Topology Modeling for Adaboost-Cascade Based Object Detection

Zhaofeng He, Tieniu Tan[*] and Zhenan Sun

Center for Biometrics and Security Research
National Laboratory of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences. P.O. Box 2728, Beijing, P.R.China, 100190
{zfhe, tnt, znsun}@nlpr.ia.ac.cn

Abstract

Several important issues involved in Adaboost-cascade learning still remain open problems. In this work, several novel ideas are proposed for improved Adaboost-cascade object detection. The most important one is the novel Topology Oriented Adaboost (TOBoost) algorithm. TOBoost immediately minimizes the classification error of each selected feature, and thus enables the final detector to be more discriminative and to converge more quickly. Moreover, a simple cascading scheme is presented for tuning the cascade parameters of TOBoost; and Gaussian kernel density estimation is introduced to enhance the generalization ability of TOBoost. Another important contribution is the topology modeling of Haar-like (HL) features, which reveals an interesting property of negative HL features and significantly avoids unnecessary training computations. Non-adjacent Haar-like features are consequently configured for more effective object representation. The above enhancements result in a more efficient and stable detector with fewer features. Extensive experiments in the application of iris detection are conducted and encouraging performance is achieved.

Key words: Object Detection, Haar-like Features, Adaboost-cascade Learning, AUC Optimization, Iris Detection, Iris Biometrics.

1. Introduction

Object detection is a challenging problem in computer vision due to the variations of appearances caused by illuminations, poses, occlusions, etc., and still remains a hot research topic [2, 14, 20, 21]. Recently, Adaboost-Cascade (AC) learning has been regarded as a break-through in object detection [2, 3, 11, 12, 20, 21]. As illustrated in Fig. 1, Adaboost-cascade learning incorporates three advantages for robust real-time object detection [21]: 1) a specific feature set (e.g., Haar-like features) for object representation; 2) efficient Adaboost learning for feature selection; and 3) the cascade structure for fast training and execution.

Although having been exhaustively explored in the literature [4, 6, 15, 18, 19], several important issues in Adaboost cascade learning are still not adequately addressed and remain open problems. The first problem is related to the feature set. Existing features (e.g. Haar-like features [13, 20]) commonly concentrate on adjacent image structures. While adjacent structures are powerful in capturing local details, we argue that non-adjacent structures should be more efficient and stable in object detection (especially under the mechanism of Adaboost learning). Another important problem associated with the feature set is the double-sliding of negative features. In Adaboost-cascade learning, negative samples are cropped by sliding a sub-window over a negative image, while the candidate features are obtained by sliding some kind of features over different positions of each sample. Such double-sliding mechanism results in severe duplicated computations and hence an unacceptably time-consuming training procedure. By carefully modeling the candidate features, we show that such duplications can be avoided, which greatly accelerates the training procedure.

[*]Corresponding Author: Tel: +86 10 82614515; Fax: +86 10 62551993.

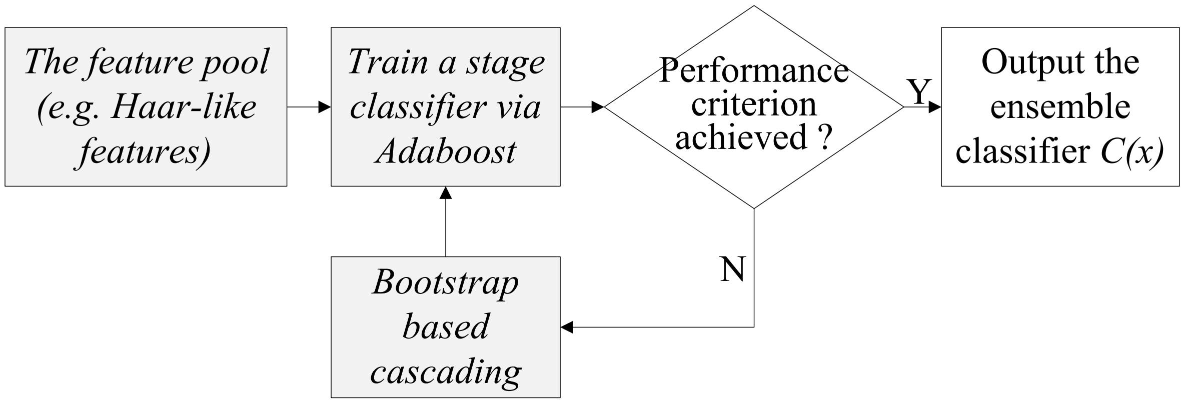

The feature pool (e.g. Haar-like features)
Train a stage classifier via Adaboost
Performance criterion achieved ?
Y
Output the ensemble classifier C(x)
N
Bootstrap based cascading

Figure 1: A generalized flowchart of Adaboost-cascade learning, which incorporates three advantages: features, Adaboost learning and cascading.

A second problem is related to the Adaboost learning. In conventional Adaboost algorithms, two histograms are used to approximate the probability distributions of the positive and negative samples. A weak learner is then developed based on some statistical measures (e.g. Kullback-Leibler divergence [14]) of the two distributions, and a look up table (LUT, or equally decision tree [19]) based component classifier is built. However, these empirical statistics remain intermediate objectives to classification error [15]. That is, the feature that minimizes these statistics is not necessarily the one that minimizes the classification error, thus resulting in a suboptimal detector. Moreover, the histogram-based scheme also leaves another inevitable problem: in order to approach the genuine probability distributions, a large number of training samples must be maintained, which requires heavy memory and computation demands, or else the resultant classifier can be over-fitting and sensitive to noise.

A third problem is related to the cascade structure. Due to the explosively large computation and memory demands of Adaboost learning, cascade learning (via bootstrap) has become a necessity (so that only a fraction of possible negative samples are maintained at each stage). Unfortunately, this conflicts with LUT classifier which prefers a large number of training samples. In other words, cascade learning can be harmful to the generalization ability of Adaboost detectors. Moreover, optimal tuning of the cascade parameters is not trivial and still remains an open problem.

In this paper, we present several novel ideas for Adaboost learning to address each of these problems. Firstly, non-adjacent Haar-like features are adopted to represent the structure of the object. Secondly, we model the Haar-like features by their topological properties. This topology model reveals an interesting intrinsic property of Haar-like features, which not only avoids duplicated computations on negative samples but also enhances the robustness of Adaboost learning. Thirdly, a novel topology oriented Adaboost algorithm is proposed. In contrast with its predecessors, TOBoost's weak learner directly minimizes the classification error, and therefore enables the selected feature to be more discriminative. Fourthly, we present a simple scheme to tune the cascade parameters for Adaboost-cascade learning. Finally, Gaussian kernel density estimation is introduced into Adaboost, which greatly enhances its generalization ability. The above enhancements result in a more efficient and stable detector with fewer features.

The rest of the paper is organized as follows. In Section 2, we describe how to avoid duplicated computations during Adaboost learning by simply modeling the Haar-like features by their topological properties. In Section 3, the novel TOBoost algorithm and the Gaussian kernel density estimation scheme are introduced. In Section 4, extensive experiments and discussions are presented for iris detection. In Section 5, we draw some conclusions.

2. Topology Modeling of Haar-like Features

A central objective of object detection, compared with object identification, is to tolerate individual variations but reveal the intrinsic properties (e.g. the structure) that all or most objects should have. In this work, Haar-like features are adopted for object representation. The justifications are as follows: 1) Haar-like (HL) features are computationally effective, especially when armed with integral image [20]. 2) Haar-like features show tolerance to appearance variations (partly due to the averaging over the rectangular region), and are efficient for representing image structures for object detection.

Despite of the fast computation of Haar-like features, existing Adaboost-cascade learning algorithms suffer greatly from heavy computation due to the huge number of candidate features and negative samples. In this section, we show how the computation can be dramatically reduced via topology modeling of Haar-like features. This begins with a generalized representation of them.

2.1. Generalized Haar-like Features

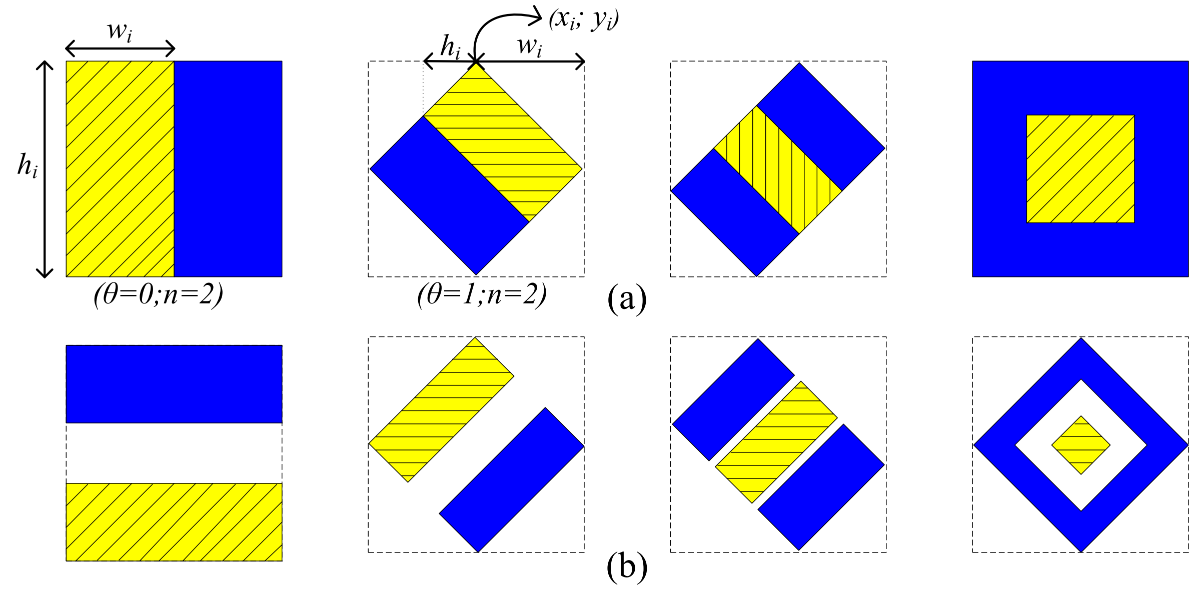


Figure 2: Generalization of Haar-like features. (a): adjacent Haar-like features and (b): non-adjacent Haar-like features.

As illustrated in Fig. 2 (a), Haar-like features are defined as the intensity difference between the diagonal yellow block and the blue block. Considering that each individual Haar-like feature can be decomposed into several (upright or tilted) rectangles, we propose to use the following representation to denote Haar-like feature h_t :

$$h_t = \left\{\theta, n, \left[x^i, y^i, h^i, w^i, \pi^i\right]_{i=1}^{n}\right\} \tag{1}$$

where $\theta \in \{0, 1\}$ is the type of each rectangle (with 0 for upright and 1 for tilted), $n \in \{2, 3\}$ denotes the number of rectangles in h_t, $\left(x^i, y^i\right)$ denotes the coordinates of the upper-left corner of the i-th rectangle (with respect to the upper-left corner[1] of the minimal rectangle that encompasses h_t, see Fig. 2), $\left(h^i, w^i\right)$ is the height and width of i-th rectangle, and π^i is the weight of the i-th rectangle such that $\sum_i \pi^i Area^i = 0$ (where $Area^i$ is the area (i.e., number of pixels) of the i-th rectangle.). The Haar-like feature value of h_t is calculated as:

$$\phi(h_t) = \sum_i \pi^i RectSum^i \tag{2}$$

where $RectSum^i$ denotes the pixel sum of the i-th rectangle.

Equation 1 indicates that different Haar-like features can be obtained by simply tuning these topology parameters (and thus enables flexible configurations for representing image structures of different complexities). In particular, Eq 1 directly enables the extension of the traditional adjacent Haar-like features (Fig. 2(a)) to non-adjacent ones (Fig. 2(b)). As pointed out in Section 1, non-adjacent structures should be more efficient and stable in object detection.

More importantly, Eq. 1 paves the way for the following topology modeling of Haar-like features.

[1]For titled Haar-like features, the upper-left corner usually refers to the top corner of each titled rectangle [13].

2.2. *Topology Modeling of Haar-like Features*

In Adaboost learning, the candidate features are obtained by sliding the Haar-like feature h_t over different locations (x_s, y_s) of sample s, denoted as follows:

$$\phi_{x_s,y_s,h_t} = \{x_s, y_s, h_t\} \tag{3}$$

where (x_s, y_s) is the coordinates of the upper-left corner of h_t on s. For example, features ϕ_{x_s,y_s,h_t} and $\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ in negative sample $I_{x,y}$ of Fig. 3 are considered as different features even though they are derived from the same Haar-like feature h_t (i.e. with the same topology). From this point of view, h_t is considered as parent-HL feature while ϕ_{x_s,y_s,h_t} child-HL feature. Clearly, the number of child-HL features ϕ_{x_s,y_s,h_t} that can be derived from parent-HL feature h_t is:

$$M = (WinH - HaarH)(WinW - HaarW) \tag{4}$$

where $WinH \times WinW$ is the size of the samples and $HaarH \times HaarW$ is the size of the minimal rectangle that encompasses h_t, see Fig. 3.

To go through the feature calculation of Adaboost learning, we continue to consider the extraction of negative samples. In Adaboost learning, the negative samples are cropped from a set of negative images at different scales[2] and locations. Here, we denote each cropped negative sample with $s = I_{x,y}$, where (x, y) is the coordinates of the upper-left corner of the cropped negative sample on I, see Fig. 3. Suppose the negative images are of size $ImH \times ImW$ and the negative samples are of size $WinH \times WinW$, then at most we can crop N_s negative samples from each negative image in total:

$$N_s = (ImH - WinH)(ImW - WinW) \tag{5}$$

With the above definitions, it is straightforward that the number of feature values that should be extracted from one negative image I associated with one parent-HL feature h_t is then:

$$N_{I,\phi_{h_t}} = N_s \times M \tag{6}$$

where N_s is the number of candidate samples that can be cropped from I, and M is the number of child-HL features ϕ_{h_t} derived from the parent-HL feature h_t. Equation 6 means we will have to calculate $N_{I,\phi_{h_t}}$ feature values associated with one parent-HL feature h_t on each negative image I.

Equation 6 contains many duplicated computations. To get some light on this, we consider the above procedure from another point of view. If we directly convolve the parent-HL feature h_t with the negative image I, the total number of valid values is then:

$$N_{I,h_t} = (ImH - HaarH)(ImW - HaarW) \tag{7}$$

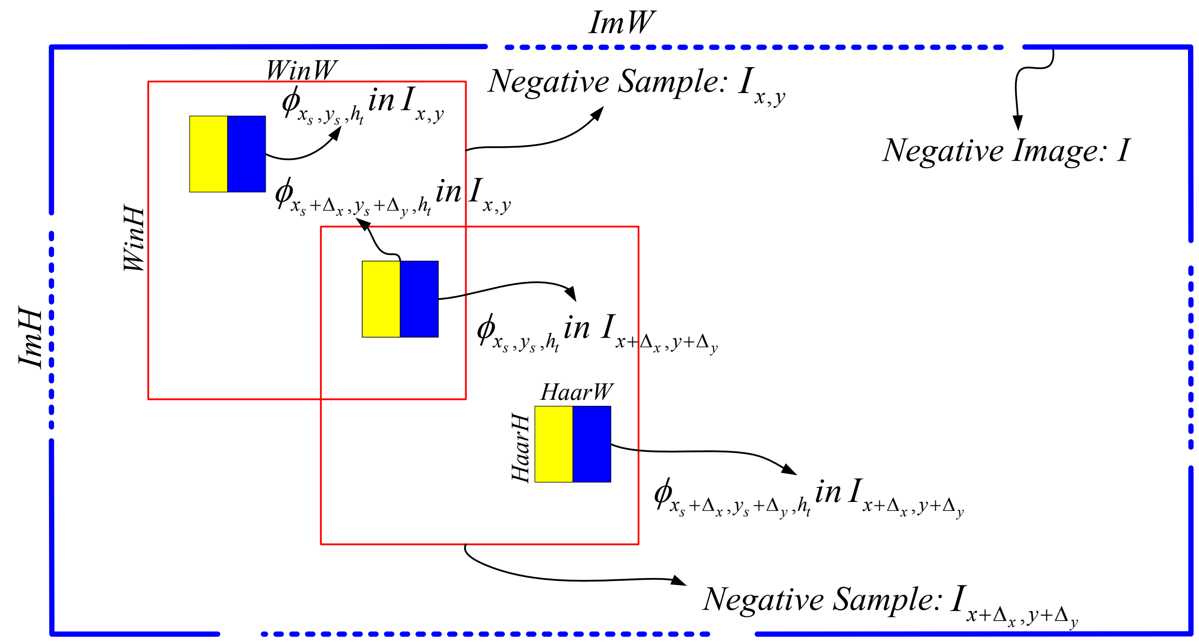
where $HaarH \times HaarW$ is the size of h_t. Equation 7 means we can at most calculate N_{I,h_t} valid feature values associated with the parent-HL feature h_t on image I. Clearly, $N_{I,h_t} \simeq N_s$ because $ImH \gg WinH \simeq HaarH$ and $ImW \gg WinW \simeq HaarW$. For example, $M = 256$, $N_s = 1,000,000$, $N_{I,h_t} = 1032256$ when $ImW = ImH = 1024$, $WinH = WinW = 24$ and $HaarH = HaarW = 8$. That is surprising because Eq. 6 shows that $N_{I,\phi_{h_t}} = N_s \times M \simeq N_{I,h_t} \times M$ feature values have to be calculated, but Eq. 7 shows that we can at most get N_{I,h_t} valid feature values associated with h_t and I. Clearly, this indicates severe duplicated computations during feature calculation on negative images.

Such duplications can also be clearly demonstrated as follows. Informally, Haar-like feature calculation can be denoted as:

$$\begin{aligned} I_{x+\Delta_x,y+\Delta_y} \cdot \phi_{x_s,y_s,h_t} &= (I_{x,y} * \delta_{\Delta_x,\Delta_y}) \cdot \phi_{x_s,y_s,h_t} \\ &= I_{x,y} \cdot (\phi_{x_s,y_s,h_t} * \delta_{\Delta_x,\Delta_y}) = I_{x,y} \cdot \phi_{x_s+\Delta_x,y_s+\Delta_y,h_t} \end{aligned} \tag{8}$$

where $\cdot$ is the pointwise multipliers, $*$ is the convolution operator, δ is the Dirac's delta function, and $I_{x,y} * \delta_{\Delta_x,\Delta_y}$ means sliding the subwindow with (Δ_x, Δ_y) displacements. From Eq. 8, we can see that $I_{x+\Delta_x,y+\Delta_y} \cdot \phi_{x_s,y_s,h_t} = I_{x,y} \cdot \phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$, which is schematically illustrated in Fig. 3, where $\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ in sample $I_{x,y}$ overlaps with feature ϕ_{x_s,y_s,h_t} in sample

[2]Here, we fix at original scale without loss of generalization.



ImW
$WinW$
ϕ_{x_s,y_s,h_t} in $I_{x,y}$
Negative Sample: $I_{x,y}$
Negative Image: I
$WinH$
$\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ in $I_{x,y}$
ImH
ϕ_{x_s,y_s,h_t} in $I_{x+\Delta_x,y+\Delta_y}$
$HaarW$
$HaarH$
$\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ in $I_{x+\Delta_x,y+\Delta_y}$
Negative Sample: $I_{x+\Delta_x,y+\Delta_y}$

Figure 3: Illustration of duplicated computations. Due to the "double sliding" on negative images, child-HL feature $\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ in sample $I_{x,y}$ overlaps with child-HL feature ϕ_{x_s,y_s,h_t} in sample $I_{x+\Delta_x,y+\Delta_y}$ ϕ_{x_s,y_s,h_t}, resulting in duplicated computations.

$I_{x+\Delta_x,y+\Delta_y}$. That is, different child-HL features on different samples 'happen'to overlap each other and thus produces unnecessary duplicated computations.

Furthermore, taking insights into the calculation of ϕ_{x_s,y_s,h_t} on negative image I, we can see that the entire feature values of ϕ_{x_s,y_s,h_t} on I are calculated (on exhaustively randomly cropped negative samples $I_{x,y}$) as follows:

$$\begin{aligned} F_{I,\phi_{x_s,y_s,h_t}} &= \left\{ I_{x,y} \cdot \phi_{x_s,y_s,h_t} \right\}_{x,y} \\ = I * \phi_{x_s,y_s,h_t} &= (I * h_t) * \delta_{x_s,y_s} \end{aligned} \tag{9}$$

Similarly, the entire feature values of $\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ are calculated as:

$$\begin{aligned} F_{I,\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}} &= \left\{ I_{x,y} \cdot \phi_{x_s+\Delta_x,y_s+\Delta_y,h_t} \right\}_{x,y} \\ =I * \phi_{x_s+\Delta_x,y_s+\Delta_y,h_t} &= (I * h_t) * \delta_{x_s+\Delta_x,y_s+\Delta_y} \end{aligned} \tag{10}$$

Considering the double-sliding of child-HL features and negative samples, and $ImH \gg \Delta_x, ImW \gg \Delta_y$, we can approximately say that $F_{I,\phi_{x_s,y_s,h_t}} \simeq F_{I,\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}}$ except a few samples on the margins of I. This is quite surprisingly because it means that the feature values of ϕ_{x_s,y_s,h_t} is almost the same as $\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ (and thus duplicated computations again). Hence, rather than calculating $F_{I,\phi_{x_s,y_s,h_t}}$ exhaustively on different (x_s, y_s), we can instead calculate $I * h_t$ and then all the $F_{I,\phi_{x_s,y_s,h_t}}$ can be derived from Eq. 9 and Eq. 10. As a result, most duplicated computations are avoided by simply considering the feature calculation from another novel point of view, i.e., the topology modeling of Haar-like features. This enables a much faster Adaboost trainer. More importantly, the topological model reveals an intrinsic property of negative samples: the feature values associated with different child-HL features from the same parent-HL feature should be almost the same at the first stage. This interesting property is desirable for enhancing the robustness of the learned detector. We will discuss the usefulness of the topological model in more details in Sec. 4.1.

However, it must be emphasized that although topology modeling is demonstrated on Haar-like features, it can easily be generalized to other features, because the underlying idea of topology modeling is not the topology of Haar-like features but the double-sliding mechanism that is common in Adaboost learning.

It must also be emphasized that although millions of child-HL features can be topologically modeled, they remain individually distinctive on positive samples. This is because the positive samples are not obtained by 'sliding/sampling over some a positive image'. For example, ϕ_{x_s,y_s,h_t} and $\phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$ in Fig. 3 on positive sample $I_{x,y}$ are distinctive because there does not exist positive sample $I_{x-\Delta_x,y-\Delta_y}$ (so that $I_{x-\Delta_x,y-\Delta_y} \cdot \phi_{x_s+\Delta_x,y_s+\Delta_y,h_t} = I_{x,y} \cdot \phi_{x_s,y_s,h_t}$) or $I_{x+\Delta_x,y+\Delta_y}$ (so that $I_{x+\Delta_x,y+\Delta_y} \cdot \phi_{x_s,y_s,h_t} = I_{x,y} \cdot \phi_{x_s+\Delta_x,y_s+\Delta_y,h_t}$). That is, $I_{x+\Delta_x,y+\Delta_y}$ and $I_{x-\Delta_x,y-\Delta_y}$ can not be positive samples if $I_{x,y}$ is a positive sample. Clearly, the huge number of distinctive child-HL features on positive samples must contain much redundancy, which requires an enhanced Adaboost learning algorithm for selecting the most discriminative ones for object detection.

3. Topology Oriented Adaboost Learning

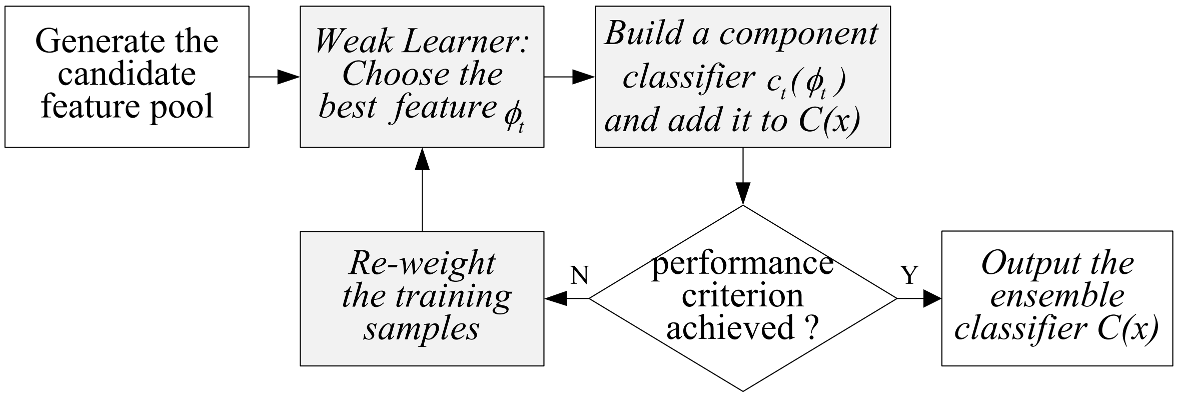

Generate the candidate feature pool
Weak Learner: Choose the best feature ϕ_t
Build a component classifier $c_t(\phi_t)$ and add it to C(x)
performance criterion achieved ?
N
Y
Re-weight the training samples
Output the ensemble classifier C(x)

Figure 4: A generalized flowchart of Adaboost learning.

Adaboost is a well-known large margin learning algorithm [6] that can select a small set of the most discriminative features from a candidate feature pool, and combines them into an additive model:

$$C(x) = \text{sign}\left(\sum_{t=1}^{T} c_t(\phi_t(x))\right) \tag{11}$$

where ϕ_t denotes the selected child-HL feature ϕ_{x_s,y_s,h_t}, $c_t(\phi_t(x))$ denotes the component classifier constructed based on ϕ_t. A generalized flowchart of Adaboost learning is illustrated in Fig. 4. We can see that it involves three key modules, namely the weak learner, the component classifier and the re-weighting function [9].

a) The Weak Learner: The weak learner is essentially the criterion for choosing the best feature ϕ_t on the weighted training set.

b) The Component Classifier: The component classifier $c_t(\phi_t(x))$ outputs the confidence of a sample being a positive based on its ϕ_t value.

c) Sample Re-weighting: Sample re-weighting enables that the subsequent component classifier can concentrate on the hard samples by assigning higher weights to the samples that are wrongly classified by previous component classifiers.

By tuning the above modules, several Adaboost variants have been proposed in the literature such as DiscreteBoost [20], BayesianBoost [21], RealBoost [6], KLBoost [14], etc. as shown in Table 1, where $P^+_{\phi_i}$, $P^-_{\phi_i}$ are the positive and negative probability distributions of ϕ_i on the weighted training set, L is the number of histogram bins of $P^+_{\phi_i}$ and $P^-_{\phi_i}$, and *FAR*, *FRR*, *AUC* denote *False Accept Rate*, *False Reject Rate*, *Area Under ROC*[3] curve respectively. From Table 1, we can see that DiscreteBoost [20] and the BayesianBoost [21] try to minimize some kind of classification error (e.g. Bayesian error in [21]), so that the selected feature is discriminative. However, their stump based component classifiers loss much discriminative information. In contrast, confidence-rated boosting [6, 14, 19] evolves with LUT-based component classifiers, however, their weak learners remain intermediate to the classification

[3]ROC: Receiver Operating Characteristics. ROC curve is a plot of FRR against FAR, see Fig. 8.

Adaboost Variants	**Weal Learner**	**Component Classifier**	**Reweighting Function**
Discrete Adaboost (Viola and Jones, 2004)	$\phi_t = \arg\min_{\phi_i} \sum_j w_j \delta_{[y_j \mathrm{sign}(p_i \phi_i(x_j) - \theta_i) > 0]}$	$c_t(x) = \mathrm{sign}\left(p_t \phi_t(x) - \theta_t\right)$	See Eq. 14
Bayesian Adaboost (Xiao et al., 2007)	$\phi_t = \arg\min_{\phi_i} \sum_l \min\left(P^+_{\phi_i}(l), P^-_{\phi_i}(l)\right)$	$c_t(x) = \mathrm{sign}\left(P^+_{\phi_t}(\phi_t(x)) - P^-_{\phi_t}(\phi_t(x))\right)$	
KL-Adaboost (Liu and Shum, 2003)	$\phi_t = \arg\max_{\phi_i} \sum_l \left[P^+_{\phi_i}(l) - P^-_{\phi_i}(l)\right] \log \frac{P^+_{\phi_i}(l)}{P^-_{\phi_i}(l)}$	$c_t(x) = \frac{1}{2} \log \frac{P^+_{\phi_t}(\phi_t(x)) + \epsilon}{P^-_{\phi_t}(\phi_t(x)) + \epsilon}$	
Real-Adaboost (Friedman et al., 2000)	$\phi_t = \arg\min_{\phi_i} \sum_l 2\sqrt{P^+_{\phi_i}(l) P^-_{\phi_i}(l)}$		
Topology Oriented Adaboost	$\phi_t = \arg\min_{\phi_i} AUC = \arg\min_{\phi_i} \int_0^1 FRR\, dFAR$		

Table 1: Several variants of Adaboost learning via tuning the weak learner, the component classifier.

error (i.e., not a tight approximation to the classification error). In other words, the feature that optimizes the weak learner (e.g., KL divergence [14]) can not necessarily minimize the classification error. This means unnecessary loss of efficiency of the weak learner in the viewpoint of classification error. Therefore, an desirable Adaboost algorithm should at least satisfy the following criteria: 1) its component classifier makes full use of the discriminative ability of the selected feature; 2) its weak learner immediately minimizes the classification error. This motivates the following Topology Oriented Adaboost algorithm.

3.1. Topology Oriented Adaboost Learning

The Topology Oriented Adaboost (TOBoost) works as follows:

a) The Component Classifier: In TOBoost, the confidence-rated (i.e. LUT-based) component classifier is adopted because it makes full use of the discriminative information of each selected feature. Given that the probability distributions of positive and negative samples are $P^+_{\phi_t}(\phi_t(x))$ and $P^-_{\phi_t}(\phi_t(x))$ respectively (which can be considered as empirical posterior probability), a common choice for the confidence being a positive sample is then the log ratio likelihood [6]:

$$c_t(\phi_t(x)) = \frac{1}{2} \log \frac{P^+_{\phi_t}(\phi_t(x)) + \epsilon}{P^-_{\phi_t}(\phi_t(x)) + \epsilon} \tag{12}$$

where ϵ is a regularization factor that deals with possible noise in $P^+_{\phi_t}(\phi_t(x))$ and $P^-_{\phi_t}(\phi_t(x))$.

b) The Weak Learner: Ideally, the weak learner should be able to choose the component classifier $c_t(\phi_t(x))$ that minimizes the classification error on the current weighted training samples. *ROC* (Receiver Operating Characteristic) curve has been considered as an appropriate indicator of the classification ability of a classifier. It is then desirable for the weak learner to choose the feature ϕ_t whose c_t produces the best *ROC* curve. In TOBoost, *AUC* (the Area Under the *ROC* Curve) is adopted to quantitatively measure the *ROC* curves.

$$AUC = \int_0^1 FRR\, dFAR \tag{13}$$

where (*FAR*, *FRR*) is a point on the *ROC* curve [9]. *AUC* corresponds to the summed classification error rate of a considered classifier, and therefore can be a qualified candidate for the weak learner.

Based on Eq. 12 and Eq. 13, we then establish an enhanced Adaboost algorithm. Its weak learner (Eq. 13) immediately minimizes the classification error, and its component classifier (Eq. 12) makes full use of the discriminative information of each selected feature. This is precisely the kind of Adaboost that we are seeking at the very beginning of this section.

c) Sample Re-weighting: The underlying idea of the sample re-weighting is to assign higher weights to the samples that are wrongly classified by previous classifiers, and vice versa. In this work, the traditional reweighting function is adopted [6]:

$$w_{t+1}(x_i) \leftarrow w_t(x_i)\exp(-y_i h_t(\phi_t(x_i))) \tag{14}$$

where $w_{t+1}(x_i)$ is the weight of sample x_i after adding the t-th component classifier, and $y_i \in \{+1, -1\}$ is the class label of x_i. Clearly, Equation 14 assigns higher weights to the samples that are wrongly classified by previous component classifiers so that the next component classifier can concentrate on these hard samples.

3.2. Cascade Structure

Cascade structure is advantageous because it significantly accelerates the execution speed in the run time by weeding out the vast majority of non-target-objects in the early stages which are relatively simple to evaluate. However, optimal tuning of cascade parameters during Adaboost-cascade learning is not trivial and can result in brittle detector [2, 7, 21]. Several cascade structures have been proposed in the literature, such as “Disjoint Cascade” [20], “Nested Cascade” [11], “Soft Cascade” [2] and “Dynamic Cascade” [21]. In “Disjoint Cascade” and “Nested Cascade”, the same cascade parameters (i.e., the max false alarm rate and the min hit rate) is assigned to each stage. This is obviously not reasonable because it tends to be harder and harder for the later stages to achieve these requirements, resulting in a time-consuming and brittle train procedure. In contrast, “Soft Cascade” and “Dynamic Cascade” evolve by assigning lower requirement to the later stages, and achieve promising performance. In this work, we follow this idea.

The key objective of cascade parameter tuning is to get a proper setting of the max false alarm rate and min hit rate for each stage under the requirement of the overall false alarm rate and hit rate. Similarly as “Soft Cascade” [2] and “Dynamic Cascade” [21], we assume that both the min hit rate and the max false alarm rate at each stage change exponentially as follows:

$$MinHitRate(t) = 1 - k_1 e^{-\alpha t/T} \tag{15}$$

$$MaxFalseAlarm(t) = k_2 e^{\alpha t/T} \tag{16}$$

where T is the desirable number of stages, k_1, k_2 is the normalization factor to satisfy the overall target performance requirement, α is the free parameter for the trade-off between detection speed and accuracy. The larger the α, the faster the resultant detector will be. In TOBoost, Eq. 15 and Eq. 16 are calculated given T and $k2$, under the condition that:

$$OverallHitRate \leq \Pi_t MinHitRate(t) \tag{17}$$

$$OverallFalseAlarm \geq \Pi_t MaxFalseAlarm(t) \tag{18}$$

From Eq. 15 and Eq. 16, we can see that both the max false alarm rate and the min hit rate at stage t become larger with t. The justifications are as follows: 1) a lower max false alarm rate (i.e., harder requirement on negative samples) in the early stages releases the training burden of later stages, and enables fast training; 2) a lower min hit rate (i.e., relatively looser requirement on positive samples) in the early stages (together with the lower max false alarm rate) allows us to weed out the majority of non-target-objects in the early stages, and enables fast execution.

3.3. Gaussian-KDE for $P^+_{\phi_t}$ and $P^-_{\phi_t}$

From Section 3.1, we can see that both the weak learner and the component classifier are dependent on $P^+_{\phi_t}$ and $P^-_{\phi_t}$, which are often empirically estimated via histograms. However, usually only a fraction of the candidate samples are maintained at each stage due to the cascade structure. When the number of training samples is limited, samples dropped into each bin can be insufficient for a stable estimation of the distribution but just an ad-hoc one of the current training set. Consequently, the component classifier learned based on the limited training samples will be ad-hoc, i.e., sensitive to noise and has low generalization ability [8].

A possible solution to this problem is to introduce invariance to the training set via kernel density estimation (KDE) [1]. The basic idea of KDE is to argument the training set using replicas of the existing training samples. In particular, the replicas are often drawn from an appropriate distribution (e.g. Gaussian) centered at the seminal sample. Following this idea, we directly introduce invariance into the feature values rather than the samples. Suppose

$p(\phi_t(x))$ is the probability density of a candidate feature ϕ_t, and we wish to estimate $p(\phi_t(x))$ via a random sample set $x_1, x_2, ..., x_N$. In Gaussian-KDE, $p(\phi_t(x))$ is estimated as follows [1]:

$$\hat{p}(\phi_t(x)) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi\sigma^2)^{1/2}} exp\left\{-\frac{|\phi_t(x) - \phi_t(x_n)|^2}{2\sigma^2}\right\} \tag{19}$$

where σ denotes the standard deviation of the Gaussian kernel. Here Gaussian kernel is adopted for the sake of smoothness [1, 8]. Thus our density model is obtained by placing a Gaussian over each feature value of the training data and then adding up the contributions over the whole data set. The adoption of Gaussian kernel density estimation can lead to significant improvement in generalization ability with neglectable computation, especially when the number of training samples are limited [8].

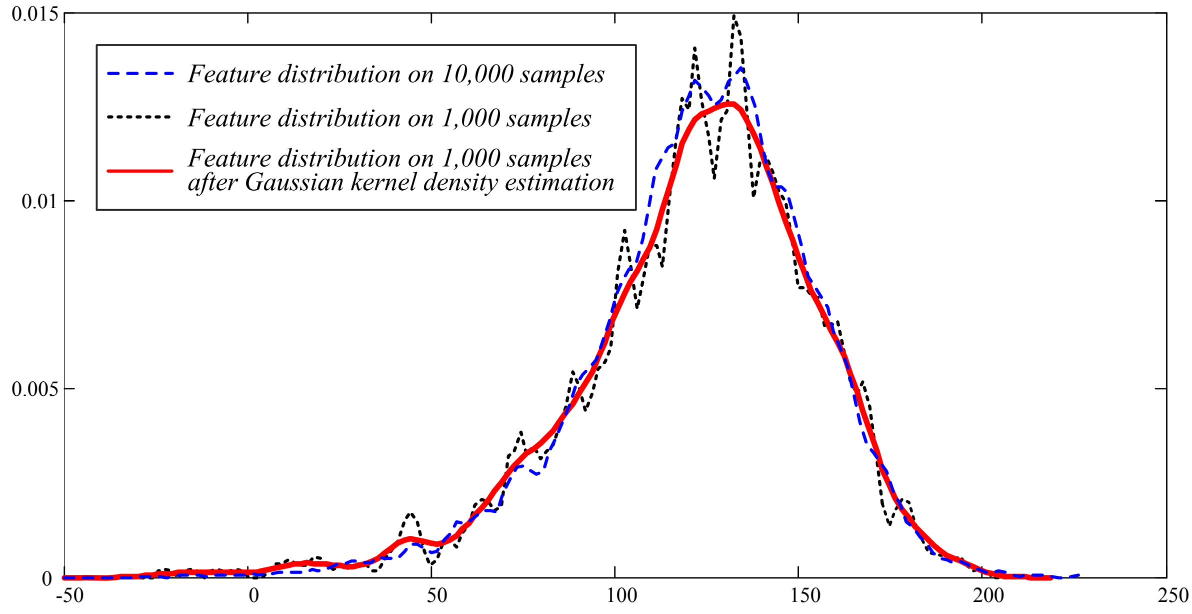


Figure 5: The distributions of a candidate child-HL feature on 10, 000 iris images, 1, 000 iris images, and the Gaussian-KDE version of the 1, 000 one. We can see that Gaussian KDE provides a better estimation of the density distribution while using much less samples.

The usefulness of Gaussian kernel density estimation is illustrated in Fig. 5, where the distribution obtained on 10,000 iris images is surprisingly similar with the distribution obtained on only 1,000 iris images with Gaussian kernel density estimation. This indicates that we can get a much better density estimation by Gaussian-KDE while using much less training samples. As a result, the generalization ability of the final detector is expected to be enhanced. A notable point in Gaussian KDE is the setting of σ which controls the trade-off between having $\hat{p}(\phi_t(x))$ close to the data (at small σ) and having $\hat{p}(\phi_t(x))$ smooth (at large σ). A reasonable σ value should be set according to the number of available training samples. For example, it is 1.5 for 30, 000 samples while 2.0 for 10, 000 samples in our experiments.

4. Experiments and Discussions

In this section, we evaluate the proposed algorithm in the context of iris detection[4]. The objective of iris detection is not only identifying the presence of iris in input video sequences but determining its position and scale as well. This is quite different from iris localization or segmentation whose goal is to locate the valid part of iris for subsequent

[4]In this work, iris detection is adopted only for the purpose of explanations and experimental demonstrations. Instead of trapping ourselves on the detailed descriptions on iris detection, we concentrate ourselves on explaining the basic ideas of these general methods.

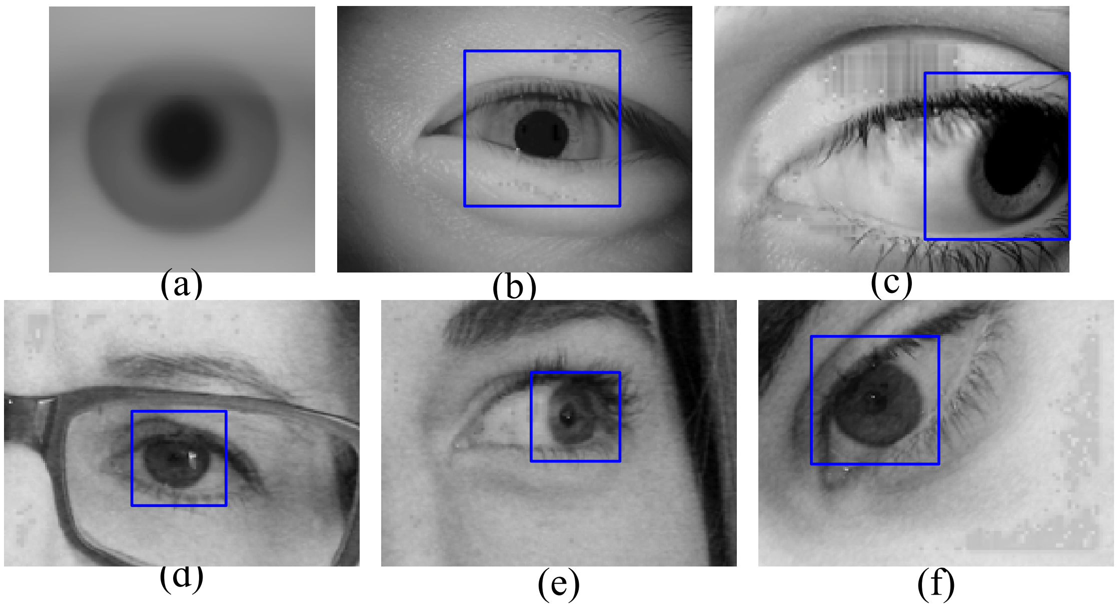


Figure 6: Illustration of TOBoost iris detector. (a) The desirable iris structure. (b)-(f): Detection results of the TOBoost detector. We can see that encouraging performance has been achieved by our TOBoost detector in the presence of occlusions, oblique view-angles, etc.

processing[5]. As an important module in iris biometrics, iris detection is challenging due to various types of noise in the iris images such as occlusions, oblique view-angles, etc. [10]. Fortunately, iris posses an annular structure that is desirable for Adaboost learning (see Fig. 6(a)). Therefore, iris detection should be another typical application of Adaboost-cascade detector except face detection. In this work, three challenging iris image databases [10] (i.e., CASIA-Iris-V3 [5], ICE v1.0 [16] and UBIRIS v1.0 [17]) are adopted for Adaboost learning and test, with more than 60,000 iris images in total. We also collected more than 20,000 non-iris images in the web as the negative images, from which billions of negative samples can be cropped. Extensive comparison experiments (with DiscreteBoost [20], RealBoost [6], "Dynamic Cascade" [2], etc.) are performed based on this large data set to demonstrate the efficiency and effectiveness of the proposed algorithm.

4.1. Overall Training Speed

AdaBoost Alg.	Discrete Boost (Viola&Jones, 2004)	Nested Cas. (Huang, 2007)	Dynamic Cas. (Xiao, 2007)	TO-Boost
Time(min)	92	56	49	36

Table 2: The training speed of different Adaboost algorithms.

One advantage of the topology oriented Adaboost is its training effectiveness. The training speed of TOBoost is expected to be faster than traditional Adaboost-cascade algorithms. To evaluate the actual effects, we train detectors with the same setting (i.e., the same training set and the same required overall error rates) by different Adaboost-cascade algorithms. The time cost of each algorithm is listed in Table 2. As expected, TOBoost is the fastest. This is because: 1) topology modeling of Haar-like feature avoids much duplicated computations; 2) Non-adjacent Haar-like features are more discriminative and stable for object detection; 3) the *AUC*-oriented weak learner is more aggressive and 4) the proposed cascade calibration scheme results in more reasonable cascade parameters.

In addition, "Nested Cascade" [11] and "Dynamic Cascade" [21] are both faster than the original Discrete Adaboost [20], because the later wastes the discriminative information of previous stages. "Dynamic Cascade" is

[5]Please refer to [10] for other state-of-the-art iris localization methods.

slightly faster than “Nested Cascade” because “Nested Cascade” uses the same cascade parameter for each stage, and therefore make it difficult to converge in the later stages.

4.2. The Usefulness of Topology Modeling of Haar-like Features

As described in Section 2, the feature calculation on negative images is accelerated because most duplications are avoided via topology modeling of Haar-like features. It must be noted that this is true only under the assumption that negative samples are extracted in adjacent pixel-wise sliding windows. In practical implementation of Adaboost-cascade learning, the windows would not slide but jump for training speedup. Therefore, the overlap is limited and the speed benefit of this topological model would be degraded. However, even in this case, the topology modeling of Haar-like features are still useful. The reasons are as follows:

1. As indicated in Eq. 6, in the pixel-wise sliding mode the number of feature values that should be extracted from one negative image I associated with one parent-HL feature h_t is:
$$N^{slide}_{I,\phi_{h_t}} = N_s \times M \tag{20}$$
where N_s is the number of candidate samples that can be cropped from I, and M is the number of child-HL features ϕ_{h_t} derived from the parent-HL feature h_t. Suppose the horizontal and vertical sampling rates (in the jump mode) are S_x and S_y respectively, then the number of feature values we should calculated becomes:
$$N^{jump}_{I,\phi_{h_t}} = \frac{N_s}{S_x S_y} \times M \tag{21}$$
On the other hand, according to the topological model, the number of feature values we have to calculate from one negative image I associated with one parent-HL feature h_t is:
$$N^{slide}_{I,h_t} \simeq N_s \tag{22}$$
From the above three equations, we can see that the topological model should be faster in case $M > S_x S_y$.
2. Except for the speed consideration, another important issue is the number of samples used in Adaboost learning. From Eq. 21 we can see that the number of calculated negative samples associated with one specific child-HL feature ϕ_{h_t} (in the jump mode) is $N_s/S_x S_y$. On the other hand, in the topological model, the number of calculated negative samples is N_s. This clearly indicates that more samples are used in TOBoost compared with the traditional jump mode. And according to our experience, more samples usually result in more stable detector with larger generalization ability.
3. Most importantly, topological model reveals an intrinsic property of negative samples: the feature values associated with different child-HL features from the same parent-HL feature should be almost the same at the first stage (Refer to Eqs. 8- 10). However, in the traditional jump mode, different child-HL features tend to have different feature distribution due to the jump mode in the extraction of negative samples. This conflicts with this intrinsic property. Here we argue that maintaining this intrinsic property is advantageous because: in object detection, our goal is to reveal the specific characteristics (the discriminative features) that all positive samples hold, rather than the discriminative features associated with negative samples. In other words, we should focus on the discriminative features of positive samples while suppressing those of negative samples, so that the resultant detector will have larger generalization ability, i.e., have less dependence on the negative training samples.

Therefore, the speed acceleration is only one of the advantages of topology modeling. Other advantages include more possible samples than jump mode, and the preserving of the intrinsic property of negative samples.

4.3. The Effects of Non-adjacent Features

An important issue in Adaboost is a sufficiently large feature pool. The required number of Haar-like features to train a TOBoost detector on the basic [20], extended [13] and the generalized Haar-like features are listed in Table 3. From Table 3 we can see that the required number of Haar-like features decreases with the feature pool growing larger. This demonstrates the importance of larger feature pool and in particular the usefulness of the non-adjacent Haar-like features (which, as expected, should be more efficient and more stable in object detection). A few examples of the selected child-HL features are illustrated in Fig. 7. We can see that the selected features are representative for the particular annular iris structure. Moreover, it is clear that non-adjacent Haar-like features play a leading role.

Type of HL Features	Basic (Viola&Jones,2004)	Extended (Lienhart&Maydt, 2002)	Topological Generalized
Num (♯)	581	492	403

Table 3: The required number of features to train a TOBoost detector with Basic, Extended and Generalized Haar-like features.

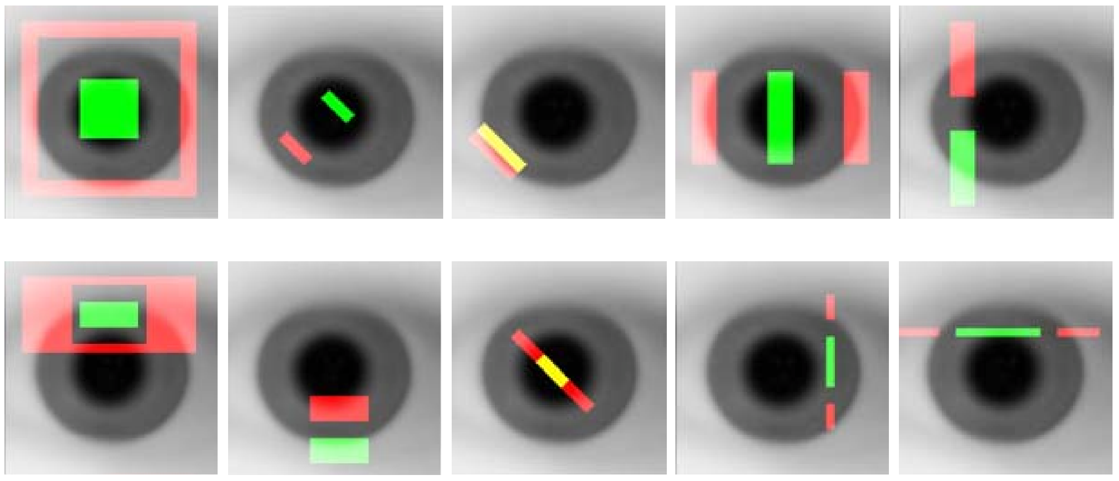


Figure 7: Illustration of selected Haar-like features by TOBoost algorithm.

4.4. The Effects of the Weak Learner

In this subsection, we evaluate the effects of the weak learner to Adaboost learning. Intuitively, it is desirable for the weak learner to select the feature that obtain the smallest classification error on the weighted training samples. To compare the efficiency of different weak learners, Figure 8 shows the *ROC* curves of the first three component classifiers selected by different weak learners listed in Table 1. It is clear that the proposed *AUC* based weak learner outperforms others by selecting the most discriminative features. This can also be partly demonstrated by the required number of features when training a detector with different weak learners. Clearly, the smaller the number of features, the more discriminative the selected feature (i.e. the more efficient of the weak learner). Our experimental results show that the smallest number is achieved by TOBoost.

4.5. The Generalization Ability of TOBoost

NO.of Pos Samples	Discrete AdaBoost [20]	Dynamic Cascade [21]	TO-Boost
5000	88.3%	92.5%	94.4%
10000	93.8%	95.6%	96.7%
30000	96.7%	98.1%	99.2%

Table 4: The detection rates of detectors learned on different training sets via different Adaboost algorithms.

To evaluate the generalization ability of TOBoost, we train detectors using training sets of different sizes with several Adaboost algorithms, and compare their performance on the same test set (with 30,000 iris images). The detection rate is adopted as an indicator of the generalization ability of the learned detector. The test results are shown in Table 4. We can see that all detectors become better while the training set growing larger, which clearly show the importance of large training set for Adaboost learning. Table 4 also shows that TOBoost achieves higher performance at all training sets and its performance degradation is much smaller than others while the train set growing smaller. In our opinion, the reasons are two-folds. Firstly, the topology modeling of Haar-like features allows us to maintain much more negative samples (than the traditional jump-based extraction of negative samples) during

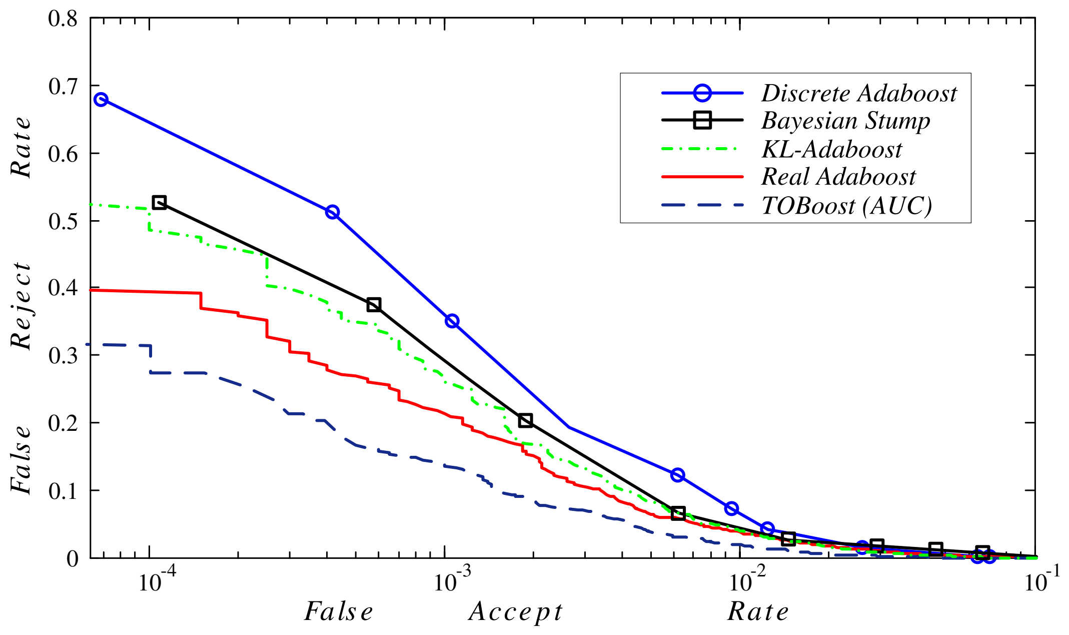


Figure 8: The *ROC* curves of the first three component classifiers selected by different weak learners. We can see that *AUC* immediately minimizes the classification error.

training and forces the learner to concentrate on the discriminative features of positive samples, which is desirable for generalization. Secondly, the Gaussian-KDE scheme is capable to incorporate invariance into the training samples, which also enhances the generalization ability.

Figure 6 gives several illustrations of detection results of the iris detector learned by topology oriented boosting[6]. We can see that encouraging performance has been achieved by our TOBoost detector in the presence of various occlusions, oblique view-angles, etc.

4.6. Further Discussions

From the above descriptions, we can see that there are two key directions for the improvement of Adaboost-cascade learning. The first direction is to make Adaboost detectors more efficient. This can be achieved by adopting more discriminative feature sets, more aggressive weak learners and component classifiers. The second direction is to improve the generalization ability of Adaboost detectors. Being a supervised learning algorithm, Adaboost is rather aggressive [6, 15] and over-fitting is almost inevitable. However, we can still try to enhance its generalization ability by several meanings, such as using more training samples or incorporating invariance into the samples via Gaussian kernel density estimation. As discussed in Sec. 4.2, the topology modeling of Haar-like features is also helpful for a more stable detector.

Moreover, in practical biometrics applications (e.g. face and iris biometrics), we find that sometimes it is a good idea to train a moderately good detector. A moderate detector tends to miss some objects that are of low quality, and therefore avoid possible false alarms due to image quality degradation in biometric applications. Therefore, careful design of the most desirable detector according to specific applications is also important in practical implementation of Adaboost-cascade learning.

5. Conclusions

In this paper, we have presented five novel ideas for Adaboost-cascade learning, which enable a more efficient and stable object detector with fewer features. Firstly, non-adjacent Haar-like features are adopted for efficient object

[6]Note that the specular spots in these images has been eliminated to keep the iris structure uninterrupted (with the techniques described in [10]).

representation. Secondly, the Haar-like features are modeled by their topological properties. This topological model reveals an interesting intrinsic property of Haar-like features (at the first stage), which not only avoids duplicated computations on negative samples but also enhances the robustness of Adaboost learning. Thirdly, a novel topology oriented Adaboost algorithm is proposed. TOBoost's weak learner immediately minimizes the classification error, enabling the selected features to be more discriminative. Fourthly, a simple and reasonable scheme is presented for cascade design. Finally, Gaussian-KDE is introduced into Adaboost to enhance its generalization ability. Extensive experiments in iris detection have shown that the proposed algorithm achieves state-of-the-art performance.

In our future work, we will explore the application of TOBoost in non-cooperative iris recognition systems (e.g. iris recognition at-a-distance and on-the-move) which involve face and eye detection in addition to iris detection.

Acknowledgement

This work is supported by research grants from the National Hi-Tech Research and Development Program of China (Grand No. 2007AA01Z162), and the Natural Science Foundation of China (Grant No. 60723005).

References

[1] Bishop, C. M., 2006. Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, New York, NY, Ch. 6, pp. 291–324.

[2] Bourdev, L., Brandt, J., 2005. Robust object detection via soft cascade. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'05). Vol. 2. pp. 236–243.

[3] Brubaker, S. C., Wu, J. X., Sun, J., Mullin, M. D., Rehg, J. M., 2008. On the design of cascades of boosted ensembles for face detection. International Journal of Computer Vision 77 (1), 65–86.

[4] Buhlmann, P., Yu, B., Apr. 2000. Invited discussions on 'additive logistic regression: a statistical view of boosting (friedman, hastie and tibshirani)'. The Annals of Statictics 28 (2), 377–386.

[5] CASIA, 2008. Casia iris image database v3.0. Center for Biometrics and Security Research.
URL `http://www.cbsr.ia.ac.cn/IrisDatabase.htm`

[6] Friedman, J., Hastie, T., Tibshirani, R., Apr. 2000. Additive logistic regression: a statistical view of boosting. The Annals of Statictics 28 (2), 337–374.

[7] Grossmann, E., 2004. Automatic design of cascaded classifiers. In: Lecture Notes in Computer Science. Vol. 3138. pp. 983–991.

[8] He, Z., Tan, T., Sun, Z., 2009. Encient iris spoof detection via boosted local binary patterns. In: Proc. of 3rd IAPR/IEEE Int'l Conf. on Biometrics, Lecture Notes on Computer Science (LNCS). Vol. 5558. Italy, pp. 687–697.

[9] He, Z., Tan, T., Sun, Z., Qiu, X. C., 2008. Boosting ordinal features for accurate and fast iris recognition. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'08). pp. 1–8.

[10] He, Z., Tan, T., Sun, Z., Qiu, X. C., Sep. 2009. Towards accurate and fast iris segmentation for iris biometrics. IEEE Trans. on Pattern Analysis and Machine Intelligence, Accepted 31 (9), 1670–1684.

[11] Huang, C., Ai, H., Li, Y., Lao, S., 2007. High-performance rotation invariant multiview face detection. IEEE Trans. on Pattern Analysis and Machine Intelligence 29 (4), 671–686.

[12] Li, S. Z., Zhang, Z. Q., 2004. Floatboost learning and statistical face detection. IEEE Trans. on Pattern Analysis and Machine Intelligence 26 (9), 1112–1123.

[13] Lienhart, R., Maydt, J., 2002. An extended set of haar-like features for rapid object detection. In: Proc. of IEEE Int'l Conf. on Image Processing (ICIP'02). Vol. 1. pp. 900–903.

[14] Liu, C., Shum, H.-Y., 2003. Kullback-leibler boosting. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'03). Vol. 1. pp. 587–594.

[15] Mease, D., Wyner, A., 2008. Evidence contrary to the statistical view of boosting. Journal of Machine Learning Reserch 9, 131–151.

[16] NIST-ICE, 2005. Ice v1.0 iris image database. The National Institute of Standards and Technology,NIST.
URL `http://iris.nist.gov/ICE/`

[17] Proença, H., Alexandre, L. A., 2005. Ubiris: a noisy iris image database. In: Lect. Notes Comput. Sci. Vol. 3617. pp. 970–977.
URL `http://iris.di.ubi.pt.`

[18] Schapire, R. E., 1999. A brief introduction to boosting. In: Proc. of the 16th Int'l Joint Conf. on Articficial Intelligence.

[19] Schapire, R. E., Singer, Y., 1999. Improved boosting algorithms using confidence-rated predictions. Machine Learning 37, 297–336.

[20] Viola, P., Jones, M., 2004. Robust real-time face detection. International Journal of Computer Vision 57 (2), 137–154.

[21] Xiao, R., Zhu, H., Sun, H., Tang, X., 2007. Dynamic cascades for face detection. In: Proc. of IEEE 11th Int'l Conf. on Computer Vision (ICCV'07). pp. 1–8.