usability of the entire face recognition system. Given a single image or a video, an ideal face detector should be able to identify and locate all the present faces regardless of their position, scale, orientation, age, and expression. Furthermore, the detection should be done irrespectively of extraneous illumination conditions and the image and video content.

Face detection can be performed based on several cues: skin color (for faces in color images and videos), motion (for faces in videos), facial/head shape, facial appearance, or a combination of these parameters. Most successful face detection algorithms are appearance-based without using other cues. The processing is done as follows: An input image is scanned at all possible locations and scales by a subwindow. Face detection is posed as classifying the pattern in the subwindow as either face or nonface. The face/nonface classifier is learned from face and nonface training examples using statistical learning methods.

This chapter presents appearance-based and learning-based methods.[1] It will highlight AdaBoost-based methods because so far they are the most successful ones in terms of detection accuracy and speed. Effective postprocessing methods are also described. Experimental results are provided.

[1]The reader is referred to a review article [50] for other earlier face detection methods.

S.Z. Li (✉)
Center for Biometrics and Security Research & National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
e-mail: szli@cbsr.ia.ac.cn

J. Wu
School of Computer Engineering, Nanyang Technological University, Singapore, Singapore
e-mail: jxwu@ntu.edu.sg

S.Z. Li, A.K. Jain (eds.), *Handbook of Face Recognition*,
DOI 10.1007/978-0-85729-932-1_11,

Fig. 11.1 Face (*top*) and nonface (*bottom*) examples

11.2 Appearance and Learning-Based Approaches

With appearance-based methods, face detection is treated as a problem of classifying each scanned subwindow as one of two classes (that is, face and nonface). Appearance-based methods avoid difficulties in modeling 3D structures of faces by considering possible face appearances under various conditions. A face/nonface classifier may be learned from a training set composed of face examples taken under possible conditions as would be seen in the running stage and nonface examples as well (see Fig. 11.1 for a random sample of 10 face and 10 nonface subwindow images). Building such a classifier is possible because pixels on a face are highly correlated, whereas those in a nonface subwindow present much less regularity.

However, large variations brought about by changes in facial appearance, lighting, and expression make the face manifold or face/nonface boundaries highly complex [4, 37, 40]. Changes in facial view (head pose) further complicate the situation. A nonlinear classifier is needed to deal with the complicated situation. The speed is also an important issue for realtime performance.

Great research effort has been made for constructing complex yet fast classifiers and much progress has been achieved since 1990s. Turk and Pentland [41] describe a detection system based on principal component analysis (PCA) subspace or eigenface representation. Whereas only likelihood in the PCA subspace is considered in the basic PCA method, Moghaddam and Pentland [23] also consider the likelihood in the orthogonal complement subspace; using that system, the likelihood in the image space (the union of the two subspaces) is modeled as the product of the two likelihood estimates, which provide a more accurate likelihood estimate for the detection. Sung and Poggio [38] first partition the image space into several face and nonface clusters and then further decompose each cluster into the PCA and null subspaces. The Bayesian estimation is then applied to obtain useful statistical features. The system of Rowley et al.'s [31] uses retinally connected neural networks. Through a sliding window, the input image is examined after going through an extensive preprocessing stage. Osuna et al. [24] train a nonlinear support vector machine to classify face and nonface patterns, and Yang et al. [51] use the SNoW (Sparse Network of Winnows) learning architecture for face detection. In these systems, a bootstrap algorithm is used iteratively to collect meaningful nonface examples from images that do not contain any faces for retraining the detector. Schneiderman and Kanade [34] use multiresolution information for different levels of wavelet transform. A nonlinear classifier is constructed using statistics of products of histograms computed from face and nonface examples. The system of five

view detectors takes about 1 minute to detect faces for a 320×240 image over only four octaves of candidate size [34]. The speed is later improved in [35] to five seconds for an image of size 240×256 using a Pentium II at 450 MHz.

Recent progresses in face detection mostly are made within the cascade detector framework proposed by Viola and Jones [43, 45], which provides fast and robust face detection system. Three major components contribute to the cascade face detector: an over-complete set of local features that can be evaluated quickly, an AdaBoost based method to build strong nonlinear classifiers from the weak local features, and a cascade detector architecture that leads to realtime detection speed.

An over-complete set of rectangle features, which are simple scalar Haar wavelet-like features, are shown to be effective in distinguishing faces from nonfaces. Viola and Jones make use of several techniques [7, 36] for effective computation of a large number of such features under varying scale and location, which is important for realtime performance. A single Haar-like feature, however, is far from enough to build a powerful nonlinear classifier. The AdaBoost algorithm is used to solve the following three fundamental problems: (1) selecting effective 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 to construct a strong classifier. Moreover, the simple-to-complex cascade of classifiers makes the computation even more efficient, which follows the principles of pattern rejection [3, 8] and coarse-to-fine search [2, 10]. Their system is the first realtime frontal-view face detector, and it runs at about 15 frames per second on a 384×288 image [45].

Various improvements have been proposed for the cascade detector, including reducing the training and testing time, and achieving higher detection accuracies. Extensions of the simple Haar-like feature set have been proposed to introduce more complex local features (for example, in [14, 21, 27]). The original cascade detector takes weeks of training time [45]. More local features lead to higher detection accuracies, but also incur even higher time and storage requirements. A strategy is introduced by Wu et al. in [47] that reduces the training time to a few hours by using a precomputation strategy. The speedup of [47] is achieved by reducing the training time of weak classifiers. An alternative strategy by Pham and Cham [27] uses one dimensional Gaussian distributions to model faces and nonfaces for a single feature, and saves more than half of the training time compared to [47].

Variants of the discrete AdaBoost algorithm used in [45] have been shown to improve the trained nonlinear classifiers, for example, using real-valued variants of AdaBoost [21]. Face detection poses an asymmetric learning problem, because we usually have only thousands of face training examples, but billions of nonfaces. It is important to specifically deal with this asymmetric learning goal. Asymmetric boosting [44] and linear asymmetric classifier (LAC) [47] are two examples that achieve higher detection performances using asymmetric learning methods.

The cascade structure has been altered for faster detection speed. Instead of evaluating all the weak classifiers in a strong classifier, the strong classifier can make a decision prematurely (evaluating only a subset of weak classifiers), for example, in the soft cascade method [5]. This "multi-exit" strategy [28, 49] usually leads to higher detection performance besides reducing testing time.

The ability to deal with nonfrontal faces is important for many real applications because approximately 75% of the faces in home photos are nonfrontal [16]. A reasonable treatment for the multiview face detection problem is the view-based method [26], in which several face models are built, each describing faces in a certain view range. This way, explicit 3D face modeling is avoided. Feraud et al. [9] adopt the view-based representation for face detection and use an array of five detectors, with each detector responsible for one facial view. Wiskott et al. [46] build elastic bunch graph templates for multiview face detection and recognition. Huang et al. [13] use SVMs to estimate the facial pose. The algorithm of Schneiderman and Kanade [34] consists of an array of five face detectors in the view-based framework.

Li et al. [17–19] present a multiview face detection system. A new boosting algorithm, called FloatBoost, is proposed to incorporate Floating Search [29] into AdaBoost (RealBoost). An extended Haar feature set is proposed for dealing with out-of-plane (left-right) rotation. A modified cascade detector (following the coarse-to-fine and simple-to-complex principle) is designed for the fast detection of multiview faces. This work leads to the first realtime multiview face detection system. It runs at 200 ms per image (320×240 pixels) on a Pentium-III CPU of 700 MHz.

Huang et al. [14] presents a similar solution that detects full-range in-plane and out-of-plane rotated faces. The main contributions of [14] include a manually designed new cascade architecture, a new set of local features called granular features, and a new multi-class boosting learning algorithm called Vector Boosting.

Given that the cascade detector based on boosting learning methods has achieved the best performance to date in terms of both accuracy and speed, our presentation in the following sections focuses on this thread of research efforts. Strategies are also described for efficient detection of multiview faces.

11.3 AdaBoost-Based Methods

For AdaBoost learning, a complex nonlinear *strong classifier* $H_M(x)$ is constructed as a linear combination of M simpler, easily constructible *weak classifiers* in the following form [11]

$$H_M(x) = \sum_{m=1}^{M} \alpha_m h_m(x) \tag{11.1}$$

where x is a pattern to be classified, $h_m(x)$ are the M weak classifiers, $\alpha_m \geq 0$ are the combining coefficients in $\mathbb{R}$. In the discrete version, $h_m(x)$ takes a discrete value in $\{-1, +1\}$, whereas in the real-valued version, the output of $h_m(x)$ is a number in $\mathbb{R}$. $H_M(x)$ is real-valued, but the prediction of class label for x is obtained as $\hat{y}(x) = \text{sign}[H_M(x)]$.

The AdaBoost learning procedure is aimed at learning a sequence of best weak classifiers $h_m(x)$ and the best combining weights α_m. A set of N labeled training examples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ is assumed available, where $y_i \in \{+1, -1\}$ is the class label for the example $x_i \in \mathbb{R}^n$. A distribution $[w_1, \ldots, w_N]$ of the training

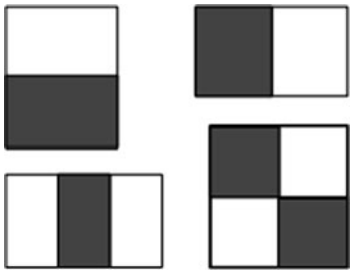

Fig. 11.2 Four types of rectangular Haar wavelet-like features. A feature is a scalar calculated by summing up the pixels in the white region and subtracting those in the dark region

examples, where w_i is associated with a training example (x_i, y_i), is computed and updated during the learning to represent the distribution of the training examples. After iteration m, harder-to-classify examples (x_i, y_i) are given larger weights $w_i^{(m)}$, so that at iteration $m+1$, more emphasis is placed on these examples. AdaBoost assumes that a procedure is available for learning a weak classifier $h_m(x)$ from the training examples, given the distribution $[w_i^{(m)}]$.

In Viola and Jones's face detection work [43, 45],[2] a weak classifier $h_m(x) \in \{-1, +1\}$ is obtained by thresholding on a scalar feature $z_k(x) \in \mathbb{R}$ selected from an overcomplete set of Haar wavelet-like features [25, 39]. In the real-valued versions of AdaBoost, such as RealBoost and LogitBoost, a real-valued weak classifier $h_m(x) \in \mathbb{R}$ can also be constructed from $z_k(x) \in \mathbb{R}$ [19, 21, 33]. The following discusses how to generate candidate weak classifiers.

11.3.1 Local Features

Viola and Jones propose four basic types of scalar features for face detection [25, 45], as shown in Fig. 11.2. Such a block feature is located in a subregion of a subwindow and varies in shape (aspect ratio), size, and location inside the subwindow. For a subwindow of size 20×20, there can be tens of thousands of such features for varying shapes, sizes and locations. Feature k, taking a scalar value $z_k(x) \in \mathbb{R}$, can be considered a transform from the n-dimensional space ($n = 400$ if a face example x is of size 20×20) to the real line. These scalar numbers form an overcomplete feature set for the intrinsically low-dimensional face pattern.

One Haar-like feature can be viewed as a mask consisting of three values: 1 for those pixels in the white region of the feature, -1 for those dark region pixels in the feature, and 0 for those pixels outside of the feature region. The mask is of the same size as the subwindow. If we stack pixels of a subwindow into a vector x, and stack the mask associated with a feature into a vector m, this particular feature will have feature value $m^{\mathrm{T}}x$.

[2] Viola and Jones [43, 45] used $h_m(x) \in \{0, 1\}$. Our notation is slightly different from but equivalent to theirs.

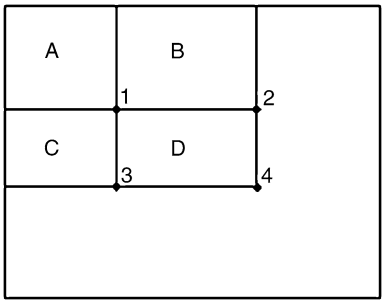

Fig. 11.3 The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $(4 + 1) - (2 + 3)$. From Viola and Jones [43], © 2001 IEEE, with permission

These Haar-like features are interesting for two reasons: (1) powerful face/non-face classifiers can be constructed based on these features (see later); and (2) they can be computed efficiently [36] using the summed-area table [7] or integral image [43] technique.

The integral image $\mathrm{II}(x, y)$ at location x, y contains the sum of the pixels above and to the left of x, y, defined as [43]

$$\mathrm{II}(x, y) = \sum_{x' \le x, y' \le y} \mathrm{I}(x, y). \tag{11.2}$$

The image can be computed in one pass over the original image using the following pair of recurrences

$$S(x, y) = S(x, y - 1) + \mathrm{I}(x, y), \tag{11.3}$$

$$\mathrm{II}(x, y) = \mathrm{II}(x - 1, y) + S(x, y), \tag{11.4}$$

where $S(x, y)$ is the cumulative row sum, $S(x, -1) = 0$ and $\mathrm{II}(-1, y) = 0$. Using the integral image, any rectangular sum can be computed in four array references, as illustrated in Fig. 11.3. The use of integral images leads to enormous savings in computation for features at varying locations and scales.

With the integral images, the intensity variation within a rectangle D of any size and any location can be computed efficiently; for example $V_D = \sqrt{V * V}$ where $V = (4 + 1) - (2 + 3)$ is the sum within D, and a simple intensity normalization can be done by dividing all the pixel values in the subwindow by the variation.

Equation (11.4) shows that the feature value $m^{\mathrm{T}}x$ can be evaluated extremely fast when the rectangular structures in the mask m is utilized. Recently, extended sets of Haar-like features have been proposed for improving detection accuracy [27], dealing with out-of-plane head rotation [14, 19] and for in-plane head rotation [14, 21]. The extended features are carefully designed such that special structures exist in their corresponding masks, thus the feature values can be computed quickly using ideas similar to (11.4).

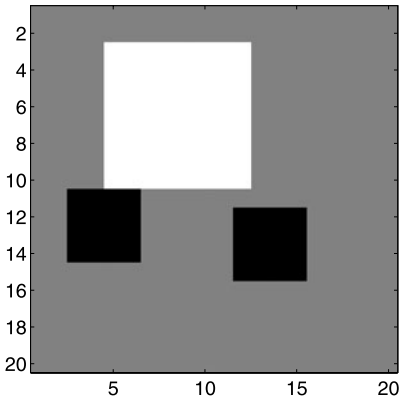

Fig. 11.4 One example granular feature

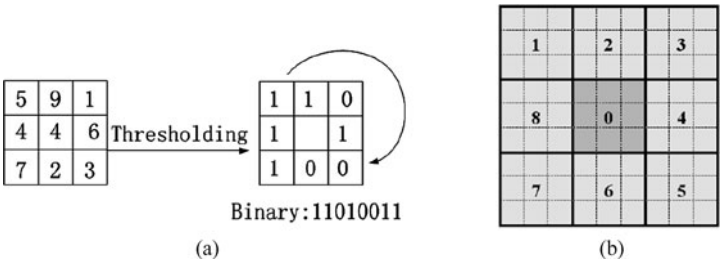

Fig. 11.5 Illustration of LBP (**a**) and MB-LBP (**b**)

Figure 11.4 shows an example of the granular features used in [14]. The masks corresponding to granular features have this special structure: most of the mask values are 0, while a few nonoverlapping square subregions in the mask have values of all $+1$ (or all -1). The square subregions are confined to be of size 1×1, 2×2, 4×4, or 8×8, so that the integral image trick can be used to quickly compute granular features.

The subwindow in Fig. 11.4 is of size 20×20, thus a feature is equivalent to a 20×20 mask, in which the gray pixels correspond to the value 0, the 8×8 subregion correspond to the value $+1$ in the mask, and the two 4×4 subregions correspond to the value -1. By precomputing three integral images for size 2, 4, and 8 subregions correspondingly, the granular feature values can be quickly computed.

Recently other local features are also proposed for usage in face detection, among which the local binary pattern feature (LBP) has exhibited promising results.[3] As illustrated in Fig. 11.5(a), the original LBP operator compares a pixel with its 8 neighbors, generating a single bit '1' if the neighboring pixel has higher intensity values (and a bit '0' if otherwise). The LBP value is then a combination of these 8 bits. The multi-block LBP (MB-LBP) generalize LBP by comparing the average

[3]The modified Census Transform feature [15] is also used for face detection, and is very similar to LBP.

intensity of pixels within a region instead of comparing single pixel intensities (illustrated in Fig. 11.5(b)).

MB-LBP is used in [20] and [52] for face detection and recognition, which shows that MB-LBP can produced improved performance than Haar-like features in face detection. The MB-LBP features can also be computed efficiently using integral images [20].

11.3.2 Learning Weak Classifiers

As mentioned earlier, the AdaBoost learning procedure is aimed at learning a sequence of weak classifiers $h_m(x)$ and the combining weights α_m in (11.1). It solves the following three fundamental problems: (1) learning effective 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 to construct a strong classifier.

AdaBoost assumes that a "weak learner' procedure is available. The task of the procedure is to select the most significant feature from a set of candidate features, given the current strong classifier learned thus far, and then construct the best weak classifier and combine it into the existing strong classifier. Here, the "significance" is with respect to some given criterion (see below).

In the case of discrete AdaBoost, the simplest type of weak classifiers is a "stump." A stump is a single-node decision tree. When the feature is real-valued, a stump may be constructed by thresholding the value of the selected feature at a certain threshold value; when the feature is discrete-valued, it may be obtained according to the discrete label of the feature. A more general decision tree (with more than one node) composed of several stumps leads to a more sophisticated weak classifier.

For discrete AdaBoost, a stump may be constructed in the following way. Assume that we have constructed $M-1$ weak classifiers $\{h_m(x)|m=1,\ldots,M-1\}$ and we want to construct $h_M(x)$. The stump $h_M(x) \in \{-1,+1\}$ is determined by comparing the selected feature $z_{k^*}(x)$ with a threshold τ_{k^*} as follows

$$h_M(x) = \begin{cases} +1 & \text{if } z_{k^*} > \tau_{k^*}, \\ -1 & \text{otherwise.} \end{cases} \tag{11.5}$$

In this form, $h_M(x)$ is determined by two parameters: the type of the scalar feature z_{k^*} and the threshold τ_{k^*}. The two may be determined by some criterion, for example, (1) the minimum weighted classification error, or (2) the lowest false alarm rate given a certain detection rate.

Supposing we want to minimize the weighted classification error with real-valued features, then we can choose a threshold $\tau_k \in \mathbb{R}$ for each feature z_k to minimize the corresponding weighted error made by the stump with this feature; we then choose the best feature z_{k^*} among all k that achieves the lowest weighted error.

0. (Input)
   (1) Training examples $\mathscr{X} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, and example weights $(w_1, \ldots, w_N)$, where $N = a + b$; of which a examples have $y_i = +1$ and b examples have $y_i = -1$.
   (2) The mask m corresponds to a feature.
1. (Initialization)
   Compute the feature values, $v_1, \ldots, v_N$, where $v_i = m^{\mathrm{T}} x_i$.
   Sort the feature values as $v_{i_1} \leq \cdots \leq v_{i_N}$, such that $(i_1, \ldots, i_N)$ is a permutation of $(1, \ldots, N)$.
   $\varepsilon \Leftarrow \sum_{y_i=-1} w_i$
2. (Updates)
   For $k = 1, \ldots, N$:
      if $y_{i_k} = -1$ then
         $\varepsilon \Leftarrow \varepsilon - w_{i_k}, \varepsilon_i \Leftarrow \varepsilon$
      else
         $\varepsilon \Leftarrow \varepsilon + w_{i_k}, \varepsilon_i \Leftarrow \varepsilon$
3. (Output)
   $k = \arg\min_{1 \leq i \leq N} \varepsilon_i$.
   Optimal threshold τ^*: $\tau^* = m^{\mathrm{T}} \tau_{i_k}$.

Fig. 11.6 Finding the optimal threshold of a weak classifier

Wu et al. show in [47] that only $N+1$ possible τ_k values need to be evaluated, and evaluating each τ_k is only $O(1)$ if we sort the feature values for z_k beforehand. This method to find the optimal threshold of a weak classifier is illustrated in Fig. 11.6.

Suppose the features are sorted in the order $v_{i_1} \leq \cdots \leq v_{i_N}$, and τ_{k_1} and τ_{k_2} satisfy that $v_{i_j} < \tau_{k_1}, \tau_{k_2} < v_{i_{j+1}}$, then setting the threshold to either τ_{k_1} or τ_{k_2} will result in the same weighted error. Thus, in Fig. 11.6 only the feature values (plus $-\infty$) are considered as possible thresholds. Given the weighted error at $\tau_k = v_{i_j}$, a simple update is sufficient to compute the error when $\tau_k = v_{i_{j+1}}$, because at most one example has changed its classification result.

Most of the computations in Fig. 11.6 is spent in the initialization part. One important observation in [47] is that the feature values need to be computed and sorted only once, because they do not change during the AdaBoost process even though the weights w change at each iteration. By storing the sorted feature values for all features in a table, the AdaBoost training time is reduced from weeks ([45]) to hours ([47]).

More features usually lead to higher detection accuracy [27]. However, it also means that the table of sorted feature values may be too large to be stored in the main memory. Pham and Cham construct weak classifiers using the mean and standard deviation of feature values. Given a feature, its associated mask m, and the AdaBoost weights $w^{(M-1)}$, the average feature value is $\sum_i w_i^{(M-1)} m^{\mathrm{T}} x_i$, where x_i is a set of training examples.

The integral image trick can be used to accelerate the computation of the mean feature value and standard deviation, that is, providing a way to utilize the structures in the mask m and computes $m^{\mathrm{T}} x$ quickly. Let x be an image subwindow in the stacked vector form and y be the corresponding integral image, it is clear from (11.2) that the transformation that generates y from x is linear, that is, there exists a square

matrix B such that $y = Bx$, and $m^{\mathrm{T}}x = m^{\mathrm{T}}B^{-1}y$. The average feature value is then [27]

$$\sum_i w_i^{(M-1)} m^{\mathrm{T}} x_i = m^{\mathrm{T}} B^{-1} \left(\sum_i w_i^{(M-1)} y_i \right). \tag{11.6}$$

In (11.6), the weighted average integral image $\sum_i w_i^{(M-1)} y_i$ can be computed in linear time, and the transformed mask $m^{\mathrm{T}}B^{-1}$ is sparse because of the structure in the mask m. Thus, the average feature value can be computed very quickly. Similarly, the (weighted) standard deviation can be quickly computed, too.

The faces are then modeled as a one-dimensional Gaussian distribution $N(\mu_+, \sigma_+^2)$, where μ_+ and σ_+ are computed from face examples. Nonfaces are modeled by $N(\mu_-, \sigma_-^2)$ similarly. The best threshold that separate two 1-d Gaussians can then be solved in a closed form.

This method is faster than examining all possible τ_k values, and has a much smaller storage requirement [27]. It is reported in [27] that the training speed is about two times faster than the algorithm presented in Fig. 11.6. This method has much less storage requirements and thus can train a strong classifier with more local features.

A decision stump is simple but may not fully utilize the information contained in a feature. A more complex weak classifier can be constructed by using piecewise decision functions [14, 22]: dividing the range of feature values into k nonoverlapping cells, and learn a decision function (for example, a decision stump) for every cell. Piece-wise decision functions take longer training time but usually have higher discrimination power than simple decision stumps.

Supposing that we want to achieve the lowest false alarm rate given a certain detection rate, we can set a threshold τ_k for each z_k so a specified detection rate (with respect to $w^{(M-1)}$) is achieved by $h_M(x)$ corresponding to a pair (z_k, τ_k). Given this, the false alarm rate (also with respect to $w^{(M-1)}$) due to this new $h_M(x)$ can be calculated. The best pair (z_{k^*}, τ_{k^*}) and hence $h_M(x)$ is the one that minimizes the false alarm rate.

There is still another parameter that can be tuned to balance between the detection rate and the false alarm rate: The class label prediction $\hat{y}(x) = \mathrm{sign}[H_M(x)]$ is obtained by thresholding the strong classifier $H_M(x)$ at the default threshold value 0. However, it can be done as $\hat{y}(x) = \mathrm{sign}[H_M(x) - T_M]$ with another value T_M, which can be tuned for the balance.

The form of (11.5) is for Discrete AdaBoost. In the case of real-valued versions of AdaBoost, such as RealBoost and LogitBoost, a weak classifier should be real-valued or output the class label with a probability value. For the real-value type, a weak classifier may be constructed as the log-likelihood ratio computed from the histograms of the feature value for the two classes. (See the literature for more details [17–19].) For the latter, it may be a decision stump or tree with probability values attached to the leaves [21].

11.3.3 Learning Strong Classifiers Using AdaBoost

AdaBoost learns a sequence of weak classifiers h_m and boosts them into a strong one H_M effectively by minimizing the upper bound on classification error achieved by H_M. The bound can be derived as the following exponential loss function [32]

$$J(H_M) = \sum_i e^{-y_i H_M(x_i)} = \sum_i e^{-y_i \sum_{m=1}^{M} \alpha_m h_m(x)} \tag{11.7}$$

where i is the index for training examples. AdaBoost constructs $h_m(x)$ ($m = 1, \ldots, M$) by stagewise minimization of (11.7). Given the current $H_{M-1}(x) = \sum_{m=1}^{M-1} \alpha_m h_m(x)$, and the newly learned weak classifier h_M, the best combining coefficient α_M for the new strong classifier $H_M(x) = H_{M-1}(x) + \alpha_M h_M(x)$ minimizes the cost

$$\alpha_M = \arg\min_{\alpha} J\big(H_{M-1}(x) + \alpha_m h_M(x)\big). \tag{11.8}$$

The minimizer is

$$\alpha_M = \log \frac{1 - \varepsilon_M}{\varepsilon_M} \tag{11.9}$$

where ε_M is the weighted error rate

$$\varepsilon_M = \sum_i w_i^{(M-1)} 1\big[\text{sign}\big(H_M(x_i)\big) \neq y_i\big] \tag{11.10}$$

where $1[C]$ is 1 if C is true but 0 otherwise.

Each example is reweighted after an iteration that is, $w_i^{(M-1)}$ is updated according to the classification performance of H_M:

$$\begin{aligned} w_i^{(M)} &= w_i^{(M-1)} \exp\big(-y_i \alpha_M h_M(x_i)\big) \\ &= \exp\big(-y_i H_M(x_i)\big) \end{aligned} \tag{11.11}$$

which is used for calculating the weighted error or another cost for training the weak classifier in the next round. This way, a more difficult example is associated with a larger weight so it is emphasized more in the next round of learning. The algorithm is summarized in Fig. 11.7.

11.3.4 Alternative Feature Selection Methods

In boosting based methods (cf. Fig. 11.7), the weak classifiers h_M and their related weights α_M are determined simultaneously: h_M is chosen to minimize certain objective value (for example, weighted error rate of the feature) and α_M is a function

0. (Input)
   (1) Training examples $\mathscr{Z} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$,
   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 optimal h_m to minimize the weighted error.
   (2) Choose α_m according to (11.9).
   (3) Update $w_i^{(m)} \leftarrow w_i^{(m)} \exp[-y_i \alpha_m h_m(x_i)]$ and normalize to $\sum_i w_i^{(m)} = 1$.
3. (Output)
   Classification function: $H_M(x)$ as in (11.1).
   Class label prediction: $\hat{y}(x) = \text{sign}[H_M(x)]$.

Fig. 11.7 AdaBoost learning algorithm

of the objective value. Wu et al. [47] show that if these tasks (learning h_M and setting α_M) are decoupled into two sequential steps, a more accurate strong classifier H_M can be obtained.

Different methods can be used to select features and train weak classifiers. Besides AdaBoost and other boosting variants, [47] showed that a greedy Forward Feature Selection (FFS) method can successfully select a subset of features from a large feature pool and learn corresponding weak classifiers. In boosting methods, a feature is selected if its corresponding weak classifier has minimum weighted error rate. FFS uses a different selection criterion that is directly related to the strong classifier's performance. In FFS, if a partial strong classifier H_{M-1} is already constructed, a feature h_{M^*} is selected in iteration M only if it leads to highest strong classifier accuracy, that is, $H_{M-1} \cup h_{M^*}$ has the highest accuracy among all possible h_M. FFS uses majority vote (that is, $\alpha_i = 1$ for all i). A table of feature values are stored to ensure fast weak classifier training. FFS trains faster than the AdaBoost method, and achieves comparable but slightly lower detection accuracy than AdaBoost.

In fact, 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 [32]. Conceptually, FFS and AdaBoost shares the greedy feature selection idea, although different objective functions are used to guide the greedy search procedures.

A crucial heuristic assumption used in such a sequential forward search procedure is the monotonicity (that is, that addition of a new weak classifier to the current set does not decrease the value of the performance criterion). The premise offered by the sequential procedure in AdaBoost or FFS breaks down when this assumption is violated. Floating Search [29] is a sequential feature selection procedure with backtracking, aimed to deal with nonmonotonic criterion functions for feature selection. The sequential forward floating search (SFFS) methods [29] adds or deletes a single ($\ell = 1$) feature and then backtracks r steps, where r depends on the current situation.

0. (Input)
   (1) Training examples $\mathscr{Z} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
   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 $J(H_M)$, 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 $m = 1, \ldots, M_{\max}$), $M = 0$, $\mathscr{H}_0 = \{\}$.
2. (Forward inclusion)
   (1) $M \leftarrow M + 1$.
   (2) Learn h_M and α_M.
   (3) Update $w_i^{(M)} \leftarrow w_i^{(M-1)} \exp[-y_i \alpha_M h_M(x_i)]$, normalize to $\sum_i w_i^{(M)} = 1$.
   (4) $\mathscr{H}_M = \mathscr{H}_{M-1} \cup \{h_M\}$;
   If $J_M^{\min} > J(H_M)$, then $J_M^{\min} = J(H_M)$.
3. (Conditional exclusion)
   (1) $h' = \arg\min_{h \in \mathscr{H}_M} J(H_M - h)$.
   (2) If $J(H_M - h') < J_{M-1}^{\min}$, then
      (a) $\mathscr{H}_{M-1} = \mathscr{H}_M - h'$.
      $J_{M-1}^{\min} = J(H_M - h')$; $M = M - 1$.
      (b) If $h' = h_{m'}$, then
      recalculate $w_i^{(j)}$ and h_j for $j = m', \ldots, M$.
      (c) Go to 3.(1).
   (3) Else
      (a) If $M = M_{\max}$ or $J(\mathscr{H}_M) < J^*$, then go to 4.
      (b) Go to 2.(1).
4. (Output)
   Classification function: $H_M(x)$ as in (11.1).
   Class label prediction: $\hat{y}(x) = \text{sign}[H_M(x)]$.

Fig. 11.8 FloatBoost algorithm

The FloatBoost Learning procedure is shown in Fig. 11.8. It is composed of several parts: the training input, initialization, forward inclusion, conditional exclusion, and output. In step 2 (forward inclusion), the currently most significant weak classifiers are 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 the set $\mathscr{H}_M$ of current weak classifiers, subject to the condition that the removal leads to a lower cost than $J_{M-1}^{\min}$. Supposing that the weak classifier removed was the m'th in $\mathscr{H}_M$, then $h_{m'}, \ldots, h_{M-1}$ and the α_m's must be relearned. These steps are repeated until no more removals can be done.

11.3.5 Asymmetric Learning Methods

The face detection (and other object detection) problem is a rare-event detection problem [47], in the sense that the face (or target object) only occupies a small

number of subwindows while the nonface (or nonobject) subwindows are on the order of millions even in a small-sized image. This asymmetric nature of the classifier learning problem is long recognized and methods have been proposed to build a strong classifier that takes into account the asymmetry property.

Viola and Jones proposed the AsymBoost method [44], which is a modification of the AdaBoost algorithm. The essence of AsymBoost is to focus more on positive examples by changing the weight update rule (11.11) to

$$w_i^{(M)} = C \exp\bigl(-y_i H_M(x_i)\bigr), \tag{11.12}$$

where $C = (\sqrt{K})^{(1/T)}$ if $y_i > 0$ and $C = (\sqrt{K})^{(-1/T)}$ if $y_i < 0$, and $K > 1$ is a parameter that measures that level of asymmetry. We assume that the boosting learning procedure will repeat T rounds. In the T rounds of AdaBoost algorithm, positive examples are continuously assigned higher weights than negative examples.

Wu et al. propose another asymmetric learning method. Wu et al. [47] shows that it is advantageous to adjust the values of α_i after the features are selected, according to the cascade detection framework. Assuming the false alarm rate of all strong classifiers in a cascade is 0.5, a 20-node cascade will have a 10^{-6} false alarm rate if we assume the strong classifiers reject nonfaces independent to each other. Thus, Wu et al. propose the following learning goal for the strong classifiers in a cascade: "*for every node, design a classifier with very high (e.g. 99.9%) detection rate and only moderate (e.g., 50%) false positive rate*." Linear Asymmetric Classifier (LAC) is designed to find the α that achieve this goal.

Given weak classifiers $h_1, h_2, \ldots, h_M$, an example x is mapped to a vector of responses $\mathbf{h}(x) = (h_1(x), h_2(x), \ldots, h_M(x))$. LAC computes the distributions of the vector $\mathbf{h}(x)$: μ_+ and Σ_+ are mean and covariance matrix of $\mathbf{h}(x)$ when x is the set of faces. Similarly, μ_- and Σ_- are the mean and covariance matrix computed using nonfaces. It is showed in [47] that the following LAC solution vector $\alpha^* \in \mathbb{R}^M$ is globally optimal for the cascade learning goal under certain reasonable assumptions:

$$\alpha^* = \Sigma_+^{-1}(\mu_+ - \mu_-). \tag{11.13}$$

Another way to set the α vector is to use the Fisher's Discriminant Analysis (FDA). Experiments in [47] show that using LAC or FDA to set the α vector consistently improve cascade detection accuracy, no matter the weak classifiers are selected and trained using AdaBoost or FFS.

11.3.6 Cascade of Strong Classifiers

A boosted strong classifier effectively eliminates a large portion of nonface subwindows while maintaining a high detection rate. Nonetheless, a single strong classifier may not meet the requirement of an extremely low false alarm rate (for example, 10^{-6} or even lower). A solution is to arbitrate between several detectors (strong classifier) [31], for example, using the "AND" operation.

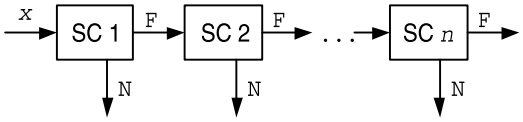

Fig. 11.9 A cascade of n strong classifiers (SC). The input is a subwindow x. It is sent to the next SC for further classification only if it has passed all the previous SCs as the face (F) pattern; otherwise it exits as nonface (N). x is finally considered to be a face when it passes all the n SCs

Viola and Jones [43, 45] further extend this idea by training a cascade consisting of a cascade of strong classifiers, as illustrated in Fig. 11.9. A strong classifier is trained using bootstrapped nonface examples that pass through the previously trained cascade. Usually, 10 to 20 strong classifiers are cascaded. For face detection, subwindows that fail to pass a strong classifier are not further processed by the subsequent strong classifiers. This strategy can significantly speed up the detection and reduce false alarms, with a little sacrifice of the detection rate.

Various improvements have also been proposed to the cascade structure. Xiao et al. argue that historical information is useful during the cascade training [48], that is, we should not ignore the information contained in the strong classifiers $\mathrm{SC}_1, \ldots, \mathrm{SC}_{M-1}$ when we train the M-th strong classifier SC_M (which is the practice in Fig. 11.7). The Boosting Chain framework is proposed in [48] to incorporate such historical information: the strong classifier SC_{M-1} is treated as the first "weak classifier" in SC_M. This modification to the cascade framework reduces the number of required weak classifiers and increases detection accuracy [6, 48].

Another attempt to modify the cascade framework is the soft cascade method [5] or similar ideas [28, 49]. Soft cascade is an extreme cascade structure: a "monolithic" strong classifier composed of multiple weak classifiers, much similar to the strong classifier in the cascade framework. Let $c_1(x), \ldots, c_T(x)$ be the weak classifiers that form a soft cascade:

$$H_T(x) = \sum_{i=1}^{\mathrm{T}} c_i(x). \tag{11.14}$$

A soft cascade associates a rejection threshold r_t for every partial strong classifier $H_t(x) = \sum_{i=1}^{t} c_i(x)$. If $H_t(x) < r_t$, the input subwindow x is rejected as nonface and the weak classifiers $c_{t+1}, \ldots, c_T$ are not evaluated. In other words, a soft cascade is similar to a cascade structure that requires only 1 weak classifier per node. However, since historical information is preserved in H_t, soft cascades achieves high detection performances.

After the weak classifiers $c_1, \ldots, c_T$ are trained, the soft cascade method rearranges the order of these weak classifiers. This step is carried out using a separate set of validation examples. An optimal ordering of weak classifiers and rejection thresholds r_t are chosen to minimize both the detection errors and testing time computational costs [5].

Similar ideas are proposed to improve the original cascade framework in [6]. Suppose that a cascade consists of strong classifiers $\mathrm{SC}_1, \ldots, \mathrm{SC}_M$, where SC_i is

trained using the AdaBoost method. By adjusting the threshold of the strong classifier, we can get different strong classifier performance in terms of false alarm rates and detection rates. This threshold is usually determined manually by setting a fixed goal for either detection or false alarm rate, which not necessarily leads to optimal cascade detection performance. Brubaker et al. proposed a "two-point" algorithm to automatically find optimal thresholds of strong classifiers. The two-point algorithm uses less weak classifiers than fixed goal AdaBoost (and thus faster detection speed), and achieves higher detection performances.

11.4 Dealing with Head Rotations

Multiview face detection should be able to detect nonfrontal faces. Face detection methods usually handle two major types of head rotation: (1) out-of-plane (left-right) rotation; (2) in-plane rotation.

Rowley et al. [30] propose to use two neural network classifiers for detection of frontal faces subject to in-plane rotation. The first is the router network, trained to estimate the orientation of an assumed face in the subwindow, though the window may contain a nonface pattern. The inputs to the network are the intensity values in a preprocessed 20×20 subwindow. The angle of rotation is represented by an array of 36 output units, in which each unit represents an angular range. With the orientation estimate, the subwindow is derotated to make the potential face upright. The second neural network is a normal frontal, upright face detector.

Within the cascade detector framework, detector-pyramids have been proposed to detect and merge faces in different poses and have achieved the state-of-the-art detection performance.

11.4.1 Hierarchical Organization of Multi-view Faces

The Width-First-Search structure (Fig. 11.10) by Huang et al. in [14] handles in-plane and out-of-plane rotations simultaneously. Huang et al. [14] manually divides the face range into 15 different poses, and arranges such poses in a four level tree structure. The top level tree node includes all face poses. The second level contains 3 nodes, which correspond to left profile, frontal, and right profile faces. The third level further refines to 5 nodes, where left and right profile faces are split into 2 different nodes based on the out-of-plane rotation angle. The first 3 levels handle out-of-plane rotations. Each node in the third level is split to 3 nodes in the final level, handling different in-plane rotation angles.

The tree structure in [14] handles out-of-plane rotation in $\Theta = [-90°, +90°]$ and in-plane rotation in $\Phi_2 = [-45°, +45°]$. The full in-plane rotation range $\Phi = [-180°, +180°]$ is covered by rotating the features 90°, 180°, and 270°. It is noticed that for these specific rotation angles, rotating the features is equivalent to rotate the

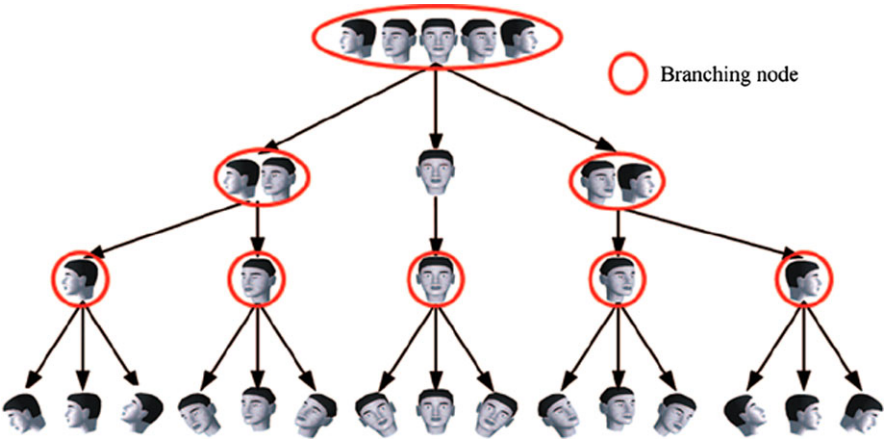

Fig. 11.10 Illustration of the structure for detecting multi-view faces. From Huang et al. [14], © 2007 IEEE, with permission

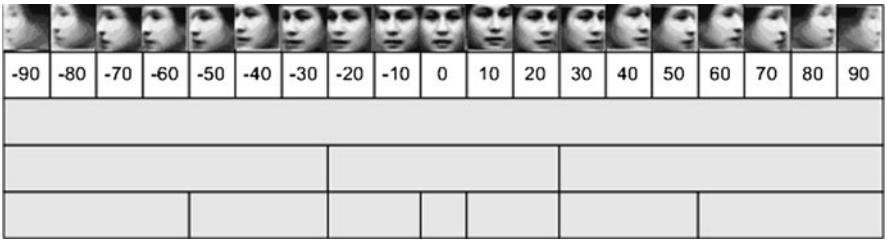

Fig. 11.11 Out-of-plane view partition. Out-of-plane head rotation (*row 1*), the facial view labels (*row 2*), and the coarse-to-fine view partitions at the three levels of the detector-pyramid (*rows 3* to *5*)

mask m associated with a feature by a corresponding angle. Rotating features is more efficient than rotating images.

A similar hierarchy is used by Li et al. [17, 19] to handle out-of-plane rotation in $[-90°, +90°]$, shown in Fig. 11.11. It is worth noting that the face-pose hierarchy in Fig. 11.11 does not handle in-plane rotations. The leaf detectors are designed to handle in-plane rotations in the range $[-15°, +15°]$. The full in-plane rotation in $\Phi = [-45°, +45°]$ is dealt with by also applying the detector-pyramid on the rotated test images ($\pm 30°$).

11.4.2 From Face-Pose Hierarchy to Detector-Pyramid

A detector-pyramid that detects multi-view faces can be derived directly from the face-pose hierarchy. Every node in Fig. 11.10 corresponds to a strong classifier. For example, the root node determines whether a subwindows contains a left profile,

Table 11.1 Desired vectors for different face poses and non-faces in Vector Boosting training

Cases	Desired output vector
Left profile face	(1, 0, 0, 0)
Frontal face	(0, 1, 0, 0)
Right profile face	(0, 0, 1, 0)
Nonfaces	(−1, 0, 0, 0), (0, −1, 0, 0), (0, 0, −1, 0)

Fig. 11.12 Merging from different channels. From left to right: Outputs of frontal, left, and right view channels and the final result after the merge

frontal, or right profile face. It is important to note that multiple children nodes of the same parent node can be activated simultaneously (that is, a width-first-search of a tree structure). Huang et al. made this choice based on the following argument: different face poses are jointly competing with nonfaces, and the discrimination among them is less important (except in the final level). This hypothesis requires nodes in the top 3 levels to be multi-class classifiers, and a Vector Boosting algorithm is proposed in [14] to satisfy this special requirement.

For example, a partial profile face with a 45° out-of-plane rotation angle can be detected by both the right profile face node and the frontal face node in the second layer. In order to achieve higher detection accuracy, it is reasonable to further examine the two sub-trees rooted at both nodes. In the Vector Boosting classifier for the root node, the desired output is a vector and the desired vectors for different cases are summarized in Table 11.1.

At the final level, the single node with the highest confidence is chosen as the detected face pose. In order to get a classifier with very low false alarm rate, the classifier in the leaf node of the width-first-search tree in [14] is in fact a cascade detector.

Figure 11.11 leads to another detector pyramid. Instead of using a multi-class boosting algorithm that generates a vector output, [17] uses k binary RealBoost strong classifiers if a node has k children nodes. Multiple children of a node can be activated (that is, further examining the subtree rooted at a child node) if more than one binary RealBoost classifiers output positively. At the final level, multiple leaf nodes (corresponding to different face poses) can be active for one subwindow. Different from [14], faces detected by the seven channels at the final level of Fig. 11.11 are merged to obtain the final result. This is illustrated in Fig. 11.12.

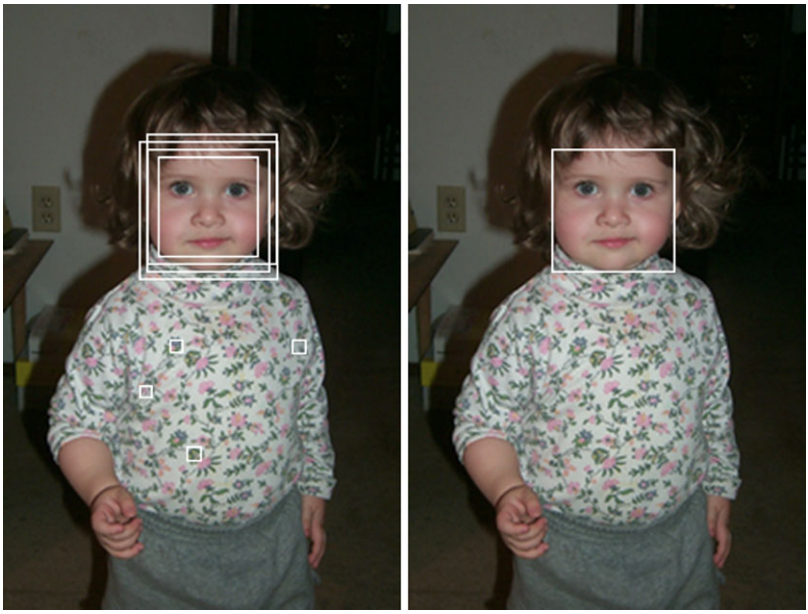

Fig. 11.13 Merging multiple detections

11.5 Postprocessing

A single face in an image may be detected several times at close locations or on multiple scales. False alarms may also occur but usually with less consistency than multiple face detections. The number of multiple detections in a neighborhood of a location can be used as an effective indication for the existence of a face at that location. This assumption leads to a heuristic for resolving the ambiguity caused by multiple detections and eliminating many false detections. A detection is confirmed if the number of multiple detections is greater than a given value; and given the confirmation, multiple detections are merged into a consistent one. This is practiced in most face detection systems [31, 38]. Figure 11.13 gives an illustration. The image on the left shows a typical output of initial detection, where the face is detected four times with four false alarms on the cloth. On the right is the final result after merging. After the postprocessing, multiple detections are merged into a single face and the false alarms are eliminated. Figures 11.14 and 11.15 show some typical frontal and multiview face detection examples; the multiview face images are from the Carnegie Mellon University (CMU) face database [42].

11.6 Performance Evaluation

The result of face detection from an image is affected by the two basic components: the face/nonface classifier and the postprocessing (merger). To understand

Fig. 11.14 Results of frontal face detection

how the system works, it is recommended that the two components be evaluated separately [1], with two types of test data. The first consists of face icons of a fixed size (as are used for training). This process aims to evaluate the performance of the face/nonface classifier (preprocessing included), without being affected by merging. The second type of test data consists of normal images. In this case, the face detection results are affected by both trained classifier and merging; the overall system performance is evaluated.

11.6.1 Performance Measures

The face detection performance is primarily measured by two rates: the correct detection rate (which is 1 minus the miss rate) and the false alarm rate. The performance can be observed by plotting on the receiver operating characteristic (ROC) curves.

The false alarm rate is computed as the percentage of the subwindows that are nonfaces but wrongly classified as faces. However, the number of false detections (remaining after merging multiple detections) is a better suited metric because it

Fig. 11.15 Results of multiview face detection

reflects the effect of postprocessing and directly links to the final output of a face detection system. Although the false alarm rate is usually positively correlated with the number of false detections, recently more authors are reporting number of false detections (after postprocessing) in the X-axis of the ROC curves. Figure 11.16 shows several examples of the ROC curves, including four recent methods evaluated on the benchmark MIT-CMU frontal face dataset [42].

An ideal face detection system should have a detection rate of 100%, with a false alarm rate of 0, though none of the current systems can achieve this generally. In practical systems, increasing the detection rate is usually accompanied by an increase in the false alarm rate. In the case where a confidence function is used to

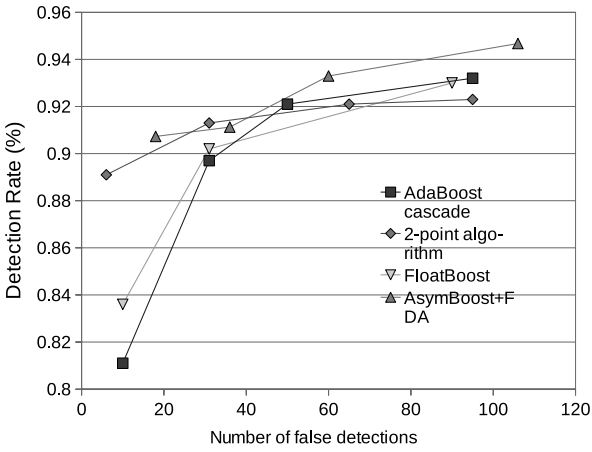

Fig. 11.16 Typical ROC curve for face detection on the MIT-CMU frontal face dataset. The algorithms include the AdaBoost implementation in [6], the 2-point algorithm [6], FloatBoost [17], and AsymBoost + FDA [47]

distinguish between the face and nonface subwindows, with the high output value indicating the detection of face and low value nonface, a trade-off between the two rates can be made by adjusting the decisional threshold. In the case of the AdaBoost learning method, the threshold for (11.1) is learned from the training face icons and bootstrapped nonface icons, so a specified rate (usually the false alarm rate) is under control for the training set. Remember that performance numbers of a system are always with respect to the data sets used; two algorithms or systems cannot be compared directly unless the same data sets are used.

11.6.2 Comparison of Cascade-Based Detectors

As the cascade-based methods (with local features) have so far provided the best face detection solutions in terms of the statistical rates and the speed, the following provides a comparative evaluation on different boosting algorithms (DAB: discrete AdaBoost; RAB: real AdaBoost; and GAB: gentle AdaBoost), different training sets preparations, and different weak classifiers. The results provide empirical references for face detection engineers.

- **Boosting Algorithms**. Three 20-stage cascade classifiers were trained with DAB, RAB, and GAB using the Haar-like feature set of Viola and Jones [43, 45] and stumps as the weak classifiers. It is reported that GAB outperformed the other two boosting algorithms [21]. Also, a smaller rescaling factor for scanning images was beneficial for a high detection rate.
- **Weak Classifiers**. Stumps are the simplest tree type of weak classifiers (WCs) that can be used in discrete AdaBoost. A stump is a single-node tree that does not allow learning dependence between features. In general, n split nodes are needed to model dependence between $n - 1$ variables. It is reported in [6] that using a decision tree as weak classifier, the cascade achieves up to 15% higher detection

Table 11.2 Average number of features evaluated per nonface subwindow of size 20×20 (reproduced from Lienhart et al. [21])

	Number of splits			
AdaBoost type	1	2	3	4
DAB	45.09	44.43	31.86	44.86
GAB	30.99	36.03	28.58	35.40
RAB	26.28	33.16	26.73	35.71

rate with the same number of false detections. Other complex weak classifier (for example, piece-wise decision functions) can also increase detection performance.

- **Detection Speed**. Table 11.2 compares the CART tree weak classifiers of varying number of nodes in terms of the effectiveness of rejecting nonface subwindows. RAB is the most effective. It is also reported in [6] that RAB has the fastest detection speed. Reusing historical information [48] is confirmed to be effective in reducing testing time by [6].
- **Haar-like and Other Local Features**. The experiments in [21] and other works (for example, [14, 27]) suggest that whereas the larger Haar-like feature set makes it more complex in both time and memory in the boosting learning phase, gain is obtained in the detection phase. Using approximately the same training time, [27] (with 295 920 local features) reported about 5% lower false alarms than the method in Fig. 11.6 (with 40 000 local features). Other local features such as MB-LBP also exhibits excellent detection results [52].
- **Subwindow Size**. Different subwindow sizes, ranging from 16×16 up to 32×32, have been used on face detection. The experiments [21] show that a subwindow size of 20×20 achieves the highest detection rate at an absolute number of false alarms between 5 and 100 on the CMU test set of frontal faces. A subwindow size of 24×24 worked better for false alarms fewer than five.

11.7 Conclusions

Face detection is the first step in automated face recognition and has applications in biometrics and multimedia management. Owing to the complexity of the face and nonface manifolds, highly accurate face detection with a high detection rate and low false alarm rate has been challenging. Now this difficult problem has almost been solved to meet the minimum requirements of most practical applications, because of the advances in face recognition research and machine learning.

Boosting-based face detection methods [14, 17, 19, 21, 43, 45, 47] have been the most effective of all those developed so far. In terms of detection and false alarm rates, they are comparable to the neural network method of Rowley et al. [31], but are several times faster.

Regarding the boosting based approach, the following conclusions can be drawn in terms of feature sets, boosting algorithms, weak classifiers, subwindow sizes, and training set sizes according to reported studies [14, 17, 19, 21, 43, 45, 47]:

- An over-complete set of Haar-like features are effective for face detection. The use of the integral image method makes computation of these features efficient and achieves scale invariance. Extended Haar-like features help detect nonfrontal faces.
- AdaBoost learning can select best subset from a large feature set and construct a powerful nonlinear classifier.
- The cascade structure significantly improves the detection speed and effectively reduces false alarms, with a little sacrifice of the detection rate.
- Selecting the weak classifiers and learning the weights that combine those weak classifiers can be decoupled.
- Alternative feature selection methods can be used to reduce training time (for example, FFS), or to achieve lower error rate or detection time (for example, FloatBoost).
- Asymmetry needs to be taken care of in learning classifiers for face detection. Weights that are specifically learned to satisfy learning goals in the cascade framework (for example, using LAC) improve detection accuracy.
- Less aggressive versions of AdaBoost, such as GentleBoost and LogitBoost, may be preferable to discrete and real AdaBoost in dealing with training data containing outliers [12].
- Representationally, more complex weak classifiers such as small CART trees can model second-order and/or third-order dependencies, and may be beneficial for the nonlinear task of face detection.

Although face detection technology is now sufficiently mature to meet the minimum requirements of many practical applications, much work is still needed before automatic face detection can achieve performance comparable to the human performance. The Haar + AdaBoost approach is effective and efficient. However, the current approach has almost reached its power limit. Within such a framework, improvements may be possible by designing additional sets of features that are complementary to the existing ones and adopting more advanced learning techniques, which could lead to more complex classifiers while avoiding the overfitting problem.

Acknowledgements This work was partially supported by the Chinese National Natural Science Foundation Project #61070146, the National Science and Technology Support Program Project #2009BAK43B26, and the AuthenMetric R&D Funds (2004–2011). The work was also partially supported by the TABULA RASA project (http://www.tabularasa-euproject.org) under the Seventh Framework Programme for research and technological development (FP7) of the European Union (EU), grant agreement #257289.

References

1. Alvira, M., Rifkin, R.: An empirical comparison of SNoW and svms for face detection. Technical Report AI Memo 2001-004 & CBCL Memo 193, MIT (2001)
2. Amit, Y., Geman, D., Wilder, K.: Joint induction of shape features and tree classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **19**, 1300–1305 (1997)

3. Baker, S., Nayar, S.: Pattern rejection. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 544–549 (1996)
4. Bichsel, M., Pentland, A.P.: Human face recognition and the face image set's topology. CVGIP, Image Underst. **59**, 254–261 (1994)
5. Bourdev, L.D., Brandt, J.: Robust object detection via soft cascade. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. II, pp. 236–243 (2005)
6. Brubaker, S.C., Wu, J., Sun, J., Mullin, M.D., Rehg, J.M.: On the design of cascades of boosted ensembles for face detection. Int. J. Comput. Vis. **77**(1–3), 65–86 (2008)
7. Crow, F.: Summed-area tables for texture mapping. In: SIGGRAPH, vol. 18(3), pp. 207–212 (1984)
8. Elad, M., Hel-Or, Y., Keshet, R.: Pattern detection using a maximal rejection classifier. Pattern Recognit. Lett. **23**, 1459–1471 (2002)
9. 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)
10. Fleuret, F., Geman, D.: Coarse-to-fine face detection. Int. J. Comput. Vis. **20**, 1157–1163 (2001)
11. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
12. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequoia Hall, Stanford University, July 1998
13. 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)
14. Huang, C., Ai, H., Li, Y., Lao, S.: High-performance rotation invariant multiview face detection. IEEE Trans. Pattern Anal. Mach. Intell. **29**(4), 671–686 (2007)
15. Küblbeck, C., Ernst, A.: Face detection and tracking in video sequences using the modified census transformation. Image Vis. Comput. **24**(6), 564–572 (2006)
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, May 1999
17. Li, S.Z., Zhang, Z.: FloatBoost learning and statistical face detection. IEEE Trans. Pattern Anal. Mach. Intell. **26**(9), 1112–1123 (2004)
18. Li, S.Z., Zhang, Z.Q., Shum, H.-Y., Zhang, H.: FloatBoost learning for classification. In: Proceedings of Neural Information Processing Systems, Vancouver (2002)
19. Li, S.Z., Zhu, L., Zhang, Z.Q., Blake, A., Zhang, H., Shum, H.: Statistical learning of multiview face detection. In: Proceedings of the European Conference on Computer Vision, vol. 4, pp. 67–81, Copenhagen, Denmark, 28 May–2 June 2002
20. Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: International Conference on Biometrics, pp. 828–837 (2007)
21. Lienhart, R., Kuranov, A., Pisarevsky, V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. MRL Technical Report, Intel Labs, December 2002
22. Liu, C., Shum, H.-Y.: Kullback–Leibler boosting. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. I, pp. 587–594 (2003)
23. Moghaddam, B., Pentland, A.: Probabilistic visual learning for object representation. IEEE Trans. Pattern Anal. Mach. Intell. **7**, 696–710 (1997)
24. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: An application to face detection. In: CVPR, pp. 130–136 (1997)
25. Papageorgiou, C.P., Oren, M., Poggio, T.: A general framework for object detection. In: Proceedings of IEEE International Conference on Computer Vision, pp. 555–562, Bombay (1998)
26. 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, pp. 84–91 (1994)

27. Pham, M.-T., Cham, T.-J.: Fast training and selection of Haar features using statistics in boosting-based face detection. In: Proceedings of IEEE International Conference on Computer Vision (2007)
28. Pham, M.-T., Hoang, V.-D.D., Cham, T.-J.: Detection with multi-exit asymmetric boosting. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2008)
29. Pudil, P., Novovicova, J., Kittler, J.: Floating search methods in feature selection. Pattern Recognit. Lett. **15**(11), 1119–1125 (1994)
30. Rowley, H., Baluja, S., Kanade, T.: Rotation invariant neural network-based face detection. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1998)
31. Rowley, H.A., Baluja, S., Kanade, T.: Neural network-based face detection. IEEE Trans. Pattern Anal. Mach. Intell. **20**(1), 23–28 (1998)
32. Schapire, R., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. Ann. Stat. **26**(5), 1651–1686 (1998)
33. Schneiderman, H.: A statistical approach to 3D object detection applied to faces and cars (CMU-RI-TR-00-06). PhD thesis, RI (2000)
34. 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)
35. Schneiderman, H., Kanade, T.: Object detection using the statistics of parts. Int. J. Comput. Vis. **56**(3), 151–177 (2004)
36. 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, vol. 11, pp. 571–577. MIT Press, Cambridge (1998)
37. 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, New York (1998)
38. Sung, K.-K., Poggio, T.: Example-based learning for view-based human face detection. IEEE Trans. Pattern Anal. Mach. Intell. **20**(1), 39–51 (1998)
39. Tieu, K., Viola, P.: Boosting image retrieval. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 228–235 (2000)
40. Turk, M.: A random walk through eigenspace. IEICE Trans. Inf. Syst. **E84-D**(12), 1586–1695 (2001)
41. Turk, M.A., Pentland, A.P.: Eigenfaces for recognition. J. Cogn. Neurosci. **3**(1), 71–86 (1991)
42. Various Face Detection Databases. www.ri.cmu.edu/projects/project_419.html
43. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii (2001)
44. Viola, P.A., Jones, M.J.: Fast and robust classification using asymmetric AdaBoost and a detector cascade. In: Advances in Neural Information Processing Systems, vol. 14, pp. 1311–1318 (2001)
45. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
46. Wiskott, L., Fellous, J., Kruger, N., v. d. Malsburg, C.: Face recognition by elastic bunch graph matching. IEEE Trans. Pattern Anal. Mach. Intell. **19**(7), 775–779 (1997)
47. Wu, J., Brubaker, S.C., Mullin, M.D., Rehg, J.M.: Fast asymmetric learning for cascade face detection. IEEE Trans. Pattern Anal. Mach. Intell. **30**(3), 369–382 (2008)
48. Xiao, R., Zhu, L., Zhang, H.J.: Boosting chain learning for object detection. In: Proceedings of IEEE International Conference on Computer Vision, pp. 709–714 (2003)
49. Xiao, R., Zhu, H., Sun, H., Tang, X.: Dynamic cascades for face detection. In: Proceedings of IEEE International Conference on Computer Vision (2007)

50. Yang, M.-H., Kriegman, D., Ahuja, N.: Detecting faces in images: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **24**(1), 34–58 (2002)
51. Yang, M.-H., Roth, D., Ahuja, N.: A SNoW-based face detector. In: Proceedings of Neural Information Processing Systems, pp. 855–861 (2000)
52. Zhang, L., Chu, R., Xiang, S., Liao, S., Li, S.Z.: Face detection based on multi-block LBP representation. In: International Conference on Biometrics, pp. 11–18 (2007)