High-Performance Rotation Invariant Multiview Face Detection

Chang Huang, *Student Member*, *IEEE*, Haizhou Ai, *Member*, *IEEE*,
Yuan Li, and Shihong Lao, *Member*, *IEEE*

Abstract—Rotation invariant multiview face detection (MVFD) aims to detect faces with arbitrary *rotation-in-plane* (RIP) and *rotation-off-plane* (ROP) angles in still images or video sequences. MVFD is crucial as the first step in automatic face processing for general applications since face images are seldom upright and frontal unless they are taken cooperatively. In this paper, we propose a series of innovative methods to construct a high-performance rotation invariant multiview face detector, including the *Width-First-Search (WFS) tree* detector structure, the *Vector Boosting* algorithm for learning vector-output strong classifiers, the *domain-partition-based weak learning* method, the *sparse feature in granular space*, and the *heuristic search* for sparse feature selection. As a result of that, our multiview face detector achieves low computational complexity, broad detection scope, and high detection accuracy on both standard testing sets and real-life images.

Index Terms—Pattern classification, AdaBoost, vector boosting, granular feature, rotation invariant, face detection.

1 Introduction

In the past several decades, we have witnessed a burst of activities in applying robust computer vision systems for Human-Computer-Interaction (HCI). The face, which contains very important biological information of human being, is a very interesting object in images and videos. Naturally, face detection, which locates face regions at the very beginning, is considered as a fundamental part of any automatic face processing system. Also, it is a challenging work since the difficulties of developing a robust face detector arise from not only the diversities in the nature of human faces (e.g., the variability in size, location, pose, orientation, and expression) but also the changes of environment conditions (e.g., illumination, exposure, occlusion, etc.) [1].

Generally speaking, there are mainly two methodologies for face detection task: one is knowledge-based and the other is learning-based. The knowledge-based methodology attempts to depict our prior knowledge about the face pattern with some explicit rules, such as the intensity of faces, elliptic face contour, and equilateral triangle relation between eyes and mouth [2], [3]. Unfortunately, it is impossible to translate all human knowledge exactly into those required explicit rules that could be accurately comprehended by computers. As a result, methods of this type often perform poorly when the rules mismatch unusual faces or match too many background patches. On the other hand, the learning-based methodology, of which the representatives include Osuna et al.'s SVM method [4], Rowley et al.'s ANN method [5] and Schneiderman and Kanade's Bayesian-rule method [6], tries to model the face pattern with distribution functions or discriminant functions under the probabilistic framework. Methods of this kind are not limited by our describable knowledge on faces but determined by the capability of learning model and training samples, hence being able to deal with more complex cases compared with the knowledge-based approach. Specifically, the breakthrough of learning-based methodology happened in 2001 when Viola and Jones proposed a novel boosted cascade framework [7]. This work showed amazing real-time speed and high detection accuracy. People usually attribute the achievements of this work to the fast calculated Haar-like features via the integral image and the cascade structure of classifiers learned by AdaBoost. Here, for further analysis, we decompose their framework into four levels from top to bottom as shown in Table 1.

Based on the premise that a significant disparity of occurrence rate between faces and background region in common images exists, Viola and Jones adopted an asymmetric cascade model that connected a series of strong classifiers with AND logic operators and each classifier made unbalanced decisions for face and nonface categories. Consequently, most of the background region could be rejected rapidly by the first several classifiers with very little computation. To learn such strong classifiers, they employed the AdaBoost algorithm [8], which could efficiently combine many weak classifiers, acting as a feature selection mechanism, and guarantee a strong generalization bound for final classification. Finally, on the bottom level, they enumerated a large number of Haar-like features based on the integral image and associated them with corresponding stump functions to form a redundant weak classifier pool, which provided fundamental discriminability for the AdaBoost algorithm. All these active factors were organized effectively by Viola and Jones to yield their distinguished work on face detection [7].

Although the frontal face detection seems to be mature so far, it is often inadequate to meet the rigorous requirements of

- C. Huang, H. Ai, and Y. Li are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: {huangc99, yuan-li}@mails.tsinghua.edu.cn, ahz@mail.tsinghua.edu.cn.
- S. Lao is with Sensing and Control Technology Laboratory, OMRON Corporation, Kyoto 619-0283, Japan. E-mail: lao@ari.ncl.omron.co.jp.

Manuscript received 19 Jan. 2006; revised 5 June 2006; accepted 12 June 2006; published online 18 Jan. 2007.
Recommended for acceptance by S. Prabhakar, J. Kittler, D. Maltoni, L. O'Gorman, and T. Tan.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org and reference IEEECS Log Number TPAMISI-0029-0106.
Digital Object Identifier no. 10.1109/TPAMI.2007.1011.

0162-8828/07/$25.00 © 2007 IEEE Published by the IEEE Computer Society

TABLE 1
Hierarchy of Viola and Jones' Detector

Level	Method
Detector Structure	*Cascade Model*
Strong Classifier Learning	*AdaBoost Algorithm*
Weak Classifier Learning	*Stump Function*
Feature Space	*Haar-like Feature*

general applications (e.g., visual surveillance system, digital equipments that need autofocus on faces, etc.) as human faces in real-life images are seldom upright and frontal. Naturally, multiview face detection and rotation invariant face detection are defined to handle faces with ROP and RIP angles, respectively, (Fig. 1). However, both are more formidable problems because of the extension of detectable face range. In particular, the multiview face detection is more complicated than rotation invariant one since compared with frontal faces, profile faces tend to be less informative, more diverse, and more sensitive to noise. Moreover, as shown in Fig. 1, the ubiquitous concomitance of ROP and RIP of faces further compounds the difficulties to learn a face detector.

In recent years, there have been many works that developed new methods to enhance Viola and Jones' framework in some aspect. For instance, the detector structure has been extended to Li et al.'s pyramid model [9], Jones and Viola's decision tree [10], and Huang et al.'s Width-First-Search (WFS) tree [11] in order to cater to the multiview face detection, while Xiao et al.'s boosting chain [12] and Wu et al.'s nesting cascade model [13] transformed Viola and Jones' loose cascade model [7] into a more compact one. On the level of strong classifier learning, the original AdaBoost algorithm, which adopts binary-output predictors, was replaced by the superior Real AdaBoost [14] and Gentle Boost [15] that employ confidence-rated predictors. Moreover, a finer partition of the feature space was adopted to alleviate the weakness of stump function, e.g., Liu and Shum's histogram method [16], Wu et al.'s piece-wise function [13], and Mita et al.'s joint binarizations of Haar-like feature [17]. As for the level of feature space, there have been works of Lienhart and Maydt's extended Haar-like feature set [18], Liu and Shum's Kullback-Leibler features [16], Baluja's pair-wise points [19], Wang and Ji's RNDA algorithm [20], and Abramson and Steux's control point [21]. More details about these works can be found in the following sections, where comparisons are made.

In this paper, we aim at constructing a fast and accurate rotation invariant multiview face detector, which is capable of detecting faces with pose changes of $-/+90^\circ$ ROP (Yaw) and 360° RIP (Roll). Besides, the tolerance to $-/+30^\circ$ up-down (Pitch) change is combined to conform to surveillance environment. Our main contributions include the Width-First-Search (WFS) tree structure, the Vector Boosting algorithm, the sparse features in granular space, and the weak learner based on the heuristic search method. The remainder of this paper is organized as follows: Section 2 focuses on the comparison of existed detector structures and then introduces the WFS tree structure, Section 3 describes the Vector Boosting algorithm preceded by a brief review of the classical AdaBoost algorithm, Section 4 first discusses the effects of different types of features, then proposes sparse features in granular space and finally embodies the weak learner that adopts heuristic search method to train weak hypotheses for Vector Boosting, Section 5 shows related experiment results, and Section 6 gives the conclusion.

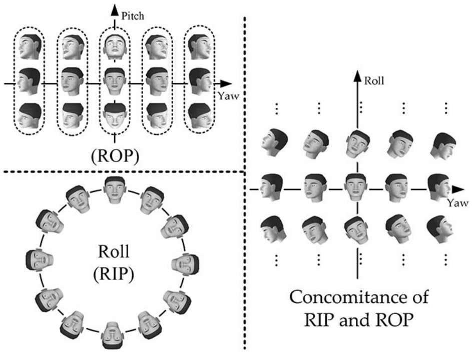


Fig. 1. Faces of RIP, ROP, respectively, and concomitance of both RIP and ROP. In this paper, multiview means yaw varies from -90° to $+90^\circ$, including composite pitch varying from -30° to $+30^\circ$, while rotation invariant denotes 360° rolling. Pitch change is omitted in the right figure.

2 Width-First-Search Tree Structure

2.1 Related Works

Viola and Jones' cascade structure [7], as shown in Fig. 2a, has been proven very efficient for dealing with rare event detection problems such as face detection because of its asymmetric decision-making process. However, such a succinct structure does not have enough capacity to handle multiview or rotation invariant face detection, both of which involve two distinct tasks: face detection and pose estimation. Face detection aims to distinguish faces from nonfaces, so it is inclined to utilize similarities between faces of different poses for rapid rejection of nonfaces; on the contrary, pose estimation is to identify the probable pose of a pattern no matter whether it is a face or not, so it seeks for diversities between different face poses but ignores the nonfaces.

Unifying or separating these two tasks will lead to different approaches. Osadchy et al.'s manifold method [22] could be taken as an example of the unified framework. A convolutional network was trained to map the face patterns to points on a face manifold, whose parameters indicated the pose variation, while nonface points were kept far away from the manifold. In this way, minimizing the energy function defined on the manifold was essentially a synchronous procedure to handle both tasks of multiview face detection. On the other hand, for the separated framework, the entire face range is usually divided into several individual categories according to their RIP or ROP angles. Such separated framework is also known as the view-based approach since each category usually refers to some certain view of faces.

The most straightforward way to implement a view-based approach was Wu et al.'s work [13], which trained different cascades individually for each view and used them in parallel as a whole like Fig. 2b. Though such a simple parallel-cascade strategy could achieve rather good performance in multiview face detection, the unexploited correlation between faces of different views suggested a large capacity of improvement through a better designed detector structure.

One way for improvement is the coarse-to-fine strategy. In [23], Fleuret and Geman employed a scalar tree detector (Fig. 2c) to adapt to large variation of location and scale of faces, while Li et al. [9] used pyramid to handle the great

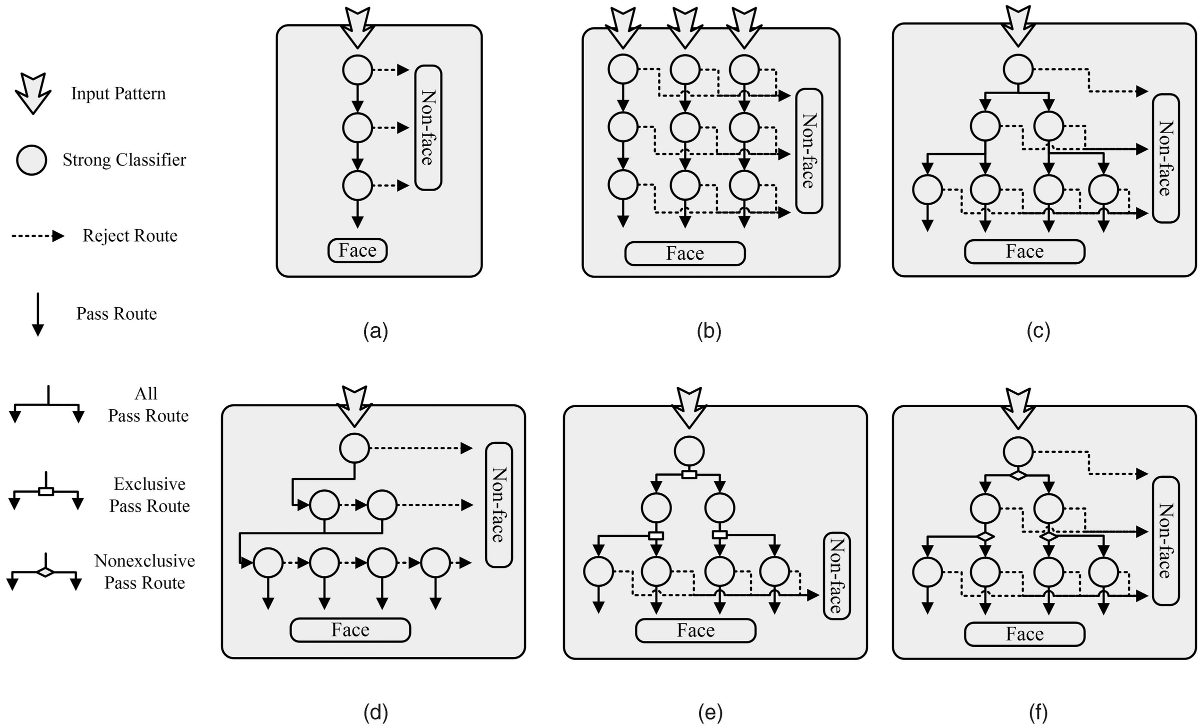


Fig. 2. Illustrations of different detector structures. (a) is Viola and Jones' original cascade structure for frontal face detection. Adopting view-based strategy to cope with the MVFD problem, several detector structures are developed, including (b) Wu et al.'s parallel cascade [13], (c) Fleuret and Geman's scalar tree [23], (d) Li et al.'s pyramid [9], (e) Jones and Viola's decision tree [10], and (f) Huang et al.'s Width-First-Search (WFS) tree [11]. Each leaf node in (d), (e), and (f) is actually an individual cascade as (a). Special attention should be paid to the difference among (c) scalar tree, (e) decision tree, and (f) WFS tree. Although they have got the similar tree-structures, their decision-making processes are distinct. Take the root node that splits into three child-nodes for example, the selection of pass route is uniform in scalar tree, exclusive in decision tree, and nonexclusive in WFS tree. Moreover, the scalar tree and WFS tree are able to make rejection decision at not only the leaf nodes but also the nonleaf ones. More details can be found in the following section.

change of appearance in MVFD (Fig. 2d). Though their structures were of a little difference, they both divided the complicated entire face space into finer and finer subspaces. In higher levels of their structure, neighboring views were assigned to a single node and, thus, corresponding faces were treated as one ensemble positive class so as to be separated from the nonfaces. This convenient combination did help to improve the efficiency and reusability of extracted features due to the similarities exist in faces of neighboring views, whereas neglected the inherent diversities between them (though they were neighboring views). As a result, a sample that had been identified as a face by a node would have to be processed by its every child-node, since it had no discrimination in corresponding views. In other words, the decision was uniform for child-nodes: either all active or all inactive. Such an all-pass route selection strategy considerably delayed the entire face identification procedure of input pattern.

On the contrary, another way for improvement, the decision tree method [10], put emphasis upon the diversities between different views. A decision tree was trained as a pose estimator to tell which view the input pattern belonged to, which was followed by individually learned cascade detectors for each view, respectively (Fig. 2e). A similar "pose estimation + detection" approach can be found in [29], which employed the support vector machine rather than the decision tree. With the imperative judgments made by the pose estimator, original complicated MVFD problem was reduced to several simple individual-views. Nevertheless, the pose estimation results were somewhat unstable, which weakened the generalization ability of the whole system. The instability should partly be attributed to the fact that face pose change was a continuous process rather than a discrete one. In fact, there must be a large number of face samples lying close to those artificially defined category boundaries, and, intuitively, the training of classifier with these "hard" boundaries often suffered from these ambiguous samples. From another point of view, fast and robust pose estimation applied before face detection is probably a problem that is even more difficult than the face detection itself.

To sum up, as mentioned at the beginning of this section, pose estimation focuses on the diversities of different views whereas face detection requires finding the similarities of different views to reject nonfaces as quickly as possible. Such conflict eventually leads to the dilemma that at the beginning of view-based approach, either treating all faces as a single class (the pyramid approach) or different individual classes (the decision tree approach) is unsatisfactory for the MVFD problem. Fortunately, a moderate approach, the Width-First-Search (WFS) tree, could be employed to harmonize the two tasks (pose estimation and face detection), balancing both aspects between different views (diversity and similarity).

2.2 WFS Tree-Structured Detector

In our approach to the rotation invariant multiview face detector described in Section 1, first a multiview face detector is constructed which covers the upright quarter of Roll and full Yaw, and then three more detectors are obtained by rotating the upright one by 90°, 180°, and 270° (Fig. 3). Such reduction from one rotation invariant detector to four

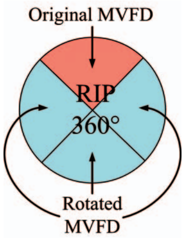


Fig. 3. Reduction of the rotation invariant multiview face detection. In fact, sparse features in the original multiview face detector are rotated by 90°, 180°, and 270°, which is equivalent to rotating the input image but more efficient to compute during detection.

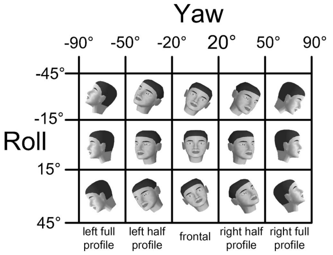


Fig. 4. View definition in the upright multiview face detector. The detector covers the face range of Roll from $-45^\circ \sim +45^\circ$ and Yaw from $-90^\circ \sim +90^\circ$, which is partitioned into 3×5 nonoverlapped views as shown above. Additional tolerance to Pitch variance from $-30^\circ \sim +30^\circ$ is embedded into each view.

quartered ones makes our system highly scalable (e.g., in applications where inverted faces are rarely encountered, the rotated detectors can be shut down easily). We further divide the upright quartered face space into 15 basic views according to Yaw and Roll variance (Fig. 4), and then empirically organize them as a tree illustrated in Fig. 5. The root node comprises all 15 views, which covers the entire quartered face space. In the coming two layers, according to Yaw angle, the root branching node is gradually partitioned into five disjointed ones. At last, in the bottom layer, these five branching nodes are split into 15 leaves according to Roll, attaining the finest 15 views. In such a tree-structured detector, an input pattern is identified as a face if and only if it passes at least one route from the root node to some certain leaf node. Therefore, the Width-First-Search (WFS) strategy is the right way to access every promising node of pass routes, whose pseudocode is shown as Fig. 6. Notice that for patterns that correlate with more than one view, the post pose estimation judges the final result according to the confidence of each view.

An extraordinary characteristic of the WFS strategy in Fig. 6 is the determinative vector $\mathbf{G}(\mathbf{x})$, each component of which decides whether the input pattern should be sent to the corresponding child-node or not. Compared with other related works listed in the last section, this determinative vector is much more versatile, neither restricted to be exclusive as Jones and Viola's decision tree [10] nor to be uniform as Fleuret and Geman's scalar tree [23] and Li et al.'s pyramid [9]. For instance, in the root branching node of Fig. 5, an input pattern making $\mathbf{G}(\mathbf{x}) = (1, 1, 0)$ indicates that it may be a left profile face or a frontal one but cannot be a right profile one, so in the following layer, it will be sent only to the left node and the middle one. Another pattern that has $\mathbf{G}(\mathbf{x}) = (0, 0, 0)$ is classified as outlying from any view of faces and, thus, will be rejected immediately. In fact, faces of different views are still considered as dissimilar categories in branching nodes of the WFS tree. However, these categories are not exclusive (e.g., as in the decision tree) but compatible with each other, meanwhile taking nonfaces as their collective negative class. In this way, the WFS tree not only utilizes the similarities between faces of different views to recognize nonfaces, but also reserves their diversities for further separation. Again take the root branching node in Fig. 5 for example, Table 2 compares different approaches at the aspect of pass route selection. The pose estimation made by the decision tree [10] is equivalent to a 3D determinative vector with only one nonzero component, and the pyramid structure [9], as well as the tree structure with scalar outputs [23], can only give a determinative vector with either all-zero components or all-one components. As for the WFS tree, it has the most diverse selections among all these approaches.

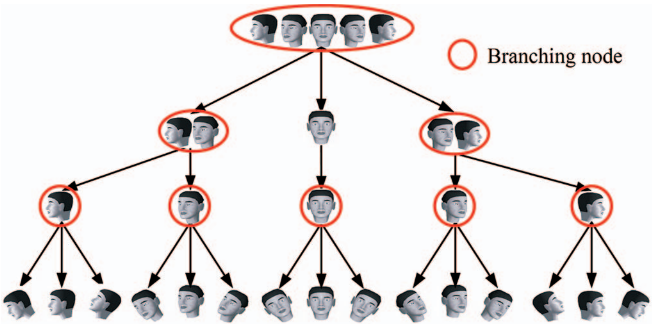


Fig. 5. Coarse-to-fine partition of face range in the tree-structured detector. The node with a red circle is a branching node, whose face range is the union of its child-nodes'. Among them, the root node contains all 15 views that are figured in the leaf nodes, of which the tilted 10 views are not displayed for clarity. The similar omissions are adopted for other branching nodes.

```
0. (Input)
        x: unidentified pattern
        T: tree detector
        r: root node of T

1. (Initialize)
        node list $L = \{r\}$, output list $Y = \varnothing$.

2. (WFS procedure)
  ■ While L is not empty, do
    ● Pop the first node d from L.
    ● Calculate the determinative vector
        $\mathbf{G}^{(d)}(\mathbf{x}) = \left[g_1^{(d)}(\mathbf{x}),\dots,g_n^{(d)}(\mathbf{x})\right],\quad g_i^{(d)}(\mathbf{x}) = \{1,0\}$
    ● For $t = 1,\dots,n$:
      ♦ If $g_t^{(d)}(\mathbf{x}) = 1$
            Get the t-th child node $s_i$ of d
            If $s_i$ is a leaf node
              Push the view label and the confidence for $s_i$
              into the output list Y.
            Else
              Push $s_i$ into node list L.
            End if
      ♦ End if
    ● End for
  ■ End do

3. (Post Pose Estimation & Output)
      Output the view label who has got the highest confidence
      in the output list Y as the final result for pattern x.
```

Fig. 6. Width-First-Search process in the tree-structured detector to identify an input pattern whether it is a face or not.

TABLE 2
Comparison of Different Approaches on Determinative Vectors for Pass Route Selection

	Output Space of Determinative Vector for Branching Nodes
Decision Tree [10] Structured Pose Estimator	(1,0,0), (0,1,0), (0,0,1)
Pyramid [9] or Tree [23] Structured Detector	(0,0,0), (1,1,1)
WFS Tree Structured Detector	(0,0,0), (1,0,0), (0,1,0), (0,0,1) (1,1,0), (1,0,1), (0,1,1), (1,1,1)

In conclusion, with the help of determinative vectors and nonexclusive pass route selection mechanism, the WFS tree structured detector is able to make moderate decision for the unidentified input pattern: neither too aggressive as decision tree nor excessively cautious as pyramid or scalar tree. Capabilities of different approaches in branching node (Table 2) explain the advantage of WFS tree in the flexibility of decision-making process.

3 Vector Boosting Algorithm

The Vector Boosting algorithm is developed to learn strong classifiers which can output the determinative vector $\mathbf{G}(\mathbf{x})$ of the WFS tree structure. Before elaborating on this novel method, we give a brief review on its origin—AdaBoost.

3.1 AdaBoost Algorithm

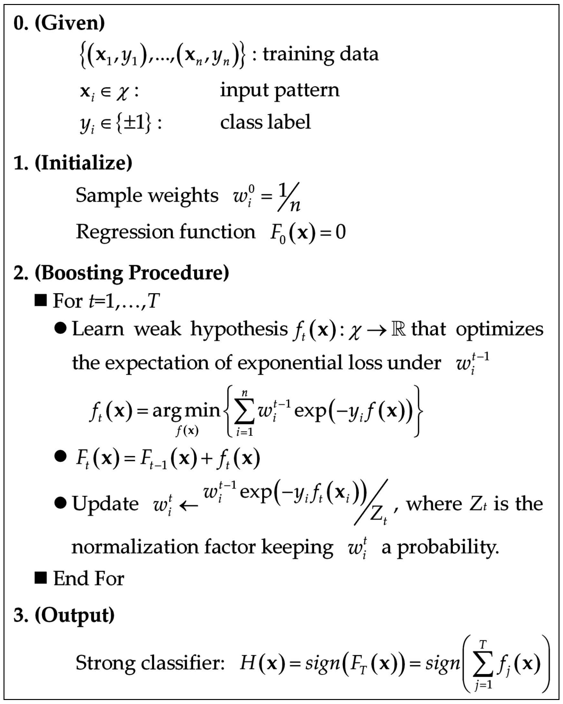
Boosting algorithm [24], which linearly combines a series of weak hypotheses to yield a superior classifier, has been regarded as one of the most significant developments in the pattern classification field during the past decade. It essentially employs an additive model to minimize the loss function of classification in a regressive manner. Consequently, different loss functions lead to different boosting algorithms. For example, AdaBoost [14], [24] and Gentle Boost [15] take exponential loss function as the optimization criterion, while LogitBoost [15] uses Bernoulli log-likelihood function. Moreover, BrownBoost [25] adopts a much more sophisticated loss function to enhance the robustness against outliers (noises) of training data. The classical AdaBoost algorithm can be formalized as shown in Fig. 7.

It is easy to verify that in the Adaboost algorithm, the weight of a sample satisfies

$$w_i^t = w_i^0 \exp(-y_i F_t(\mathbf{x}_i)) \Big/ \prod_{k=1}^{t} Z_k, \qquad (1)$$

where $y_i F_t(\mathbf{x}_i)$ is usually defined as the margin of sample $(\mathbf{x}_i, y_i)$ w.r.t $F_t(\mathbf{x})$. This weight-updating mechanism for samples is the kernel of AdaBoost, which makes the optimization procedure always emphasize those incorrectly classified samples by increasing their weights. By this means, AdaBoost manages to minimize the expectation of exponential loss of training samples

$$Loss(F(\mathbf{x})) = \sum_{i=1}^{n} w_i^0 \exp(-y_i F(\mathbf{x}_i)) = E\left(e^{-yF(\mathbf{x})}\right). \qquad (2)$$



```
0. (Given)
    $\{(\mathbf{x}_1,y_1),\dots,(\mathbf{x}_n,y_n)\}$ : training data
    $\mathbf{x}_i \in \chi$ :    input pattern
    $y_i \in \{\pm 1\}$ :    class label
1. (Initialize)
    Sample weights $w_i^0 = 1/n$
    Regression function $F_0(\mathbf{x}) = 0$
2. (Boosting Procedure)
  ■ For $t$=1,…,$T$
    ● Learn weak hypothesis $f_t(\mathbf{x}):\chi \to \mathbb{R}$ that optimizes
      the expectation of exponential loss under $w_i^{t-1}$
        $f_t(\mathbf{x}) = \arg\min_{f(\mathbf{x})} \left\{ \sum_{i=1}^{n} w_i^{t-1} \exp(-y_i f(\mathbf{x})) \right\}$
    ● $F_t(\mathbf{x}) = F_{t-1}(\mathbf{x}) + f_t(\mathbf{x})$
    ● Update $w_i^t \leftarrow w_i^{t-1} \exp(-y_i f_t(\mathbf{x}_i)) / Z_t$, where $Z_t$ is the
      normalization factor keeping $w_i^t$ a probability.
  ■ End For
3. (Output)
    Strong classifier: $H(\mathbf{x}) = sign(F_T(\mathbf{x})) = sign\left( \sum_{j=1}^{T} f_j(\mathbf{x}) \right)$
```

Fig. 7. A generalized version of the AdaBoost algorithm for two-class problems.

Essentially, (2) has a close relationship with the training error of the final classifier $H(\mathbf{x})$. It has been proved that the upper bound of training error is held by the loss defined in (2), which is equal to the product of every normalization factor [14]:

$$\frac{1}{n} \sum_{i=1}^{n} [\![y_i \neq H(\mathbf{x}_i)]\!] \leq \sum_{i=1}^{n} w_i^0 \exp(-y_i F(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t. \qquad (3)$$

Therefore, AdaBoost is actually an iterative procedure to greedily reduce the upper bound of training error. Although the generalization bound given in [14] is often too loose to make sense in practice, this algorithm has shown satisfactory performance in many practical problems.

3.2 Vector Boosting Algorithm

As extensions of the AdaBoost algorithm, AdaBoost.MH, AdaBoost.MO, and AdaBoost.MR [14] deal with multiclass problems with different definitions of loss functions. AdaBoost.MH assigns a label set for each sample and adopts the exponential loss of symmetric difference between the label set and the output of the strong classifier, and AdaBoost.MO makes use of the output code technique to generalize AdaBoost.MH. On the other hand, AdaBoost.MR treats the multiclass problem as a ranking problem and uses ranking loss in expectation that the correct labels could receive the highest ranks. Although these multiclass boosting algorithms have been successfully applied in many problems, they are still not directly applicable to the problem corresponding to branching nodes of the WFS tree, in which faces of different views are neither coherent nor disperse but nonexclusive. Therefore, the Vector Boosting algorithm, as a unified boosting framework, is developed for the learning of branching nodes, which manipulates different kinds of multiclass problems by means of the vectorization of

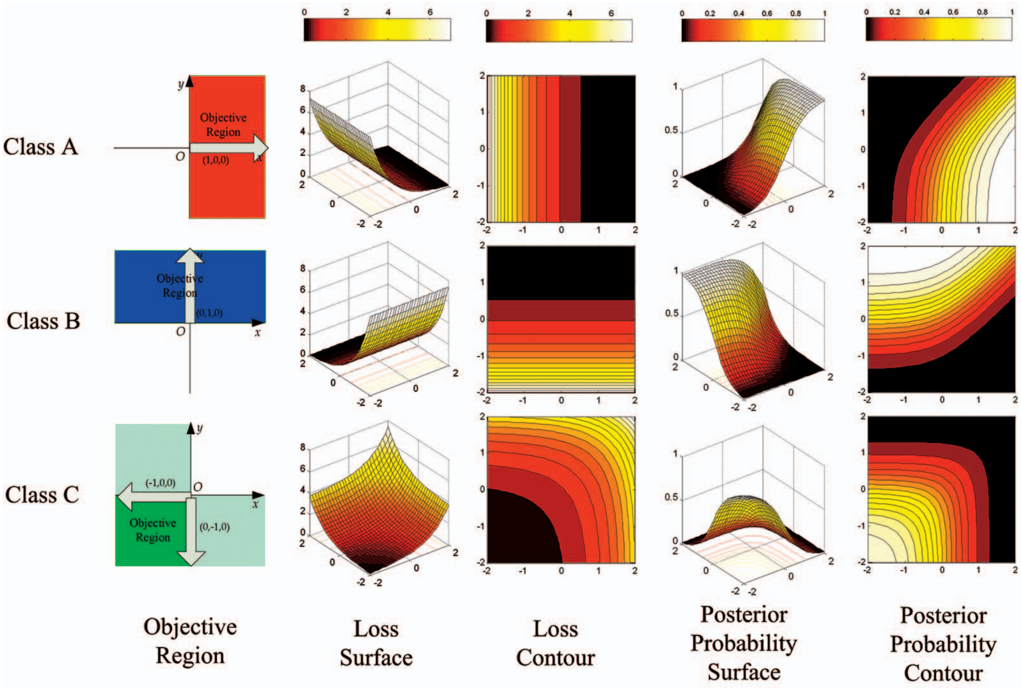


Fig. 8. A naive example of a three-class problem. Three rows correspond to three classes, respectively. The first column shows their different objective regions, which are convex sets and defined by intrinsic projection vectors (white arrows). According to (4), $C_A = \{(x, y), x \geq 0\}$, $C_B = \{(x, y), y \geq 0\}$, and $C_C = \{(x, y) : x \leq 0, y \leq 0\}$. The second and the third columns draw their losses in 2D output space for each category by surface and contour, which are $\exp(-x)$ for Class A, $\exp(-y)$ for Class B, and $\exp(x) + \exp(y)$ for Class C. The fourth and the fifth columns draw their posterior probabilities also by surface and contour, which are employed to explicitly calculate the decision boundaries in Section 3.2.4.

hypothesis output space and the flexible loss function defined by intrinsic projection vectors.

3.2.1 Convex Objective Region and Exponential Loss Function

The Vector Boosting algorithm originates from the motivation of decomposing a complicated multiclass problem into a set of simple ones, making them share the same features and calculating their respective outputs. For this purpose, it assigns different categories with different convex objective regions in the vector hypothesis output space $\mathbb{R}^k$ so as to make them distinguishable. These convex objective regions are defined as the intersection of one or more half spaces in homogeneous coordinates

$$\begin{aligned} &C = \{\tilde{z} :\ \forall \tilde{\mathbf{v}} \in V,\ \tilde{z} \cdot \tilde{\mathbf{v}} \geq 0\} \\ &\tilde{z} = (z, 1),\ \tilde{\mathbf{v}} = (\mathbf{v}, b), \mathrm{z} \in \mathbb{R}^k,\ V = \{\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_m\}. \end{aligned} \tag{4}$$

The extended vector $\tilde{\mathbf{v}}$, composed of normal vector $\mathbf{v}$ and offset b, specifies a half space that supports the convex objective region C. We name it *intrinsic projection vector* as it plays an important role in the loss function introduced later. The first column of Fig. 8 gives a naive example to show the usage of intrinsic projection vectors and the consequent convex objective regions. Note that it is unnecessary to require different objective regions to be nonoverlapped or have their union cover the entire hypothesis output space. Apparently, a hypothesis output $\mathbf{F}(\mathbf{x})$ lies in the half space specified by an intrinsic projection vector $\tilde{\mathbf{v}}$ if their inner product is nonnegative. So, the margin of a hypothesis output with regard to an intrinsic projection vector is defined as

$$margin(\mathbf{F}(\mathbf{x}), \tilde{\mathbf{v}}) = \widetilde{\mathbf{F}}(\mathbf{x}) \cdot \tilde{\mathbf{v}}, \tag{5}$$

where $\widetilde{\mathbf{F}}(\mathbf{x}) = (\mathbf{F}(\mathbf{x}), 1)$ is the extended hypothesis output in homogeneous coordinate. Furthermore, a hypothesis output lies in a convex objective region if and only if its margin with regard to every intrinsic projection vector is nonnegative. Ideally, a perfectly learned hypothesis maps every input pattern onto its corresponding objective region. Therefore, enlightened by the exponential loss adopted in AdaBoost algorithm, to penalize input patterns who have not been mapped onto the correct objective regions, the loss function in the Vector Boosting algorithm is defined as follows:

$$\begin{aligned} Loss(\mathbf{F}(\mathbf{x})) &= E\left(\sum_{\tilde{\mathbf{v}}_j \in V(\mathbf{x})} \exp\big(-margin\big(\mathbf{F}(\mathbf{x}), \tilde{\mathbf{v}}_j\big)\big)\right) \\ &= E\left(\sum_{\tilde{\mathbf{v}}_j \in V(\mathbf{x})} \exp\Big(-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \tilde{\mathbf{v}}_j\Big)\right), \end{aligned} \tag{6}$$

where $V(\mathbf{x})$is the intrinsic projection vector set of pattern $\mathbf{x}$. The second and the third columns in Fig. 8 draw the loss function of each class individually. In the training process, the expectation in (6) becomes the sum of losses absorbed from training samples as follows:

$$Loss(\mathbf{F}(\mathbf{x})) = \frac{1}{n} \sum_{i=1}^{n} \left\{ \sum_{\tilde{\mathbf{v}}_j \in V(\mathbf{x}_i)} \exp\Big[-\tilde{\mathbf{v}}_j \cdot \widetilde{\mathbf{F}}(\mathbf{x}_i)\Big] \right\}, \tag{7}$$

where n is the number of training samples. It is easy to find that a training sample with q intrinsic projection vectors is equivalent to q training samples each with one intrinsic projection vector: $(\mathbf{x}_i, \{\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_q\}) \overset{eq}{\rightleftharpoons} \{(\mathbf{x}_i, \tilde{\mathbf{v}}_1), \ldots, (\mathbf{x}_i, \tilde{\mathbf{v}}_q)\}$.

Based on this observation, in practice a training sample with more than one intrinsic projection vector will be expanded into a set of samples each with only one intrinsic projection vector. Without loss of generality, throughout the formalization of optimization procedure in the next section, we employ a minified loss function based on the expanded training samples as

$$\begin{aligned}&\widehat{Loss}(\mathbf{F}(\mathbf{x})) = \frac{1}{m}\sum_{i=1}^{m}\exp\left(-\widetilde{\mathbf{v}}_i\cdot\widetilde{\mathbf{F}}(\mathbf{x}_i)\right) = \\ &\frac{n}{m}\times\frac{1}{n}\sum_{i=1}^{m}\exp\left(-\widetilde{\mathbf{v}}_i\cdot\widetilde{\mathbf{F}}(\mathbf{x}_i)\right) = \frac{n}{m}Loss(\mathbf{F}(\mathbf{x})),\quad n\le m,\end{aligned}\tag{8}$$

where n is the number of original training samples while m is the number of expanded ones. As soon as the training samples and their intrinsic projection vectors are given, the minification n/m is determined. Therefore, using the minified loss function in (8) instead of the original one in (7) will make no difference on the final optimization results.

3.2.2 Optimization Procedure

Like other boosting algorithms, the Vector Boosting employs an additive model to minimize the loss function defined in the previous section. Suppose a strong hypothesis $\mathbf{F}(\mathbf{x})$ has been obtained in the additive model, the next step is to learn an optimal weak hypothesis $\mathbf{f}(\mathbf{x})$ to add in. According to the minified loss function in (8), the overall training loss turns into

$$\begin{aligned}\widehat{Loss}(\mathbf{F}(\mathbf{x})+\mathbf{f}(\mathbf{x})) &= \frac{1}{m}\sum_{i=1}^{m}\exp\left[-\left(\widetilde{\mathbf{v}}_i\cdot\widetilde{\mathbf{F}}(\mathbf{x}_i)+\mathbf{v}_i\cdot\mathbf{f}(\mathbf{x}_i)\right)\right] \\ &= \sum_{i=1}^{m}\frac{e^{-\widetilde{\mathbf{v}}_i\cdot\widetilde{\mathbf{F}}(\mathbf{x}_i)}}{m}\exp[-\mathbf{v}_i\cdot\mathbf{f}(\mathbf{x}_i)] \propto \sum_{i=1}^{m}w_i\exp[-\mathbf{v}_i\cdot\mathbf{f}(\mathbf{x}_i)],\end{aligned}\tag{9}$$

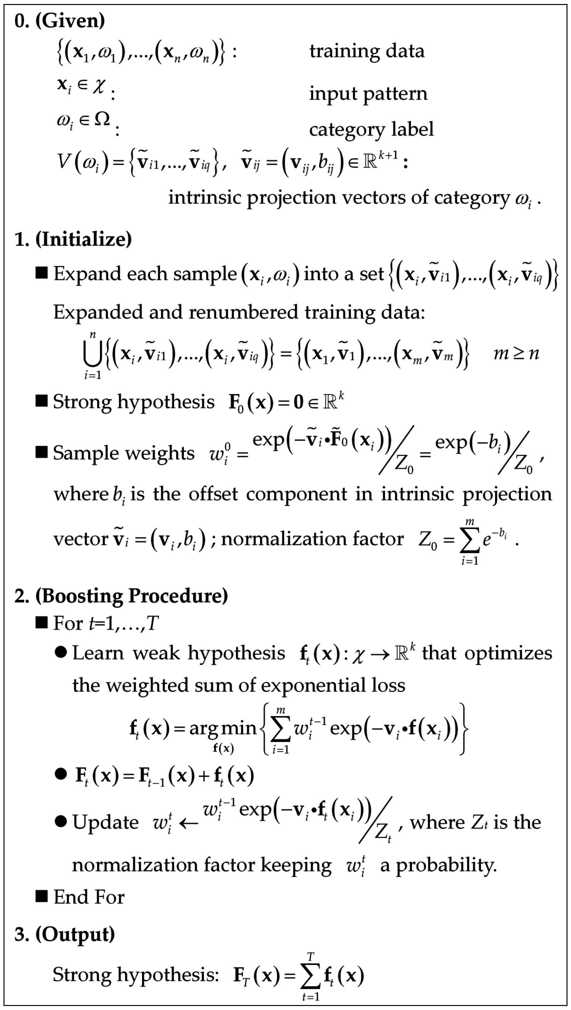
where $\widetilde{\mathbf{v}}_j=(\mathbf{v}_j,b_j)$ and $w_i\propto e^{-\widetilde{\mathbf{v}}_i\cdot\widetilde{\mathbf{F}}(\mathbf{x}_i)}/m$ (w_i is the precalculated weight for $\mathbf{x}_i$). This equation indicates that the loss absorbed from $\mathbf{x}_i$ by $\mathbf{F}(\mathbf{x})$ actually works as a prior weight of the sample during the optimization of the new weak hypothesis $\mathbf{f}(\mathbf{x})$. Thus, the optimal weak hypothesis should be

$$\begin{aligned}\mathbf{f}^*(\mathbf{x}) &= \arg\min_{\mathbf{f}(\mathbf{x})}(Loss(\mathbf{F}(\mathbf{x})+\mathbf{f}(\mathbf{x}))) \\ &= \arg\min_{\mathbf{f}(\mathbf{x})}\left\{\sum_{i=1}^{m}w_i\exp(-\mathbf{v}_i\cdot\mathbf{f}(\mathbf{x}_i))\right\}.\end{aligned}\tag{10}$$

This is indeed a weight-updating mechanism similar to other boosting algorithms. In this way, the Vector Boosting can be formalized as shown in Fig. 9. Notice that weights of training samples are always normalized to be a probability, even for w_0 initialized at the very beginning before any weak hypothesis is adopted. As the initial hypothesis output $\mathbf{F}_0(\mathbf{x})$ is zero, for any sample $(\mathbf{x}_i,\widetilde{\mathbf{v}}_i)$, its prior probability at the first round is determined only by the offset b_i as

$$w_i^0\propto\exp\left(-\widetilde{\mathbf{v}}_i\cdot\widetilde{\mathbf{F}}_0(\mathbf{x}_i)\right) = e^{-(\mathbf{v}_i,b_i)\cdot(0,\ldots,0,1)} = e^{-b_i},\tag{11}$$

which is distinct from the classical AdaBoost algorithm [14].



```
0. (Given)
    $\{(\mathbf{x}_1,\omega_1),\ldots,(\mathbf{x}_n,\omega_n)\}$ :    training data
    $\mathbf{x}_i\in\chi$ :    input pattern
    $\omega_i\in\Omega$ :    category label
    $V(\omega_i)=\{\widetilde{\mathbf{v}}_{i1},\ldots,\widetilde{\mathbf{v}}_{iq}\}$,  $\widetilde{\mathbf{v}}_{ij}=(\mathbf{v}_{ij},b_{ij})\in\mathbb{R}^{k+1}$ :
        intrinsic projection vectors of category $\omega_i$.
1. (Initialize)
    ■ Expand each sample $(\mathbf{x}_i,\omega_i)$ into a set $\{(\mathbf{x}_i,\widetilde{\mathbf{v}}_{i1}),\ldots,(\mathbf{x}_i,\widetilde{\mathbf{v}}_{iq})\}$
      Expanded and renumbered training data:
        $\bigcup_{i=1}^{n}\{(\mathbf{x}_i,\widetilde{\mathbf{v}}_{i1}),\ldots,(\mathbf{x}_i,\widetilde{\mathbf{v}}_{iq})\}=\{(\mathbf{x}_1,\widetilde{\mathbf{v}}_1),\ldots,(\mathbf{x}_m,\widetilde{\mathbf{v}}_m)\}$    $m\ge n$
    ■ Strong hypothesis  $\mathbf{F}_0(\mathbf{x})=\mathbf{0}\in\mathbb{R}^k$
    ■ Sample weights  $w_i^0=\exp(-\widetilde{\mathbf{v}}_i\bullet\widetilde{\mathbf{F}}_0(\mathbf{x}_i))/Z_0=\exp(-b_i)/Z_0$,
      where $b_i$ is the offset component in intrinsic projection
      vector $\widetilde{\mathbf{v}}_i=(\mathbf{v}_i,b_i)$; normalization factor  $Z_0=\sum_{i=1}^{m}e^{-b_i}$.
2. (Boosting Procedure)
    ■ For $t$=1,…,$T$
        ● Learn weak hypothesis  $\mathbf{f}_t(\mathbf{x}):\chi\to\mathbb{R}^k$ that optimizes
          the weighted sum of exponential loss
            $\mathbf{f}_t(\mathbf{x})=\arg\min_{\mathbf{f}(\mathbf{x})}\left\{\sum_{i=1}^{m}w_i^{t-1}\exp(-\mathbf{v}_i\bullet\mathbf{f}(\mathbf{x}_i))\right\}$
        ● $\mathbf{F}_t(\mathbf{x})=\mathbf{F}_{t-1}(\mathbf{x})+\mathbf{f}_t(\mathbf{x})$
        ● Update  $w_i^t\leftarrow w_i^{t-1}\exp(-\mathbf{v}_i\bullet\mathbf{f}_t(\mathbf{x}_i))/Z_t$, where $Z_t$ is the
          normalization factor keeping  $w_i^t$  a probability.
    ■ End For
3. (Output)
    Strong hypothesis:  $\mathbf{F}_T(\mathbf{x})=\sum_{t=1}^{T}\mathbf{f}_t(\mathbf{x})$
```

Fig. 9. k-dimensional Vector Boosting algorithm.

3.2.3 Bound of Training Error

From the view of the defined objective regions, a training sample $(\mathbf{x}_i,\omega_i)$ is correctly classified by $\mathbf{F}(\mathbf{x})$ if and only if $\mathbf{F}(\mathbf{x}_i)\in C(\omega_i)$, where $C(\omega_i)$ is the objective region of category ω_i. Here, we adopt a characteristic function

$$B(\mathbf{x}_i,\omega_i)=\begin{cases}0, & \text{if } \forall\widetilde{\mathbf{v}}_{ij}\in V(\omega_i),\ \widetilde{\mathbf{F}}(\mathbf{x}_i)\cdot\widetilde{\mathbf{v}}_{ij}\ge 0\\ 1, & \text{else,}\end{cases}\tag{12}$$

where $V(\omega_i)$ is the projection vector set that defines $C(\omega_i)$. Then, the training error of $\mathbf{F}(\mathbf{x})$ is

$$P_{error}=\frac{1}{n}\sum_{i=1}^{n}[\![\mathbf{F}(\mathbf{x}_i)\notin C(\omega_i)]\!]=\frac{1}{n}\sum_{i=1}^{n}B(\mathbf{x}_i,\omega_i).\tag{13}$$

Since $B(\mathbf{x}_i,\omega_i)\le\sum_{\widetilde{\mathbf{v}}_{ij}\in V(\omega_i)}\exp(-\widetilde{\mathbf{F}}(\mathbf{x}_i)\cdot\widetilde{\mathbf{v}}_{ij})$, the training error in (13) has an upper bound that

$$P_{error}\le\frac{1}{n}\sum_{i=1}^{n}\left\{\sum_{\widetilde{\mathbf{v}}_{ij}\in V(\omega_i)}\exp\left(-\widetilde{\mathbf{F}}(\mathbf{x}_i)\cdot\widetilde{\mathbf{v}}_{ij}\right)\right\}=Loss(\mathbf{F}(\mathbf{x})).\tag{14}$$

The equation on the right side holds due to the definition of training loss in (7).

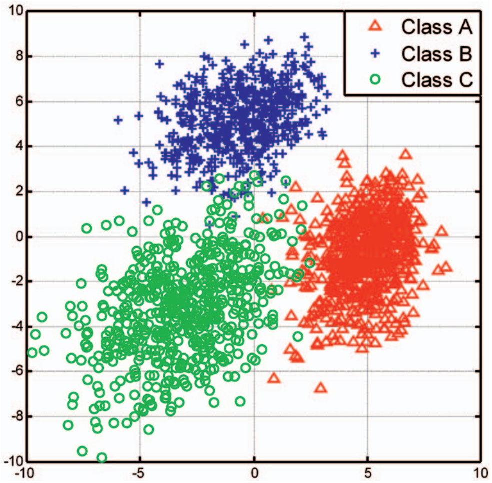


Fig. 10. The results of strong hypothesis learned by Vector Boosting for the naive three-class problem depicted in Fig. 8. In this experiment, Class A and Class B are left and right profile face categories, while Class C is the nonface category. The size of training sample is 24×24 and $\mathbf{F}(\mathbf{x}) : \mathrm{I\!R}^{576} \rightarrow \mathrm{I\!R}^{2}$.

On the other hand, after the Tth round of boosting procedure, sample weights should satisfy

$$1 = \sum_{i=1}^{m} w_i^T = \sum_{i=1}^{m} \frac{w_i^{T-1} \exp(-\mathbf{v}_i \cdot \mathbf{f}_T(\mathbf{x}_i))}{Z_T}$$
$$= \sum_{i=1}^{m} w_i^0 \prod_{j=1}^{T} \frac{\exp\bigl(-\mathbf{v}_i \cdot \mathbf{f}_j(\mathbf{x}_i)\bigr)}{Z_j} = \frac{\sum_{i=1}^{m} \exp\Bigl(-\widetilde{\mathbf{v}}_i \cdot \widetilde{\mathbf{F}}_T(\mathbf{x}_i)\Bigr)}{\prod_{j=0}^{T} Z_j}. \tag{15}$$

Associating it with (8), we have

$$\prod_{j=0}^{T} Z_j = \sum_{i=1}^{m} \exp\Bigl(-\widetilde{\mathbf{v}}_i \cdot \widetilde{\mathbf{F}}_T(\mathbf{x}_i)\Bigr) \tag{16}$$
$$= m \times \widehat{Loss}(\mathbf{F}_T(\mathbf{x})) = n \times Loss(\mathbf{F}_T(\mathbf{x})).$$

Using (16) to replace the right part of (14), we have:

$$P_{error} \leq \frac{1}{n} \prod_{j=0}^{T} Z_j. \tag{17}$$

Hence, the training error of Vector Boosting is bounded by the product of every normalization factor. Like other boosting algorithms, the Vector Boosting algorithm can train a strong classifier with low training error by greedily minimizing the normalization factor Z_t of each round.

3.2.4 Decision Boundary

Fig. 10 illustrates the power of a strong hypothesis $\mathbf{F}(\mathbf{x})$, which is learned with Vector Boosting under the configurations depicted in Fig. 8 to solve the naive 3-class problem. According to the distributions of samples, these three categories are fairly well separated in the 2D hypothesis output space. But, for classification task, one more thing must be clarified: What are the optimal decision boundaries for different categories in the hypothesis output space. Directly adopting objective regions as the decision boundaries makes sense in the analysis of training error bound, but this straightforward criterion might be unsuitable to practical cases. For example, the objective regions may have nonempty intersection (e.g., Class A and Class B in Fig. 8) and there may be some area in the output space corresponding to none of the objective regions. In fact, analytical decision boundaries in Vector Boosting are nontrivial due to the flexibility of intrinsic projection vectors and the complexity of the consequent loss function. However, with a sustainable assumption, an approximate optimal decision boundary is achievable.

Derived from the deduction in [15], the loss function in (6) can be considered as the expectation over the joint distribution of input pattern $\mathbf{x}$ and class label ω. It is sufficient for optimization procedure in Fig. 9 to minimize the criterion conditionally on variable $\mathbf{x}$ as

$$E\left(\sum_{\widetilde{\mathbf{v}}_j \in V(\mathbf{x})} \exp(-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \widetilde{\mathbf{v}}_j) \middle| \mathbf{x}\right) = \int \left(\sum_{\widetilde{\mathbf{v}}_j \in V(\mathbf{x})} \exp(-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \widetilde{\mathbf{v}}_j)\right)$$
$$p(\omega|\mathbf{x}) d\omega = \sum_{i=1}^{c} \left\{ P(\omega_i|\mathbf{x}) \sum_{\widetilde{\mathbf{v}}_j \in V(\omega_i)} \exp(-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \widetilde{\mathbf{v}}_j) \right\}, \tag{18}$$

where c is the number of categories. Notice that this conditional expectation has been rewritten as the sum weighted by the posterior probabilities of each category. During the optimization procedure in the Vector Boosting algorithm, the derivative of the conditional loss function with respect to $\mathbf{F}(\mathbf{x})$ approaches zero as more and more weak hypotheses are adopted into the strong hypothesis. Thus, the derivative can be assumed as 0 if the hypothesis is adequately optimized.

$$\frac{\partial E\left(\sum_{\widetilde{\mathbf{v}}_j \in V(\mathbf{x})} \exp(-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \widetilde{\mathbf{v}}_j) \middle| \mathbf{x}\right)}{\partial \mathbf{F}(\mathbf{x})} \tag{19}$$
$$= \sum_{i=1}^{c} \left\{ P(\omega_i|\mathbf{x}) \sum_{\widetilde{\mathbf{v}}_j \in V(\omega_i)} \left(-e^{-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \widetilde{\mathbf{v}}_j} \mathbf{v}_j\right) \right\} = -\mathbf{A}\mathbf{P} = \mathbf{0},$$

where

$$\mathbf{A} = \left[\sum_{\widetilde{\mathbf{v}}_j \in \widetilde{V}(\omega_1)} e^{-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \widetilde{\mathbf{v}}_j} \mathbf{v}_j \quad \cdots \quad \sum_{\widetilde{\mathbf{v}}_j \in V(\omega_c)} e^{-\widetilde{\mathbf{F}}(\mathbf{x}) \cdot \widetilde{\mathbf{v}}_j} \mathbf{v}_j \right]$$

and the column vector

$$\mathbf{P} = \begin{bmatrix} P(\omega_1|\mathbf{x}) \\ \vdots \\ P(\omega_c|\mathbf{x}) \end{bmatrix}.$$

In addition, since the sum of all posterior probabilities is 1, one more equation could be added into (19) as an additional constraint, resulting in a linear equation set with c unknowns as

$$\begin{bmatrix} \mathbf{A} \\ \mathbf{1}^T \end{bmatrix} \mathbf{P} = \mathbf{A}'\mathbf{P} = [0 \quad \cdots \quad 0 \quad 1]^T, \tag{20}$$

where matrix $\mathbf{A}'$ has $k+1$ rows and c columns (k is the dimension of the hypothesis output space and c is the number of categories). If $c = k+1$ and $\mathbf{A}'$ is nonsingular,

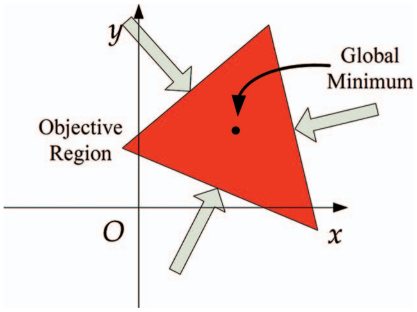


Fig. 11. Global minimum of loss function for a category that has closed objective region. White arrows are its intrinsic projection vectors and black point is at the global minimum.

the solution of P in (20) is unique. Take the naive configuration in Fig. 8 for example, we have:

$$\mathbf{P} = \begin{bmatrix} P(\omega_A|\mathbf{x}) \\ P(\omega_B|\mathbf{x}) \\ P(\omega_C|\mathbf{x}) \end{bmatrix} \text{ and } \mathbf{A}' = \begin{bmatrix} e^{-F_x(\mathbf{x})} & 0 & -e^{F_x(\mathbf{x})} \\ 0 & e^{-F_y(\mathbf{x})} & -e^{F_y(\mathbf{x})} \\ 1 & 1 & 1 \end{bmatrix}, \tag{21}$$

where $F_x(\mathbf{x})$ is the output of $\mathbf{F}(\mathbf{x})$ on x-axis and $F_y(\mathbf{x})$ is that on y-axis. The solution is:

$$\begin{aligned} P(\omega_C|\mathbf{x}) &= \frac{1}{1+\exp\bigl(2F_x(\mathbf{x})\bigr)+\exp\bigl(2F_y(\mathbf{x})\bigr)}, \\ P(\omega_A|\mathbf{x}) &= \exp(2F_x(\mathbf{x}))P(\omega_C|\mathbf{x}), \\ P(\omega_B|\mathbf{x}) &= \exp\bigl(2F_y(\mathbf{x})\bigr)P(\omega_C|\mathbf{x}), \end{aligned} \tag{22}$$

which are illustrated in the fourth and the fifth columns of Fig. 8. Accordingly, optimal decision boundaries are curves shown in the contour map of each posterior probability rather than borderlines of objective regions.

3.3 Brief Summary

Essentially, the Vector Boosting algorithm is a generalized framework for multiclass problems with additive regression model. Its loss function is defined based on intrinsic projection vectors of different categories. With proper configuration of those vectors, the Vector Boosting could be transformed into many other existing boosting algorithms due to their consistent loss functions. For instance, if the output space is one-dimensional, and intrinsic projection vectors for two classes are $(1, 0)$ and $(-1, 0)$, respectively, the loss function of Vector Boosting becomes that of the classical AdaBoost. Similarly, the Vector Boosting can be transformed into AdaBoost.MH, AdaBoost.MO, and AdaBoost.MR [14]. Moreover, if configuring an objective region as a closed convex set rather than an open one, the global minimum of loss function for its corresponding category will transfer from infinite to an inner point of the close convex set (Fig. 11), and the loss function in (7) will be equivalent to that of ExpLev algorithm introduced by Duffy and Helmbold [26]. This implies that Vector Boosting algorithm is not only a classification framework, but also of high potential to deal with regression problems upon the widely-adopted exponential loss criterion.

Practically, to learn branching nodes of the WFS tree, similar configurations as Fig. 8 are adopted, in which different face categories are assigned with orthogonal intrinsic projection vectors and the nonface category with opposite ones. As an example, Table 3 shows the setting for the root node. In this way, different face categories are irrelevant in the loss function of Vector Boosting but share a common opposite, the nonface category, which accords with the nonexclusive pass route selection criterion proposed in the WFS tree. Differently, as long as face views are allocated with different label sets, AdaBoost.MH, AdaBoost.MO, and AdaBoost.MR will have to pay much effort to separate them, which is absolutely unnecessary in the WFS tree. A comparative experiment between the Vector Boosting and AdaBoost.MH is given in Section 5.

TABLE 3
Configuration for the Training of Root Node in the WFS Tree

Class		Intrinsic Projection Vectors
Faces	Left Profile	(1,0,0,0)
	Frontal	(0,1,0,0)
	Right Profile	(0,0,1,0)
Non-faces		(-1,0,0,0),(0,-1,0,0),(0,0,-1,0)

After the transformation from the hypothesis outputs to the posterior probabilities as (22), optimal thresholds can be found according to the required detection rate or false alarm rate. In this way, a strong classifier that outputs determinative vectors for branching nodes can be learned, and the transformed posterior probabilities can be treated as the output confidences for the selection of corresponding pass route in Fig. 6.

4 Learning Sparse Granular Features for Domain-Partition Weak Hypotheses

Conventional feature extraction methods in face detection, such as Rowley et al.'s ANN [5], are directly based on high-dimensional input patterns. Although discriminative low-dimensional feature space could be obtained by means of PCA, LDA, or RNDA [20], they usually involve full scale vector inner product that is computational intensive. One of the key issues in Viola and Jones' system [7] which leads to their success is the integral image, which helps to compute Haar-like features much more efficiently. However, their Haar-like features are often deficient in distinguishing complicated and irregular patterns such as profile faces due to their rigorous structural constraints. To alleviate this difficulty, Li et al. [9] and Lienhart and Maydt [18] employ extended Haar-like features that include relatively shiftable and rotated ones, respectively. Further more, Baluja et al. [19] and Abramson and Steux [21] adopt pixel-based features to achieve more flexible form and sparser presentation compared with Haar-like features. Both of them use logic operators (i.e., whether or not pixel A is brighter than pixel B) to discriminate input patterns, avoiding normalization of mean and standard deviation of samples which is necessary for Haar-like features. As a result, these pixel-based features are extremely fast to compute but unfortunately not discriminative and robust enough. In our face detection system, a set of novel features are sparsely represented in the granular space of the input image, and an efficient weak learning algorithm is introduced which adopts heuristic search method in pursuit of discriminative sparse granular features.

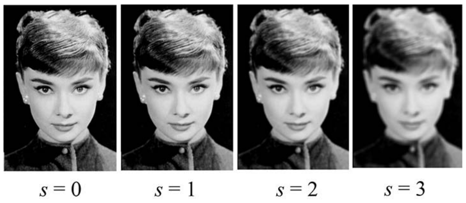


Fig. 12. Granular space of a gray image (s denotes the scale of granules).

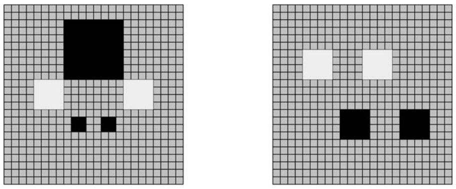


Fig. 13. Two examples of sparse features. White blocks are positive granules while black ones are negative. Calculating each granule needs to access memory only once.

4.1 Sparse Features Represented in Granular Space

The granular space as shown in Fig. 12 is made up of four bitmaps: I_0, I_1, I_2, and I_3. Denote the scale variable as s ($s = \{0,1,2,3\}$), each granular bitmap I_s is the result of smooth filtering in way of averaging over $2^s \times 2^s$ patches of the original image. Therefore, a granule $I_s(x,y)$ can be specified by the x-offset, y-offset, and the scale s. In such a granular space, a sparse feature is represented as the linear combination of several granules as

$$\theta = \sum_i \alpha_i I_{s_i}(x_i, y_i),\ \alpha_i \in \{-1,+1\},\ s_i \in \{0,1,2,3\}, \quad (23)$$

where the combining coefficient a_i is restricted to be binary value for the sake of computational efficiency (Fig. 13). Once the granular space is constructed, calculating a granule needs to access memory only once rather than four times for a rectangle of Haar-like features. Therefore, compared with the Haar-like features [7] as well as their extended versions in [9], [18], the sparse granular features are highly scalable: they can be more versatile while keeping the same computation load, or more economic to compute if keeping similar structural complexity. Moreover, in order to increase robustness and discriminability of sparse granular features, the integral image is retained to apply the normalization of mean and standard deviation like Haar-like features. Based on such normalized features, stronger weak classifiers could be learned instead of logic-operator approaches with those unnormalized features in [19], [21].

4.2 Domain-Partition-Based Weak Learner for Vector Boosting Algorithm

Weak learner in boosting algorithms aims to train a proper weak hypothesis to reduce the training loss of strong classifier as it holds the upper bound of training error. In our approach, a weak hypothesis could be decoupled into two parts: the first part is extracting a sparse granular feature from input pattern; the second part is calculating the prediction result through a piece-wise function. In the following parts, a weak hypothesis $\mathbf{f}(\mathbf{x};\theta,\mu)$ with input pattern x is characterized by two parameters: θ for sparse feature and μ for piece-wise function.

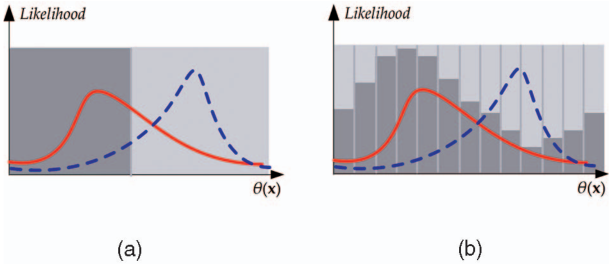


Fig. 14. Stump function versus piece-wise function. The stump function only divides the 1D feature space into two parts with an adjustable threshold, giving binary outputs, while the piece-wise function partitions the feature space in finer granularity and outputs various values for each bin. (a) Stump function. (b) Piece-wise function.

4.2.1 Learning Piece-Wise Functions for Selected 1D Features in Vector Boosting Algorithm

The piece-wise function μ, illustrated in Fig. 14b, divides the 1D feature space into a set of disjoint bins with equal widths, and outputs a constant value (scalar or vector) for samples falling into the same bin. Actually, it is a straightforward implementation of domain partition based hypothesis in [14], which is superior to stump function (shown in Fig. 14a) adopted in [7] since it is capable to fit likelihoods more precisely through finer partition granularity. Three parameters are necessary to determine the partition of a chosen 1D feature space: the lower bound, the upper bound, and the granularity. The first two are estimated through distributions of training samples on the chosen feature, and the last one, granularity, is predefined by experience.

Denote the samples that are grouped into the jth bin as

$$S_j = \{(\mathbf{x}_i, \tilde{\mathbf{v}}_i) | \theta(\mathbf{x}_i) \in bin_j\}, \quad (24)$$

where $\theta(\mathbf{x}_i)$ is the extracted feature value of $\mathbf{x}_i$ and bin_j is the jth bin after domain partition.

Let $\mathbf{c}_j$ be the constant output for bin_j and recall (9), we have the minified loss function as

$$\begin{aligned}\widehat{Loss}(\mathbf{F}(\mathbf{x}) + \mathbf{f}(\mathbf{x})) &\propto \sum_{i=1}^{m} w_i \exp(-\mathbf{v}_i \cdot \mathbf{f}(\mathbf{x}_i)) \\ &= \sum_j \sum_{(\mathbf{x}_i, \tilde{\mathbf{v}}_i) \in S_j} w_i \exp(-\mathbf{v}_i \cdot \mathbf{c}_j),\end{aligned} \quad (25)$$

where $\mathbf{F}(\mathbf{x})$ is the previously learned strong hypothesis, $\mathbf{f}(\mathbf{x})$ is the newly adopted weak hypothesis, and w_i is the weight of sample $\mathbf{x}_i$ w.r.t $\mathbf{F}(\mathbf{x})$. In particular, the loss received for bin_j is

$$loss_j(\mathbf{c}_j) \propto \sum_{(\mathbf{x}_i, \tilde{\mathbf{v}}_i) \in S_j} w_i \exp(-\mathbf{v}_i \cdot \mathbf{c}_j) \quad (26)$$

which is a convex function with regard to $\mathbf{c}_j$. Hence, the optimal constant outputs for each bin can be calculated with some proper optimization algorithm such as Newton-step method. Besides, the newly received loss given in (25) tells the fitness of the new weak hypothesis $\mathbf{f}(\mathbf{x})$ if added into the original strong hypothesis $\mathbf{F}(\mathbf{x})$, which guides the heuristic search method for the selection of discriminative features in the next section.

```
● Initialize:
    – Open list $OL = \{\theta_1, \theta_2, ..., \theta_n\}$, closed list $CL = \varnothing$
● Loop:
    – Select a promising feature $\theta^*$ from $OL$:
          $\theta^* = \arg\max_{\theta \in OL} (Fitness(\theta))$
    – $OL = OL - \{\theta^*\}$, $CL = CL \cup \{\theta^*\}$.
    – Expand $\theta^*$ to construct a new feature set $\Theta^*$
    – $OL = OL \cup (\Theta^* - CL)$
● Output:
    – The constructed feature set $OL \cup CL$
```

Fig. 15. Heuristic search method for sparse granular features.

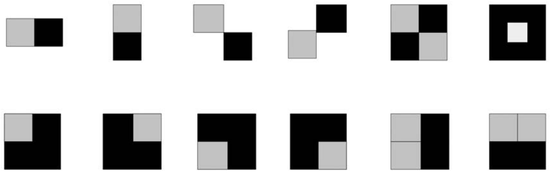


Fig. 16. Some Haar-like features in granular space for the initialization of open list in heuristic search method. Notice that each granule is a square rather than a rectangle, so the total number of enumerated Haar-like features in granular space is much less than that in integral image [7].

4.2.2 Heuristic Search for Sparse Granular Features

Although the loose constraint in (23) endows sparse granular features with great versatility, it also brings in a serious problem in practice: the astronomical figure of all possible features. To address this issue, a heuristic-search-based feature selection mechanism is developed to efficiently construct a compact and effective sparse granular feature set as follows.

The heuristic search is a classical optimization algorithm in artificial intelligence theory [28]. Two lists are maintained during its search process: A closed list for expanded elements to avoid duplicated expansions and an open list for unexpanded elements for further expansion. Roughly, it is formalized as shown in Fig. 15, in which elements in two lists are various sparse granular features defined in (23).

The fitness evaluation of a sparse granular feature, which reflects the heuristic knowledge integrated into the search process, essentially guides the expansion of elements in open set and thus dominates the whole search process. Reasonably, to improve the previously learned strong hypothesis in Vector Boosting algorithm, features that achieve low training losses with the optimal piece-wise functions are more favorable. Besides, sparseness of features should also be taken into consideration since it determines the computational complexity, and is closely related to the structural risk which affects the generalization ability. In addition, intuitively, a sparse granular feature with fewer granules is more likely to evolve into a better one by adding new granules. Therefore, the heuristic search method should prefer sparse granular features with smaller training losses and lower complexities.

Denote the *incremental logarithmic loss reduction* of $\mathbf{f}(\mathbf{x})$ w.r.t $\mathbf{F}(\mathbf{x})$ as

$$J(\mathbf{f}(\mathbf{x}), \mathbf{F}(\mathbf{x})) = -\log(Loss(\mathbf{F}(\mathbf{x}) + \mathbf{f}(\mathbf{x}))) + \log(Loss(\mathbf{F}(\mathbf{x}))) \tag{27}$$

and then the fitness function of sparse granular feature θ is empirically defined as:

$$Fitness(\theta) = J(\mathbf{f}(\mathbf{x}; \theta, \mu^*), \mathbf{F}(\mathbf{x})) - \beta\|\theta\|_1, \tag{28}$$

where μ^* is the optimal piece-wise function learned for θ (see Section 4.2.1), $\|\theta\|_1$ is the number of granules in θ, and β is a small penalty coefficient that is set as 0.001 in experiments. With this fitness function, a promising sparse feature θ^* can be selected from the open list. To expand it into a new feature set Θ^*, we employ three different operators, *add* (29), *delete* (30), and *replace* (31), which are *loading in a new granule*, *deleting an existed one* and *replacing an existed one with a new one*, respectively. These operators generate lots of features with small variations from the original sparse granular feature, and better features are likely to be found in those generated ones. Denote the granule set of θ^* as P, a granule in P and its corresponding coefficient as p_{in} and α_{in}, another granule that does not belong to P and its corresponding coefficient as p_{out} and α_{out}, and the neighboring granule set of p_{in} as $Nb(p_{in})$, *add*, *delete*, and *replace* operators can be formalized as follows:

$$\Theta_a = \{\theta_a | \theta_a = \theta^* + \alpha_{out} p_{out}\},\ p_{out} \notin P,\ \alpha_{out} = \{-1, +1\}, \tag{29}$$

$$\Theta_d = \{\theta_d | \theta_d = \theta^* - \alpha_{in} p_{in}\},\ p_{in} \in P, \tag{30}$$

$$\begin{aligned}\Theta_r = \{\theta_r | \theta_r = \theta^* - \alpha_{in} p_{in} + \alpha_{out} p_{out}\},\\ p_{in} \in P, p_{out} \in Nb(p_{in}), \alpha_{out} = \alpha_{in}.\end{aligned} \tag{31}$$

Due to the large number of generated features, it is infeasible to take all of them as the new feature set Θ^* inserted into the open list. Practically, an alternative way is to select only the best one of each set to make up this expanded feature set

$$\Theta^* = \left\{ \theta_i^* \middle| \theta_i^* = \arg\max_{\theta \in \Theta_i} (Fitness(\theta)),\ i = a, d, r \right\}. \tag{32}$$

In this way, in each round of the heuristic search three new promising features can be generated and after a certain number of rounds a discriminative feature set is obtained.

4.3 Brief Summary

For a weak hypothesis $\mathbf{f}(\mathbf{x}; \theta, \mu)$, Section 4.2.1 describes how to optimize a piece-wise function μ for a selected sparse granular feature θ, and Section 4.2.2 introduces the heuristic search method that is capable to construct a compact and effective feature set for the Vector Boosting algorithm. One thing that has not been clarified in the heuristic search (Fig. 15) is the initialization of the open list OL. Experimentally, we enumerate many Haar-like features in the granular space, and select a small part of them to constitute the open list (Fig. 16). The fitness function defined in (28) is employed again to filter out unpromising Haar-like features as the first-round selection. In summary, the entire process of weak learner in granular space for the Vector Boosting algorithm can be formalized as shown in Fig. 17.

```
0. (Given)
    $\{(\mathbf{x}_i, \tilde{\mathbf{v}}_i)\}$:    training data
    $\{w_i\}$:    sample weights
    $\Theta_{haar} = \{\theta_1, \ldots, \theta_m\}$:    Haar-like features in granular space
    $n$:    the size of initial open list
1. (Initialize)
    Set open list with promising Haar-like features
    $OL = \left\{ \theta_i \,\middle|\, \begin{array}{l} \forall \theta_i \in OL,\ \forall \theta_j \in \Theta_{haar} - OL \\ Fitness(\theta_i) \geq Fitness(\theta_j) \end{array} \right\}$, while $|OL| = n$
2. (Weak Learning Procedure)
    ● Call heuristic search procedure in Fig.15 to construct a sparse feature set $\Theta^*$
    ● For each sparse feature $\theta_i \in \Theta^*$, obtain its optimal piece-wise function $\mu_i$ to form a weak hypothesis $\mathbf{f}(\mathbf{x};\theta_i,\mu_i)$.
    ● Find the weak hypothesis with lowest loss
        $\mathbf{f}^*(\mathbf{x};\theta,\mu) = \arg\min\left(Loss\left(\mathbf{F}(\mathbf{x}) + \mathbf{f}(\mathbf{x};\theta_i,\mu_i)\right)\right)$
3. (Output)
    $\mathbf{f}^*(\mathbf{x};\theta,\mu)$, the best weak hypothesis for Vector Boosting
```

Fig. 17. Weak learner with sparse granular features and piece-wise functions for the Vector Boosting algorithm.

5 Experiments

5.1 WFS Tree Structure and Vector Boosting Algorithm

In order to demonstrate that the WFS tree structure is able to make moderate decisions for an input pattern in branching nodes with the nonexclusive-vector strong classifiers learned by the Vector Boosting algorithm as claimed in Section 2 and Section 3, a comparison is made among four approaches, including the WFS tree, the pyramid [9], the parallel cascades [13], and the AdaBoost.MH algorithm [14]. The task is to learn a strong hypothesis that computes a 3D determinative vector as the first layer of the face detector. All patterns are divided into four categories: three face classes (left profile, frontal, and right profile) and a nonface class. Every approach makes use of the same Haar-like feature set and the same domain-partition-based weak hypothesis (piece-wise function). To reject nonfaces as quickly as possible while preserving most of the faces, we reimplement the four approaches and fix the detection rate at 0.98 during the boosting procedure, illustrating their asymptotic false alarm rates on the testing set in Fig. 18.

According to this comparative experiment, among the four approaches, the WFS tree achieves the best performance since it requires the fewest weak classifiers to reach the same false alarm rate. In fact, such gain should be mainly attributed to the Vector Boosting algorithm. By means of assigning the three face views with orthogonal intrinsic projection vectors as shown in Table 3, which makes their objective regions overlapped and their targets compatible, the Vector Boosting algorithm manages to exploit similarities between faces of different views for the sake of fast background rejection, meanwhile keeping back their diversities for further identification. Contrastively, the pyramid approach treats all the three face views as one ensemble positive class, and therefore employs the classical 2-class AdaBoost algorithm to learn a binary strong classifier, which suffers from the diversities between different views. Differently, the parallel-cascade approach learns three binary classifiers to distinguish three face views from nonfaces individually, which ignores the valuable similarities between different face views. As for the AdaBoost.MH algorithm, since it regards three face classes and the nonface class as four mutually distinct ones, it puts considerable emphasis on the differentiation of the three face views. However, separating face views is indeed unnecessary here as the main task in the first layer is to reject nonfaces as quickly as possible, which inevitably slowdowns the convergence speed of false alarm rate.

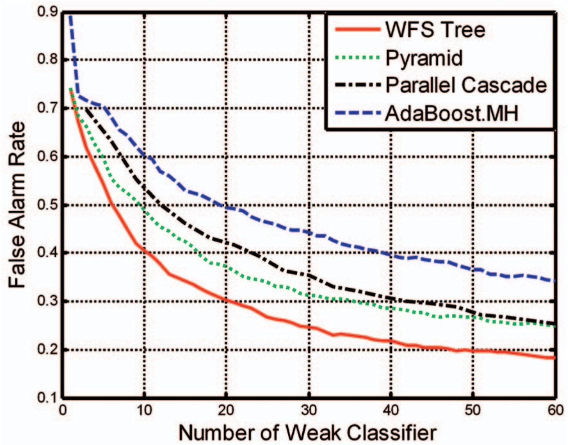


Fig. 18. Comparison of false alarm rates on testing set. The detection rate of testing face samples is adjusted to 0.98 for each approach.

5.2 Sparse Granular Features

Practically, two more constraints on sparse granular features are employed during the heuristic search procedure: one is requiring the sum of combining coefficients in (23) to be 0, and the other is restricting the number of granules in a feature up to a maximum. The first constraint makes the features "balanced" so that no zero-mean normalization is needed, and the second one controls the scale and complexity of the whole sparse granular feature set. Essentially, varying the maximum number of granules trades off between the structural risk, empirical risk and computational efficiency of resulted sparse granular features. So, finding a proper maximum granule number is an important issue in heuristic search process. In this experiment, five strong classifiers are learned with classical AdaBoost algorithm to distinguish faces of a certain view (e.g., frontal faces) from backgrounds. Each adopts a different type of feature to construct the weak hypothesis: sparse features of maximum granule number 4, 6, 8, and 10, as well as Haar-like features applied in [7]. Similar with the previous experiment, we set the detection rate at 0.98 and compare the asymptotical testing false alarm rates of each type of features on different classification problems. The results are illustrated in Fig. 19.

Compared to Haar-like features applied in [7], our sparse granular features achieve higher classification accuracy. Due to the flexible combination rules of sparse granular features, even the simplest form of the maximum granule number 4 outperforms the rigid Haar-like features. However, increasing the maximum granule number to enlarge the possible feature set cannot always improve the learned weak classifiers (compare maximum granule numbers 8 and 10).

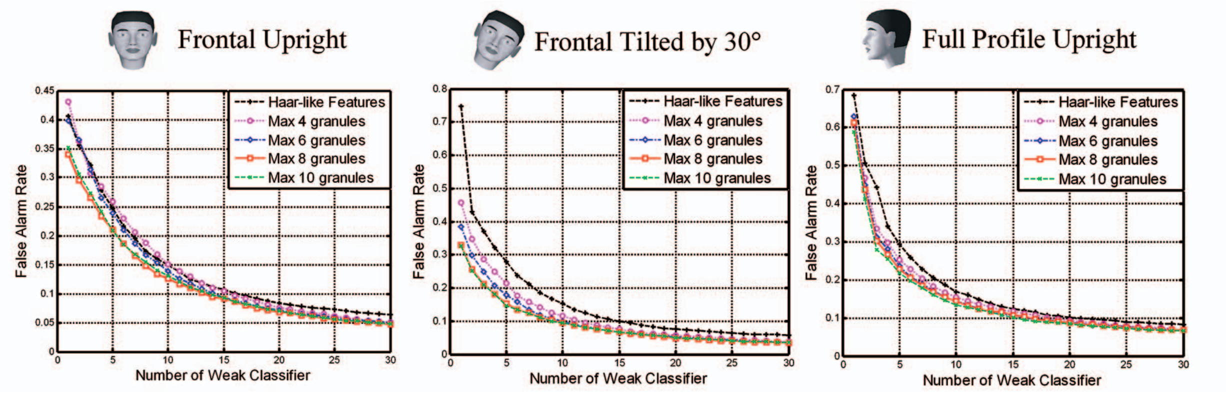


Fig. 19. Asymptotic testing false alarm rates of different types of features with the detection rate at 0.98. From left to right, the binary classification problems are frontal upright faces versus backgrounds, frontal tilted by 30° faces versus backgrounds, and full profile upright faces versus backgrounds. They are three typical views of all 15 ones defined in Fig. 1, corresponding to original form, RIP, and ROP, respectively.

On one hand, as a result of rising maximum granule number, the combination explosion of all possible features makes the heuristic search algorithm become inadequate to find optimal ones. On the other hand, more complex sparse granular features indicate higher structural risks and possibly lower generalization ability in classification. Taking the computational efficiency (Fig. 20) into consideration, in practice, we choose 8 as the maximum granule number in learning of our multiview face detector described in the following section.

5.3 Multiview Face Detection

For the training of MVFD, we collect and manually label 30,000 frontal faces, 25,000 half-profile ones, and 20,000 full-profile ones, which are taken in various conditions (different poses, illuminations, and expressions). Many faces involve up/down rotation up to 30° so that the learned detector can tolerate enough Pitch variance of faces to cater to surveillance applications. These faces are rotated, flipped, and resized into 24×24 patches to obtain training samples of all 15 views. With these samples, we train a quartered upright multiview face detector with the WFS tree structure in Fig. 5, and the training procedure is formalized in Fig. 21. The first four sparse grnular features in the root node learned by the Vector Boosting algorithm and the heuristic search method are shown in Fig. 22. To achieve the required extremely low false alarm rate F, in fact, each leaf node in Fig. 5 is extended to a cascade. Hence, the exact number of nodes is larger than that shown in Fig. 5. Experimentally, there are 234 nodes in 18 layers as a whole that constituting this multiview face detector.

With this powerful WFS tree structured detector, an exhaustive search procedure is applied to detect faces in an image. After that, positive responses are clustered according to their positions, sizes, and poses. With some simple criterions such as nonoverlapping, these clusters are merged as the final results of face detection as shown in Fig. 24. It is assured that every successful detection properly describes the corresponding face: the size is not too small or too large, the difference of RIP angle is no more than 15 degrees, and the difference of ROP angle is no more than one view. To compare with the previous works which gave

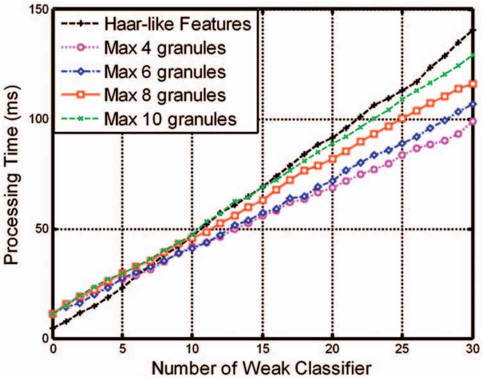


Fig. 20. Computational efficiency of different types of features. The processing time is the average time spent on each frame of a QVGA (320×240) video sequence. Each frame is scaled from its origin down to 40×30 with scale ratio 1.26 (cubic root of 2) for sparse granular features. As for the Haar-like features, their templates are scaled with the same ratio instead of scaling the image.

```
0. (Given)
    f:    false positive rate of each node
    F:    expected overall false positive
    d:    detection rate of each node
    P:    positive training sample set (15 classes)
    N:    negative training sample set
1. (Initialize)
    Let current node $E$=$E_{root}$, the root node of WFS tree
2. (Node Learning Procedure)
    ● (Bootstrap) From $P$ and $N$, collect samples that have passed all $E$'s parent nodes, constituting two sub training set $p$ and $n$ of proper size.
    ● (Learning) With $p$ and $n$, call Vector Boosting algorithm to train a strong classifier $\mathbf{G}^{(E)}(\mathbf{x})$ until it satisfies false positive rate $f$ and detection rate $d$.
    ● (Evaluating) Evaluate the overall false positive rate $F^{(E)}$ through the path from the root node.
        ■ If $F^{(E)} > F$
            ✧ For each child node of $E$, $E_{child}$, let $E = E_{child}$ and recursively call this training procedure for node $E$.
        ■ Else
            ✧ Return current node $E$
3. (Output)
    The root node of WFS tree $E_{root}$
```

Fig. 21. Training algorithm of WFS tree structured detector.

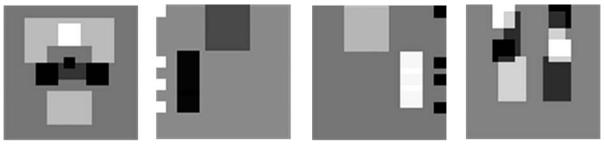


Fig. 22. The first four sparse granular features in the root node learned by the Vector Boosting algorithm and heuristic search method. They are illustrated as linear features: white blocks indicate positive granules while black ones denote negative granules. As each granule is, in fact, the average of pixels in the corresponding patch, granules of different sizes are drawn in different gray levels.

full testing results (e.g., Schneiderman and Kanade's Bayesian-rule approach [6], Wu et al.'s parallel cascade structure [13], and Huang et al.'s WFS tree + Haar-like feature [11]), we test our multiview face detector on the CMU profile testing set. Some other related works such as Jones and Viola's decision tree method [10] and Li et al.'s pyramid method [9] are not under consideration due to lack of their full testing results on this standard testing set. To trade off between detection rate and false alarm to give a whole ROC curve, we adopt a control parameter that correspondingly adjusts the threshold of strong classifier in each node. The testing results are drawn in Fig. 23a as ROC curves, in which our sparse feature + WFS tree-based MVFD system achieves the best performance. Some detection results are shown in Fig. 24b.

5.4 Rotation Invariant Multiview Face Detection

Since the upright detector covers the range of RIP from $-45°$ to $+45°$ and ROP from $-90°$ to $+90°$, to deal with the face space, three more detectors are generated by means of rotating the upright one by 90°, 180°, and 270°, respectively (Fig. 3). The four detectors, working in parallel as a whole, constitute a rotation invariant multiview face detector. Since so far there is no standard testing set for this problem, we only show some detection results in Fig. 24a.

In the root node of WFS tree (Fig. 5), disabling the first and the third branches (i.e., left profile and right profile faces) will lead to a rotation invariant frontal face detector. To compare with Rowley's ANN method [5] and Jones's decision tree method [10], we tested this detector on the CMU rotate testing set. The results are shown in Fig. 23b, in which our sparse feature + WFS tree approach again outperforms other works. Some detection results are shown in Figs. 24c and 24d.

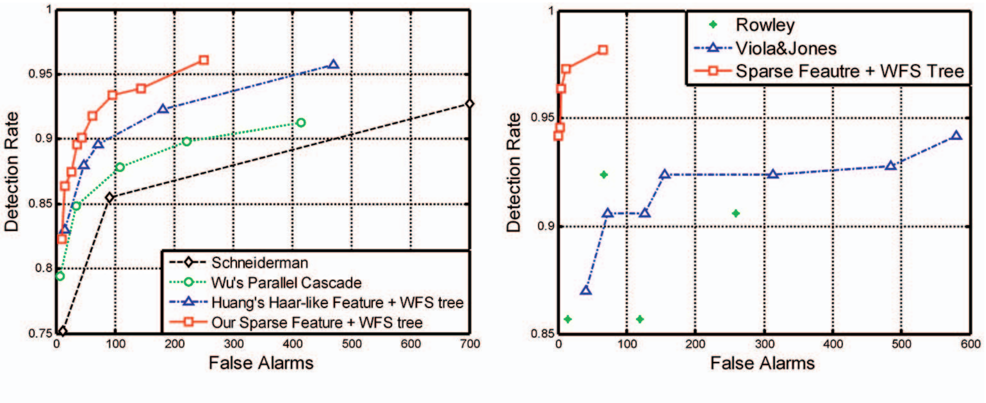
5.5 Discussion on Time Complexity of the Algorithm

Similar with previous related works, our face detection framework includes two parts: offline learning and online detecting. Understandably, the offline learning process is time-consuming. In particular, searching for appropriate sparse granular features in the vast feature space is still time-consuming although the heuristic search method is adopted. Fortunately, we can use the incremental feature selection technique [27] to significantly improve the efficiency of feature selection procedure. Approximately, training a single cascade detector for frontal faces requires only two days on a P4-3.0GHz PC. As for the WFS tree-based multiview face detector, since classifiers after branching nodes could be trained in parallel, the entire training procedure of the WFS tree spends about two weeks if three PCs are employed.

When detecting faces, instead of scaling templates (as what is done with Haar-like features), we scale the image itself to seek faces of different sizes. Experimentally, as the preprocessing stage before applying sparse granular features, it costs only 10 ms to scale an image of QVGA size (320×240) down to the size 40×30 with step scale ratio 1.26 (cubic root of 2), including the computation of integral image in each scale. Compared with the scale template approach that spends $4 \sim 5$ ms in preprocessing, the scale image approach is a little slower but can extract more accurate sparse granular features. In fact, as shown in Fig. 20, although the sparse granular features require more preprocessing time (shown as the processing time at zero weak classifier), the running time per sparse granular feature is less than that per Haar-like feature. Therefore, as the number of weak classifiers increases, the sparse granular features quickly exceed the Haar-like features in computational efficiency. According to the experiments, on common video sequences of QVGA size (320×240), our MVFD achieves a speed of about 10 fps and the rotation invariant MVFD runs at a speed of about $3 \sim 4$ fps.

6 Conclusion

In this paper, to address the problem of detecting rotation invariant multiview faces with high speed and accuracy, we develop the WFS tree structure for the construction of face detector, which partitions the complicated detection task into individual-view-based ones in terms of coarse-to-fine



(a) (b)

Fig. 23. ROC curves for comparison on standard testing sets. (a) is the ROC curves on CMU profile testing set [6] (441 multiview faces in 208 images). (b) is the ROC curves on CMU rotate testing set [5] (223 frontal faces with different RIP angles in 50 images).

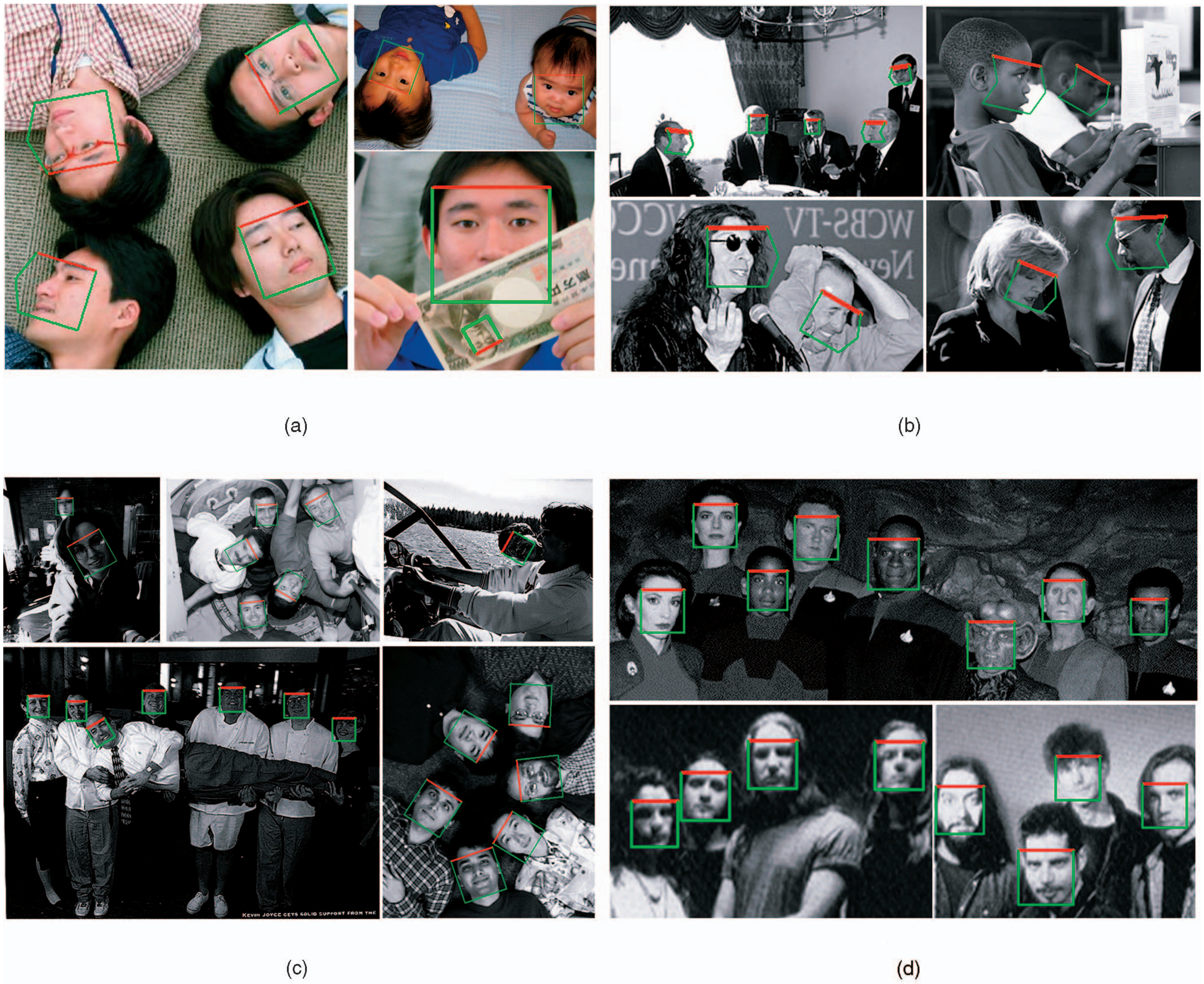


Fig. 24. Detection results. (a) Rotation invariant multiview face detection. (b) Result on CMU profile testing set. (c) Result on CMU rotate testing set. (d) Result on CMU frontal testing set.

strategy and organizes them moderately compared with the decision tree approach [10] and the pyramid approach [9]. The experiments on several standard testing sets have shown that our approach achieves significant improvements in both speed and accuracy over previous published methods for face detection.

Two innovative methods are proposed for the learning of each node of the tree: the Vector Boosting algorithm and the sparse granular feature concept. The Vector Boosting algorithm is a general framework to handle various multiclass problems with the additive regression model in [15]. Intrinsic projection vectors are employed to define the objective regions of each category and thus determine the exponential loss function. It is an extension for classical AdaBoost algorithm from which both AdaBoost.MH and AdaBoost.MR can be derived. The flexibility originating from intrinsic projection vectors enables the Vector Boosting algorithm to deal with many complicated classification, even regression problems. Sparse granular features are defined in the granular space of an image. They are more capable to match complex and irregular patterns than original Haar-like features in [7] while maintaining similar computational load. To seek enough discriminative sparse granular features in such a tremendous space for a weak learner, a heuristic search method is proposed for the construction of a compact and effective feature set. Besides, with the adaptive mapping of piece-wise functions, the weak learner succeeds in training very efficient weak hypotheses, which strongly supports the Vector Boosting algorithm on the learning of strong classifiers. Based on the flexibility of the Vector Boosting algorithm and the scalability of sparse granular features observed in the challenging face detection task, we argue that both of them are of high potential in other fields of pattern recognition and computer vision.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments on the manuscript. This work is supported in part by the National Science Foundation of China under grants No. 60332010, No. 60673107, the National Basic Research Program of China under Grant No. 2006CB303100, and it is also supported by a grant from Omron Corporation.

References

[1] M.-H. Yang, D.J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 1, Jan. 2002.

[2] T. Kanade, "Picture Processing by Computer Complex and Recognition of Human Faces," PhD thesis, Kyoto Univ., 1973.

[3] C. Kotropoulos and I. Pitas, "Rule-Based Face Detection in Frontal Views," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing,* vol. 4, pp. 2537-2540, 1997.

[4] E. Osuna, R. Freund, and F. Girosi, "Training Support Vector Machines: An Application to Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 130-136, 1997.

[5] H. Rowley, S. Baluja, and T. Kanade, "Rotation Invariant Neural Network-Based Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 38-44, 1998.

[6] H. Schneiderman and T. Kanade, "A Statistical Method for 3D Object Detection Applied to Faces and Cars," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 746-751, 2000.

[7] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2001.

[8] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," *Proc. Fourth Int'l Conf. Machine Learning,* 1997.

[9] S.Z. Li et al., "Statistical Learning of Multi-View Face Detection," *Proc. Seventh European Conf. Computer Vision,* pp. 67-81, 2002.

[10] M. Jones and P. Viola, "Fast Multi-View Face Detection," MERL-TR2003-96, July 2003.

[11] C. Huang, H.Z. Ai, Y. Li, and S.H. Lao, "Vector Boosting for Rotation Invariant Multi-View Face Detection," *Proc. 10th IEEE Int'l Conf. Computer Vision,* 2005.

[12] R. Xiao, L. Zhu, and H. Zhang, "Boosting Chain Learning for Object Detection," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* 2003.

[13] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast Rotation Invariant Multi-View Face Detection Based on Real AdaBoost," *Proc. Sixth Int'l Conf. Automatic Face and Gesture Recognition,* pp. 79-84, 2004.

[14] R.E. Schapire and Y. Singer, "Improved Boosting Algorithms Using Confidence-Rated Predictions," *Machine Learning,* vol. 37, pp. 297-336, 1999.

[15] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *Annals of Statistics,* vol. 28, pp. 337-374, 2000.

[16] C. Liu and H.Y. Shum, "Kullback-Leibler Boosting," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 587-594, 2003.

[17] T. Mita, T. Kaneko, and O. Hori, "Joint Haar-Like Features for Face Detection," *Proc. 10th IEEE Int'l Conf. Computer Vision,* 2005.

[18] R. Lienhart and J. Maydt, "An Extended Set of Haar-Like Features for Rapid Object Detection," *Proc. IEEE Int'l Conf. Image Processing,* 2002.

[19] S. Baluja, M. Sahami, and H.A. Rowley, "Efficient Face Orientation Discrimination," *Proc. IEEE Int'l Conf. Image Processing,* 2004.

[20] P. Wang and Q. Ji, "Learning Discriminant Features for Multi-View Face and Eye Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[21] Y. Abramson and B. Steux, "YEF* Real-Time Object Detection," *Proc. Int'l Workshop Automatic Learning and Real-Time,* 2005.

[22] M. Osadchy, M.L. Miller, and Y. LeCun, "Synergistic Face Detection and Pose Estimation with Energy Based Models," *Proc. Neural Information Processing Systems (NIPS),* 2004.

[23] F. Fleuret and D. Geman, "Coarse-to-Fine Face Detection," *Int'l J. Computer Vision,* vol. 41, nos. 1/2, pp. 85-107, 2001.

[24] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

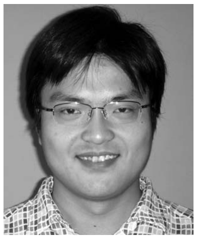
[25] Y. Freund, "Boosting a Weak Learning Algorithm by Majority Algorithm," *Machine Leaning,* vol. 43, no. 3, pp. 293-318, 2001.

[26] N. Duffy and D.P. Helmbold, "Boosting Methods for Regression," *Machine Leaning,* vol. 47, nos. 2-3, pp. 153-200, 2002.

[27] C. Huang, H. Ai, Y. Li, and S. Lao, "Learning Sparse Features in Granular Space for Multi-View Face Detection," *Proc. Seventh Int'l Conf. Automatic Face and Gesture Recognition,* pp. 401-406, 2006.

[28] N.J. Nilsson, *Artificial Intelligence, A New Synthesis,* Morgan Kaufmann, pp. 140-161, 1998.

[29] Y. Li, S. Gong, J. Sherrah, and H. Liddell, "Support Vector Machine Based Multi-View Face Detection and Recognition," *Image and Vision Computing,* vol. 22, pp. 413-427, 2004.



Chang Huang received the BS degree in computer science and technology from Tsinghua University, China, in 2003. He is currently a PhD candidate at Tsinghua University. He has won Excellent Student Scholarships at Tsinghua University in 2001, 2002, 2004, and 1999. His research interests include computer vision, machine learning, and pattern recognition, with special attention to face detection. He has published 13 papers in peer-reviewed domestic journals and international conferences. He is a student member of the IEEE.

Haizhou Ai received the BS, MS, and PhD degrees all from Tsinghua University, China in 1985, 1988, and 1991, respectively. He spent the period 1994-1996 in the Flexible Production System Laboratory at the University of Brussels, Belgium, as a postdoctoral researcher. He is currently a professor in the Computer Science and Technology Department at Tsinghua University. His current research interests are facial image processing, biometrics, and visual surveillance. He has published more than 50 papers in peer-reviewed domestic journals and international conferences. He is a member of the IEEE and the IEEE Computer Society.

Yuan Li received the BS degree in computer science and technology with the honor of Outstanding Graduating Student from Tsinghua University, China, in 2005. She has also won Excellent Student Scholarships at Tsinghua University in 2001, 2002, and 2004, and IBM Scholarship for Outstanding Students in China, in 2003. She is currently pursuing a MS degree at Tsinghua University. Her research interests are computer vision and machine learning, with a current specific focus on face detection and head tracking.

Shihong Lao received the BS degree in electrical engineering from Zhejiang University in 1984 and the MS degree in electrical engineering from Kyoto University, Japan, in 1988. He has worked in the Sensing and Control Technology Laboratory at Omron Corporation since 1992 and currently is the chief technologist of the facial image processing project. He is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.