



Fast Feature Value Searching for Face Detection

Yunyang Yan

Department of Computer Engineering

Huaiyin Institute of Technology

Huai'an 223001, China

E-mail: areyyyke@163.com

Zhibo Guo

School of Information Engineering

Yangzhou University

Yangzhou 225009, China

E-mail: zhibo_guo@163.com

Jingyu Yang

School of Computer Science and Technology

Nanjing University of Science and Technology

Nanjing 210094, China

E-mail: jingyuyang@mail.njust.edu.cn

This research is supported in part by the NSFC under grant 60632050, the High School Technology Fund of Jiangsu province under grant 06KJD520024, and Technology Fund of Huai'an under grant HAG05053 and HAG07063.

Abstract

It would cost much and much time in face detector training using AdaBoost algorithm. An improved face detection algorithm called Rank-AdaBoost based on feature-value-division and Dual-AdaBoost based on dual-threshold are proposed to accelerate the training and improve detection performance. Using the improved AdaBoost, the feature values with respect to each Haar-like feature are rearrange to a definite number of ranks. The number of ranks is much less than that of the training samples, so that the test time on each training samples is saved corresponding to the original AdaBoost algorithm. Inheriting cascaded frame is also proposed here. Experimental results on MIT-CBCL face & nonface training data set illustrate that the improved algorithm could make training process convergence quickly and the training time is only one of 50 like before. Experimental results on MIT+CMU face set also show that the detection speed and accuracy are both better than the original method.

Keywords: Rank- AdaBoost, Feature value division, Dual-AdaBoost, Face detection, Hnheriting cascade

1. Introduction

For its interesting applications, automatic face detection has received considerable attention among researchers in many fields, such as content-based image retrieval, video coding, video conference, crowd surveillance, and intelligent human-computer interface.

Many methods have been proposed to detect faces in a gray image or a color image, such as Template Matching, Mosaic Image, Geometrical Face Model, Difference Pictures, Snake, Deformable templates, Statistical Skin Color Models et al. Now the methods based on statistical learning algorithms have attracted more attention, including PCA, Artificial Neural Networks and Support Vector Machines, Bayesian Discriminant Features, etc. These methods show good performance in the detection precision, but their detection speed needs to be increased (Liang Luhong, 2002, pp.449-458).

The first real-time face detector was proposed by Viola and Jones (Viola and Jones, 2001, 2004, pp.137-154). They described a face detection framework that is capable of processing images extremely rapidly while achieving high detection rates. There are three key contributions of this detection framework. The first is the introduction of a new

image representation called an “Integral Image” which allows the features used by the detector to be computed very quickly. The second is a simple and efficient classifier which is built using the AdaBoost learning algorithm to select a small number of critical visual features from a very large set of potential features. The third contribution is a method for combining classifiers in a “cascade” which allows background regions of the image to be quickly discarded and focus on promising face-like regions. Simple Haar-like features are extracted and used as weak classifiers. The Viola and Jones frontal face detector runs at about 15 frames per second on 320 by 240 image. As in the work of Rowley (Rowley et al,1998,pp.22-38) and Schneiderman(Schneiderman,2000,pp.746-751), Viola and Jones(Viola and Jones, 2003) built a multi-view face detector with AdaBoost to handle profile views and rotated faces.

However, one problem with these approaches is that there are too many Haar-Like features in a single face image. Another difficulty is that a great deal of non-face training samples are used to reach good performance. The big set of training samples not only slow down the training, but also increase the number of weak classifiers greatly in cascaded detector. So AdaBoost on every round needs to search a large pool of candidate weak classifiers and the computation is very complex.

Attempting to get more efficient detector, improved AdaBoost algorithm called Rank-AdaBoost and Dual-AdaBoost based on feature-value-division are proposed to speed-up the training and detection performance. Firstly, for one Haar-Like feature, distribution of feature values of all face samples is divided into definite ranks. A small quantity of value is used as possible threshold in training instead of all feature values of face samples. Then, the approach of fast dual-threshold finding is developed in the enhanced AdaBoost algorithm, which makes the training process faster and the detection accuracy higher. In the training process of cascaded detector, the formers classifiers are transferred to the later, so that more non-face would be ignored. Both computational speed and the performance are improved obviously by this approach simultaneously.

Experimental results on MIT-CBCL face set and MIT+CMU face set show that our method yields higher classification performance than Viola and Jones' both on training speed and detection accuracy.

2. AdaBoost using Haar-Like feature

2.1 Haar-Like features and integral image

Haar-Like features are a kind of simple rectangle features proposed by Viola et al. as shown in Figure 1. The squares represent a face image. The value of each Haar-Like feature is computed as a difference of the sum over the white and black regions. It describes the local gray feature in the image. Using parity and threshold on this value, a class is predicted.

Viola et al. use three kinds of features. They differ by their division of two, three or four rectangular areas. Rotating these three types could easily generate other kinds of features. Every feature is characterized by its position in the face frame, pre-specified size and type. Given that the base resolution of the classifier is 24 by 24 pixels, the exhaustive set of rectangle filters is quite large, over 100,000, which is roughly $O(N^4)$ where $N=24$ (i.e. the number of possible locations times the number of possible sizes). The actual number is smaller since filters must fit within the classification window.

Computation of rectangle filters can be accelerated using an intermediate image representation called the integral image. Using this representation any rectangle filter, at any scale or location, can be evaluated in constant time. Integral image at location (x, y) is computed as the sum of the pixel values above and to the left of (x,y) . A original gray image I and its integral image II is described as follows:

$$II(x, y) = \sum_{i,j=1}^{x,y} I(i, j)$$

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

$$S(x, y) = S(x, y - 1) + I(x, y)$$

$$II(x, y) = II(x - 1, y) + S(x, y)$$

(where $S(x, y)$ is the cumulative row sum, $S(x, -1) = 0$, and $II(-1, y) = 0$). Using the integral image, the value of a Haar-Like feature can be computed by plus or minus using the integral image. Any rectangular sum can be calculated in four array references. Clearly the difference between two rectangular sums can be calculated in eight references. Since the two-rectangle features defined above involve adjacent rectangular sums they can be computed in six array references, and eight and nine references in the cases of three and four-rectangle features respectively.

In Figure 2, (d) is an integral image corresponding to the left image (a) and the feature (b). (c) shows the feature on the image. Simple $p4+p1-p2-p3-(p6+p3-p4-p5)$ could give the value of the feature. Much computing time is saved.

2.2 AdaBoost algorithm

In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple learning

algorithm (e.g., it might be used to boost the performance of a simple perceptron). It does this by combining a collection of weak classification functions to form a stronger classifier. In the language of boosting the simple learning algorithm is called a weak learner. So, for example, the perceptron learning algorithm searches over the set of possible perceptrons and returns the perceptron with the lowest classification error. The learner is called weak because we do not expect even the best classification function to classify the training data well (i.e. for a given problem the best perceptron may only classify the training data correctly 51% of the time). In order for the weak learner to be boosted, it is called upon to solve a sequence of learning problems. After the first round of learning, the examples are re-weighted in order to emphasize those who were incorrectly classified by the previous weak classifier. The final strong classifier takes the form of a perceptron, a weighted combination of weak classifiers followed by a threshold.

$$f(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

where $f(x)$ is the final strong classifier, $(h_1(x), h_2(x), \dots, h_t(x))$ is the serials of weak classifiers. The $h_t(x)$ can be thought of as one feature with a threshold. The original form of the AdaBoost named as Init-AdaBoost is shown as:

1) Given example images: $(x_1, y_1), \dots, (x_L, y_L)$, where $y_i \in \{1, 0\}$ indicates positive or negative examples; $g_j(x_i)$ is the j th

$$w_{1,i} = \begin{cases} 0.5/m & i \leq m, \\ 0.5/n & \text{otherwise} \end{cases}$$

Haar-Like feature of i th example x_i .

2) Initialize weights

Where m, n are the number of positive or negative examples respectively. $L = m + n$.

3) For $t = 1 \dots T$

a. Normalize the weights

$$w_{t,i} = w_{t,i} / \sum_{j=1}^L w_{t,j}$$

b. For each feature j , train a weak classifier h_j , and evaluate its error ε_j with respect to w_t , $\varepsilon_j = \sum_{i=1}^L w_{t,i} |h_j(x_i) - y_i|$,

$$h_j(x) = \begin{cases} 1 & p_j g_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

Where $p_j \in \{1, -1\}$ is a parity bit and θ_j is a threshold.

c. Choose the classifier h_t with the lowest error ε_t

d. Update the weights $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$, where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.

4) Final classifier:

$$H(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq 0.5 \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log(1/\beta_t)$.

2.3. Cascaded detector

In an image, most sub-images are non-face instances. In order to improve computational efficiency greatly and also reduce the false positive rate, a sequence of gradually more complex classifiers called a cascade is built.

An input window is evaluated on the first classifier of the cascade and if that classifier returns false then computation on that window ends and the detector returns false. If the classifier returns true, then the window is passed to the next classifier in the cascade. The next classifier evaluates the window in the same way. If the window passes through every classifier with all returning true, then the detector returns true for that window. The more a window looks like a face, the more classifiers are evaluated on it and the longer it takes to classify that window. Since most windows in an image do not look like faces, most are quickly discarded as non-faces. Figure 3 describes the cascade.

By using cascaded detector, it is possible to use smaller and efficient classifiers to reject many negative examples at early stage while detecting almost all the positive instances. Classifiers used at successive stages to examine difficult cases become more and more complex.

Since easily recognizable non-face images are classified in the early stages. Subsequent classifiers are trained only on examples that pass through all the previous classifiers. That is to say, classifiers of the later stages of the cascaded detector can be trained rapidly only on the harder, but smaller, part of the non-face training set. Therefore this detection approach would save the computational cost and maintain the performance simultaneously.

Stages in cascade are constructed by training classifiers using AdaBoost. Cascaded detector is trained as follows:

1) Input: Allowed false positive rate f , and detection rate d per layer; target overall false positive rate F_{target} . P denotes set of positive examples, N denotes set of negative examples, n_i is the number of weak classifiers in the i th layer classifier.

2) $F_0 = 1, D_0 = 1, i=0$

3) While $F_i > F_{target}$

a. $i++$, $n_i=0$, $F_i=F_{i-1}$

b. While $F_i > f \times F_{i-1}$

• n_i++ .

• Use P and N to train a i th layer classifier with n_i weak features using AdaBoost.

• Evaluate current cascaded classifier on validation set to determine F_i and D_i .

• Decrease threshold for the i th classifier until the current cascaded classifier has a detection rate D_i of at least $d \times D_{i-1}$, evaluate F_i .

c. If $F_i > F_{target}$ then evaluate the current cascaded classifier on the set of non-face images, and put any false detection into the N .

3. Proposed Improved AdaBoost

3.1 Problem of time cost

In AdaBoost algorithm, the step 3)b time costs expensively, because all simple classifier $h_j(j=1:k)$ is desired to compute where k is the number of the Haar-Like features and the k is a very large number. Moreover each h_j is obtained by exhaustive searching every samples so that it would take much time to only get a weak classifier $h_j(x)$.

If the *Onetime* is the time needed to get only one simple classifier, and the training time to get one weak classifier is *Traintime*, then

$$Traintime = k * Onetime.$$

The step is done repeatedly until the face detection rate is satisfied. If the number of weak classifiers is T , then it would cost *Alltime* to obtain the final strong classifier.

$$Alltime = Traintime * T = k * Onetime * T$$

If there are 600 weak classifiers needed, the mean search time of each h_j is 0.1s, only 24,000 features is used to train, then the consumed time would be $600 \times 24,000 \times 0.1 = 1,440,000$ s (i.e. 400 hours, about 16 days). It is too long. So it is necessary to save the time for computing h_j .

3.2 Division of the feature values

According to Init-AdaBoost, for one Haar-like feature, the feature value of each face sample is used as possible threshold. Under a given parity, the possible threshold is compared with that of all training samples. Consequently the false detection rate is calculated. Do it again under another parity. The threshold and the parity with this Haar-like feature could be determined by the face sample which causes the minimum false detection rate, and the feature value of this face sample is used as the threshold for the Haar-like feature. The cost time is $O(m \times n)$ where m, n is the number of face samples and non-face samples. In face detection training, m, n are all very large, generally thousands even million. So the time corresponding to $O(m \times n)$ is very expensive.

By the experiment results, we find that false detection occurred frequently when there are some noise on the image and we also find that there are little difference between the feature values of some face samples, the ability of these feature values to discriminate face sample and non-face sample is similar. So to reduce the number of possible threshold, we could get the maximum and the minimum feature values of all training face samples with one Haar-like feature, and obtain r ranks of the feature value from minimum to maximum feature value. That is:

$$\Delta_j = (\max(g_j(x_i)) - \min(g_j(x_i))) / r, \quad i=1 \dots m$$

Where $g_j(x_i)$ is the j th Haar-Like feature of i th example x_i .

Then using each Th_k as a possible threshold to find the weak classifier and its threshold with the parity. Th_k is computed by $Th_k = \min(g_j(x_i)) + k \times \Delta_j$ ($k=1 \dots r$).

Now the cost time would be $O(r \times (m+n))$ instead of $O(m \times n)$. After do many experiments, we find that the detection performance could be the same as Init-AdaBoost when r is less than 100. If let $r=100$, then the value of $r \times (m+n)$ would be smaller and smaller than $m \times n$ because m, n are usually thousands even millions. So the time used to train would be falled rapidly and could be denoted as $O(m+n)$.

Furthermore, let W_k is the sum of weight of every samples under threshold Th_k , G_k is the sum of weight of the samples whose feature value is more than Th_k but less than Th_{k+1} , then the sum of weight of every samples under threshold Th_{k+1} is W_{k+1} .

$$W_{k+1} = W_k + G_k.$$

That is to say only those samples are tested whose feature value is more than Th_k , and it is not necessary to test all samples to calculate W_{k+1} when W_k has been known. So the time for training could be also saved.

The improved AdaBoost based on the division of the feature values is called Rank-AdaBoost.

3.3 Finding of dual-threshold

According to experimental results, the local feature distributes regularity corresponding to a weak classifier. The typical local features of face and non-face are shown in Figure 4.

In Figure 4, vertical axis y says the ratio of face or non-face in total samples, the horizontal axis x is values of feature. Threshold used to indicate face or non-face could be obtained rapidly. As Figure 4 shows, face is higher than non-face from $\theta^{(1)}$ to $\theta^{(2)}$, so $\theta^{(1)}$ and $\theta^{(2)}$ are used as dual-threshold. The specific method to find threshold is described as follows.

- 1) For each x , compute $face(x) - nonface(x)$.
- 2) Choose x' with the maximum: $x' = \text{argmax} (face(x) - nonface(x))$.
- 3) From x' to left or right to find the crossing point which cause $face(x) - nonface(x) < 0$. If no crossing point is found, then the boundary point is selected as crossing point. So we could get two crossing points and use them as dual-threshold. AdaBoost based dual-threshold is called Dual-AdaBoost.

When determining a feature as a current weak classifier or not, the dual-threshold of the features may be adjusted to ensure weak classifier h_i to meet the demand of detection. So this method can greatly accelerate each weak classifier search. If there were 24,000 features, the search time would be 1/50 of that of exhaustive search.

4. Proposed Inheriting cascaded detector

In cascaded detector, fewer and fewer sub-window images need to be detected by layer classifier at later stages, a little error would make overall detection performance decline. So, a sequence of gradually more complex and more powerful classifiers are trained to increase classification performance with examples that have pass through all the previous classifiers. During cascaded detector training, the predecessor is used as a part of its successor, that is to say each layer is considered not only as an independent node of the cascaded classifier but also as a component of its successor. So the later classifier includes more classification features.

$$f_k(x) = f_{k-1}(x) + \sum_{i=1}^T \alpha_i h_i(x)$$

where $f_k(x)$ and $f_{k-1}(x)$ is a strong classification function on the k th and $k-1$ th layer respectively. It is called inheriting cascaded detector. Based on this algorithm, the overall performance of the cascaded detector is enhanced. Moreover, the threshold of classifier on different layers is adjusted to separate the training samples of face and non-face as far as possible, so that more non-face would be ignored. Both computational speed and the performance are improved obviously by this detection approach simultaneously.

As an example, we need to train strong classifiers on the k th layer. $f_k(x)$ and $H_k(x)$ is a strong classification function and strong classifier respectively. A total of L samples consist of m positive examples (face) and n negative examples (non-face). Positive examples arrange in a sequence before n negative examples.

- 1) Given example images: $(x_1, y_1), \dots, (x_L, y_L)$, where $y_i \in \{1, -1\}$ represents positive or negative examples; $g_j(x_i)$ is the j th Walsh feature of i th example x_i . $L = m + n$.
- 2) Using the last weights resulted from the strong classifiers training on $(k-1)$ th layer as $w_{1,i}$. If $k=1$ then initialize

$$w_{1,i} = \begin{cases} 0.5/m & i \leq m, \\ 0.5/n & \text{otherwise} \end{cases}$$

weights

- 3) Search $\theta_j^{(1)}$ and $\theta_j^{(2)}$ of each local feature based on its distribution in all face and non-face examples. Use $\theta_j^{(1)}$ and $\theta_j^{(2)}$

(2) as dual-threshold .

4) For $t = 1, \dots, T$

a. Normalize the weights

$$w_{t,i} = w_{t,i} / \sum_{j=1}^L w_{t,j}$$

b. For each feature j , get a weak classifier h_j with $\theta_j^{(1)}$ and $\theta_j^{(2)}$ by the method discussed above, and evaluate its error ε_j , $\varepsilon_j = \sum_{i=1}^L w_{t,i} h_j(x_i) \text{sgn}(y_i)$. Choose a weak classifier h_t with the lowest error ε_t from all these weak classifiers, then calculate coefficient

$$\alpha_t^{(0)} = (\ln((1-\varepsilon_t)/\varepsilon_t))/2.$$

c. Get a strong classifier function

$$f_k(x) = f_{k-1}(x) + \sum_{i=1}^{t-1} \alpha_i h_i(x) + \alpha_t^{(0)} h_t(x)$$

The corresponding strong classifier is:

$$H_k(x) = \begin{cases} 1 & f_k(x) \geq 0, \\ -1 & \text{otherwise} \end{cases}$$

d. Using the thresholds to test on positive samples and to make $f_k(x)$ achieve the default requirements $D_k \geq d \times D_{k-1}$.

e. Use $f_k(x)$ to test on negative samples, if $F_t \leq f \times F_{t-1}$ then exit the iteration.

f. Update the weights $w_{t+1,i} = w_{t,i} e^{-\alpha_t h_i}$, where $\alpha_t = (\ln((1-\varepsilon_t)/\varepsilon_t))/2$.

5) Get a strong classifier function

$$f_k(x) = f_{k-1}(x) + \sum_{i=1}^T \alpha_i h_i(x)$$

the corresponding strong classifier is:

$$H_k(x) = \begin{cases} 1 & f_k(x) \geq \beta_k, \\ -1 & \text{otherwise} \end{cases}$$

where $\beta_k = \min_{i=1,m} (f_k(x_i))$.

5. Experimental results

5.1 Experiments on MIT-CBCL data set

The publicly available MIT-CBCL face database is used to evaluate the performance of the proposed face detection system. The original MIT-CBCL training set contains 2,429 face images and 4,548 non-face images in 19×19 pixels grayscale PGM format. The training faces are only roughly aligned, i.e., they were cropped manually around each face just above the eyebrows and about half-way between the mouth and the chin. The data set will serve our purpose of comparing our detection system with their original system, which we shall train using the same training set. Some samples are shown as Figure 5.

The training data set we used is the subset of MIT-CBCL and consist of 1,429 face images and 3,548 non-face images. The test set consists of 1,000 face images and 1,000 non-face images left. The computer we used is with P4/2.4GHz CPU, 1 GB memory. The results are shown as Table 1.

According to Table 1, the Detection rate and False positive under Rank-AdaBoost or Dual-AdaBoost are similar with Init-AdaBoost, but the training time is obviously different. The training time used in Dual-AdaBoost is only 1/50 of that of Init-AdaBoost. The detection time also fall about half because of less classifiers used. Moreover, the robustness and the ability of generalization become better and better.

5.2 Experiments on a Real-World Test Set

5.2.1 Training data sets selection

Face samples must be selected carefully with variability in face orientation (up-right, rotated), pose (frontal, profile), facial expression, occlusion, and lighting conditions. Moreover some unimportant features of the face should be removed, such as hair and or so.

Non-face samples could be selected randomly. Any sub-window of an image containing no face can be used as a non-face sample. Almost arbitrary large training set can be easily constructed using these non-face samples.

In our experiment, the training data were collected from various sources. Face images were taken from the MIT-CBCL face training dataset, FERET face dataset, NJUST603 and web. The dataset contains face images of variable quality, different facial expressions and taken under wide range of lightning conditions. The dataset contained 8145 face samples including rotated versions of some faces.

Non-face images were collected from the web and MIT-CBCL non-face dataset. Images of diverse scenes were included. The dataset contained images such as animals, plants, countryside, man-made objects, etc. Some non-face samples were selected by randomly picking sub-windows from hundreds of images that did not contain face. More than 2,180 thousand non-face samples were used. Hundreds of these non-face images is very similar to face. Each image of all samples was cropped into size of 19×19 . Figure 6 shows some face samples. Figure 7 shows some non-face samples.

In the experiment, 9954 total Haar-like features were selected and used as weak classifiers. The smallest rectangle filter was defined as 4×4 , the biggest was 16×16 on the window of 19×19 image.

5.2.2 Experimental results

Given that non-face samples discarded on each layer was more than 40%, at the same time the face samples detection rate was more than 99.99% . We get a cascaded detector using the proposed improved AdaBoost and inheriting cascade model. The face cascaded detector using Rank-AdaBoost has 35 layers of classifiers. The face cascaded detector using Dual-AdaBoost has 21 layers of classifiers.

We tested our system on the MIT+CMU frontal face test set(Rowley et al,1998, pp.22–38). MIT+CMU dataset consists of 130 images with 507 total frontal faces. Every image was resized by 0.85 each iteration during test. The the performance of our detection system are shown in Table 2.

Experimental results on MIT+CMU face set shows that our method provides higher classification performance than Viola and Jones' method both on training speed and detection accuracy.

6. Conclusions and Future Work

In this paper, we presented a speed-up technique to train a face detector using AdaBoost by improveing the threshold searching method and new inheriting cascaded frame. Our proposed face detection system incorporating the technique reduces the number of subwindows that need preprocessing and verification. The proposed system is much faster than the original AdaBoost-based detection systems in training speed and is also higher in testing accuracy. It is suitable for realtime applications. Further, the system performs well for frontal faces in gray scale images with variation in scale and position.

A larger training set would be essential for the detector to be of practical use. In particular, the number of non-face images would have to be drastically increased in order to decrease false positives. Moreover, as mentioned earlier, using a larger number of Haar-like features would also improve the accuracy. Implementing and improving the cascade is required in order to achieve the ultimate aim of our work, i.e., to improve the accuracy of the detector while maintaining real-time detection speed.

References

- H.Rowley, S.Balujaand T.Kanade.(1998). Neural Network-based Face Detection. *IEEE Pattern Analysis and Machine Intelligence*, 20, 22–38.
- H.Rowley, S.Baluja and T.Kanade. (1998). Rotation Invariant Neural Network-based Face Detection. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition [C]*, Australia, 38-44.
- Liang Luhong, Ai Hai-zou and Xu Guang-you. (2002). A Survey of Human Face Detection. *Chinese Journal of Computer*, 25, 449~458.
- P. Viola and M. Jones. (2001). Rapid object detection using a boosted cascade of simple features [A]. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition [C]*, USA, 511~518.
- P. Viola and M.Jones. (2004). Robust real-time face detection. *International journal of Computer Vision*, 57, 137~154
- P. Viola and M. Jones. (2003). Fast Multi-view Face Detection. Shown as a demo at the IEEE Conference on *Computer*

Vision and Pattern Recognition [C], USA.

Schneiderman, H. and Kanade, T. (2000). A Statistical Method for 3D Object Detection Applied to Faces and Cars. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, USA, 746-751.

Table 1. Comparison of the training and detection

	Init-AdaBoost	Rank-AdaBoost	Dual-AdaBoost
Time for getting a Simple classifier (s)	0.1192	0.0071	0.0023
Sum of weak classifier	96	98	56
Detection rate (%)	97%	96.4%	98.6%
False positive	1	2	0
Detection time (s)	1.766	1.781	0.953

Table 2. Results on the MIT+CMU test set

	Viola's Detector			Rank-AdaBoost			Dual-AdaBoost		
False Positive	50	78	167	48	82	169	51	69	138
Detection Rate(%)	91.4	92.1	93.9	91.2	92.3	94.0	91.8	92.6	94.2

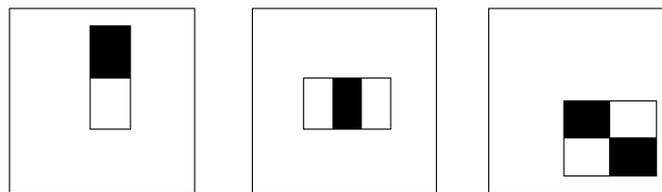


Figure 1. Examples of the Viola and Jones features

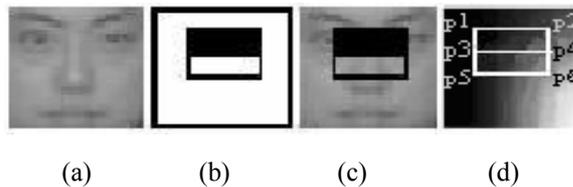


Figure 2. Features extraction using integral image

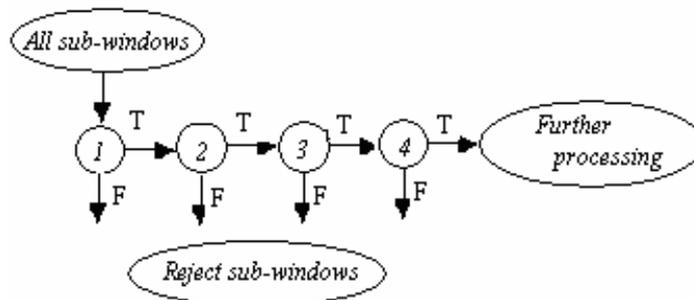


Figure 3. Schematic depiction of a cascaded detector

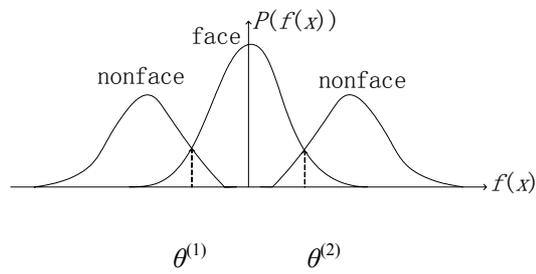


Figure 4. Distribution of typical local features



(a) Some face examples



(b) Some nonface examples

Figure 5. Some training examples



Figure 6. Some faces from the training set



Figure 7. Some non-faces from the training set