Locally Assembled Binary (LAB) Feature with Feature-centric Cascade for Fast and Accurate Face Detection

Shengye Yan[1, 2, 3], Shiguang Shan[1, 2], Xilin Chen[1, 2], Wen Gao[4, 2]

[1]Key Lab of Intelligent Information Processing, Chinese Academy of Sciences (CAS) , Beijing, China
[2]Digital Media Research Center, Institute of Computing Technology, CAS, Beijing, 100190, China
[3]Graduate School of the Chinese Academy of Sciences, Beijing, 100039, China
[4]School of EE&CS, Peking University, Beijing, 100871, China
{syyan, sgshan, xlchen, wgao}@jdl.ac.cn

Abstract

In this paper, we describe a novel type of feature for fast and accurate face detection. The feature is called Locally Assembled Binary (LAB) Haar feature. LAB feature is basically inspired by the success of Haar feature and Local Binary Pattern (LBP) for face detection, but it is far beyond a simple combination. In our method, Haar features are modified to keep only the ordinal relationship (named by binary Haar feature) rather than the difference between the accumulated intensities. Several neighboring binary Haar features are then assembled to capture their co-occurrence with similar idea to LBP. We show that the feature is more efficient than Haar feature and LBP both in discriminating power and computational cost. Furthermore, a novel efficient detection method called feature-centric cascade is proposed to build an efficient detector, which is developed from the feature-centric method. Experimental results on the CMU+MIT frontal face test set and CMU profile test set show that the proposed method can achieve very good results and amazing detection speed.

1. Introduction

In the past decade, we have witnessed the vigorous development on face detection techniques. More and more fast and accurate face detection systems are developed by different research or commercial organizations for various practical applications such as visual surveillance, robotics, image retrieval, and intelligent human computer interfaces.

Before introducing the developments and problems in face detection we first give the concept of face detection which is quoted from reference [1]: Given an arbitrary image or an image sequence, the goal of face detection is to determine whether or not there are any faces in the image, and if present, return their image locations and extents.

In general scenario faces in images change with different lighting conditions, persons, poses, expressions etc. All of these factors make face detection challenging. To cope with the changes, it is well accepted by researchers [2, 3, 4] that proper features and effective learning methods should be designed or adopted to model 'faces'. More generally speaking, this is actually the key problem for all pattern classification problems.

Many different types of features are proposed based on various physical properties of human faces. These features include intensity, color, texture, edge and figure [2, 3, 4, 5, 6]. Some features show good performance under specific conditions, but have limitations for general cases. For example, skin color can be an effective feature to segment face regions. However, it needs heuristic post-processing to extract faces from the segmentation results. Also, skin color is sensitive to illumination changes and can only be applied to color images. Other features, such as wavelet [7], may handle moderate illumination variation. However, when considering classification accuracy and computation cost together, the methods based intensity usually achieve the best performance in some special application environments [2, 7, 8].

As to the classifiers, typical classifiers applied to face detection include neural networks [8, 9], Bayesian classifier [7], Support Vector Machine (SVM) [2], and SNoW [10].

The recent milestone in face detection research is Viola and Jones's work [11]. In their work, a frontal face detection system was developed which achieved excellent accuracy and nearly real-time speed. Haar features and Adaboost are explored to build a cascaded detector. After this seminal work, many improved versions were proposed. Mostly of them focused on alternatives to AdaBoost [12, 13, 14, 15, 16, 17, 18 and 19], Haar features [19, 20], coarse-to-fine architecture [21, 22, 23, 24] and the optimization tuning of the cascade architecture [17, 18, 26, 27, 28].

Indeed, the aforementioned technologies, especially Viola and Jones' work, have greatly advanced face detection. However, it is of course not the end of face detection research in terms of both classification accuracy and detection speed, especially under complicated situations.

To further improve the efficiency of face detection system, in this paper, we propose a novel face detection method. The first key contribution of our method is a novel type of feature. We call the feature Locally Assembled Binary (LAB) feature. LAB feature is inspired by the success of Haar feature and Local Binary Pattern (LBP) for face detection, but it is far beyond their simple combination. In our method, Haar features are modified to keep only the *ordinal* relationship (named by *binary* Haar feature) rather than the difference between the accumulated intensities. Then, several neighboring binary Haar features are assembled together to capture their co-occurrence with a similar idea to LBP.

To learn an efficient face detector, a feature-centric cascade is further proposed, which bases on the original feature-centric detection method. Feature-centric cascade introduces cascade

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

idea into feature-centric method. It speeds up frontal final face detection largely, and even more for multi-view face detection. This is the second contribution of the paper.

To evaluate the classification accuracy and computation cost of the proposed LAB feature and feature-centric cascade, we conduct experiments on a fairly large frontal face dataset (includes 230,000 frontal face samples). The face images in the dataset cover various sources of variations. The final detector is evaluated on CMU+MIT frontal face test set and shows better performance compared with the known best results. We also conduct multi-view face detection experiments to further investigate the proposed method, in which the proposed method also shows good performances in both accuracy and speed.

The rest of the paper is organized as follows: we first introduce the proposed LAB feature in Section 2. Feature-centric cascade is described in Section 3. The next section presents the experimental results on both frontal face detection and multi-view face detection. Conclusions and future works are given in Section 5.

2. Locally Assembled Binary (LAB) features

In this section, we describe the proposed LAB features in three hierarchical aspects: the binary Haar feature; the assembled binary Haar feature, and LAB feature.

2.1. Binary Haar feature

For the purpose of clarity, we firstly review Haar feature and analyze the disadvantages for computational cost, and then the binary Haar feature is presented.

A Haar feature is a difference between the accumulated intensities of several adjacent rectangle areas. The classical layouts of the rectangles are illustrated in figure 1. The feature value is the difference between the filled rectangles and the unfilled rectangles. More generally, the layouts of the rectangles can vary arbitrarily. The accumulated intensities of the rectangle region can be computed efficiently by an aided image called integral image. Refer to [11] for details. The calculation of Haar features includes additions or subtractions of the accumulated intensities of the involved rectangles. For example, a 2-rectangle Haar feature as shown in Figure 1(a) and (b) can be calculated as:

$$f_j(x) = (s_1)_j - (s_2)_j, \qquad (1)$$

where $(s_1)_j$ and $(s_2)_j$ denote the intensity sum of the filled and unfilled rectangle of Haar feature *j*, *x* is the input image.

In practice, in order to extract features robust to lighting variations, lighting correction are generally used on the candidate image windows before feature extraction. Frequently used lighting correction methods include variance normalization, histogram equalization, and linear lighting correction etc. Though these lighting correction operations seem simple, the detection may become time-consuming because of their application to each candidate window in the input image. For instance, in Viola and Jones' work [11], for each candidate window of the input image, variance normalization is conducted. Haar feature is then calculated on variance normalized window.

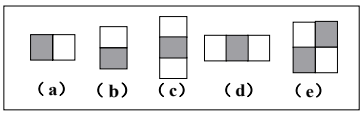

Figure 1: Haar feature.

What's more, the above lighting correcting procedure results in another problem: for the same Haar feature belonging to different candidate windows, it has to be re-calculated more than more times because of different lighting correction parameters for different windows. The re-calculation results in multiple feature evaluations and largely increases the computational cost. For instance, Haar feature with lighting correction is calculated by:

$$f_j'(x) = \frac{(s_1)_j - (s_2)_j}{\sigma_x}, \qquad (2)$$

where σ is the variance of some candidate window *x*. Because σs are different for different candidate windows, the same Haar feature is re-calculated for each different candidate windows which contains it, as illustrated in figure 2. Evidently, this leads to multiple floating point divisions.

To overcome the aforementioned problems, binary Haar feature is proposed, which keeps only the ordinal relationship in Haar feature computation:

$$b_j(x) = \begin{cases} 1 & ((S_1)_j - (S_2)_j > 0) \\ 0 & otherwise \end{cases}. \qquad (3)$$

In other words, the proposed binary Haar feature keeps only the sign information of Haar feature, while the absolute difference is discarded. By this binary operation, the feature becomes more robust to global lighting changes. Therefore, lighting correction specific for each candidate window is avoided by this light-robust feature, which decreases the computation cost. Furthermore, the above-mentioned feature re-calculation problem is avoided at the same time, and which facilitates the use of feature-centric strategy as described in section 3.1 and 3.2. As a result, the face detection speed is improved.

2.2. Assembling binary Haar features

In spite of its computational merits, we found that the discriminating power of a single binary Haar feature might be too weak to construct a robust classifier. To improve the discriminative power of the binary Haar feature, we propose to assembling multiple binary Haar features together and using their co-occurrence as a new kind of feature. The feature is called Assembled Binary Haar (ABH) feature. Figure 3 shows an example of the ABH feature. In the figure, the ABH feature integrates three binary Haar features. When the three binary Haar feature values are 1, 1 and 0, the ABH feature is calculated by:

$$a(b_1, b_2, b_3) = (110)_2 = 6, \qquad (4)$$

where *a* is the ABH feature calculation function from three binary Haar features b_1, b_2, and b_3, $(.)_2$ is the operation from a binary code to a decimal index. The feature value specifies an

index for 2^F different combinations, where F is the number of combined binary features.

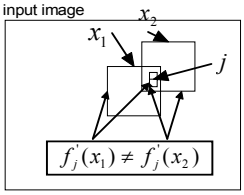

Figure 2: Re-calculation of Haar in different candidate windows.

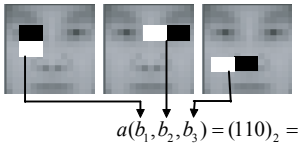

Figure 3: Assembling three binary Haar feature.

2.3. Locally Assembling Binary (LAB) features

The number of ABH feature is huge. To enumerate all of them, there are several free parameters, such as the number of binary Haar feature to be assembled, the size of each binary feature, and the position of each binary Haar feature. To learn from this large feature pool is intractable. Fortunately, we find a reduced set which is very good for face detection. the feature in the reduced set is called Locally Assembled Binary Haar feature. To be simplicity, it is called LAB feature hereinafter.

Among the assembled binary Haar features, LAB features are those ones that only combine 8 locally adjacent 2-rectangle binary Haar features with the same size and they share a common centre rectangle. The 8 binary Haar features used for assembling a LAB feature are shown in figure 4. Figure 5 gives two examples of LAB feature. In the figure, two different LAB features are showed. The centric black rectangle is shared by 8 neighboring binary Haar features. All nine rectangles are of the same size.

Formally, a LAB feature can be denoted by a 4-tuple, *l(x,y,w,h)*, where *x* and *y* denote the X-coordinate and Y-coordinate of the left top position of the feature in the image, *(w, h)* are the width and height of the rectangles.

LAB feature inherits all the advantages of binary Haar feature and is very discriminative. Also the number of it is small. LAB feature captures the local intensity structure of the image. Computation of a LAB feature needs to calculate 8 2-rectangle Haar features. The computation cost increases comparing with one Haar feature. But it possesses more discriminative power and do not need special light correction in detection process. These advantages in total reduce computation cost of the face detection process in our proposed method (section 3).

LAB feature is somewhat similar to locally binary pattern (LBP), which have been proved to be effective in texture analysis [29]. As one can see, LBP is the special case of LAB feature with one pixel size.

Similar to LBP, LAB feature value are lying in {0, …255}. Each value corresponds to a specific local structure.

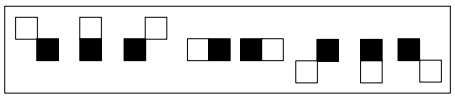

Figure 4: 8 Binary Haar features in a LAB feature.

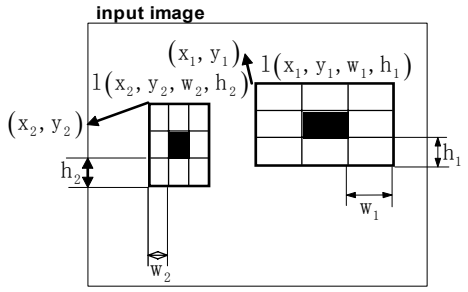

Figure 5: Two LAB features.

3. Face detection using LAB features

Cascade structure is also used in the proposed detection method. The whole cascade structure of the proposed face detector is shown in figure 6. It can be divided into two obvious parts. The first part is some sub-classifiers, which in total is called feature-centric cascade. They are run according to the proposed method in section 3.2. The second part is the other sub-classifiers, called window-centric cascade. They are run in a window-centric way similar to that of Viola and Jones' work.

This section is organized as following: Firstly, in section 3.1, two detection methods, feature-centric and window-centric, named by H. Schneiderman in [23] are introduced. Then, in section 3.2, based on the thorough analysis of theses two detection method, the feature-centric cascade method is proposed to build more efficient face detector. In section 3.3, the learning of window-centric cascade is described. Finally in section 3.4, the proposed detection method for one view is adapted to multi-view.

3.1. Feature-centric detection method

Before introducing feature-centric and window-centric method in detail, let us firstly review the total process of face detection in a high level. To find a face in the image, we need to do "exhaustive-search" in the image. This involves building a classifier that distinguishes between the object and "non-object" (any other scenery) while only having to tolerate limited variation in object location and size. The methods find the object by scanning this classifier over an exhaustive range of possible locations and scales in an image. Figure 7 illustrates this process, where the classifier calculates all possible "windows" in the image as shown by the rectangles.

Most cascades, such as the Viola and Jones [11], use "window-centric" method. These approaches compute lighting correction and feature calculation separately for each window. The scan of each possible window of the classifier means that each feature is also calculated in each position of the image. This means that the features containing in some window are calculated by other windows' classification, but it have not used for classification by this window's classifier. Feature-centric method aims to use more of the calculated

feature for each window.

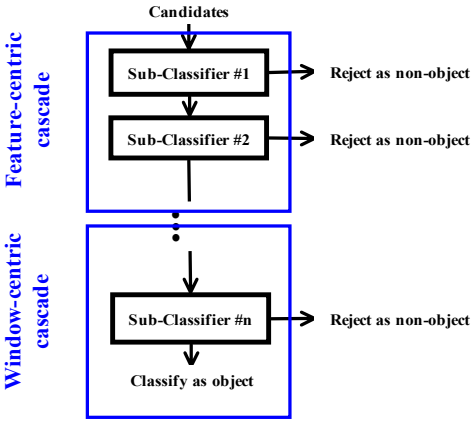

Figure 6: Overall structure of the proposed detector.

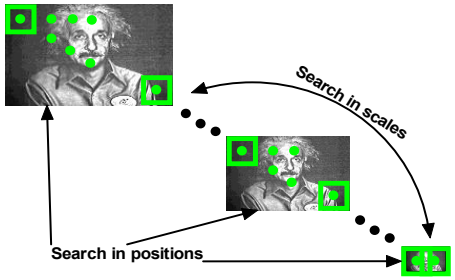

Figure 7: Exhaustive search in face detection.

To understand the window-centric and feature-centric detection method better, an example is given. The setup in the example is the same as it is in all our experiments. For window-centric method, Let us suppose that the classifier contains only one LAB feature. The feature is as shown in the rectangle in figure 8(a). While detection, each window in the image is classified, so this feature belonging to the classifier is also calculated at each position of the image. The calculation of the feature at each position means that the detection process produces a byproduct, which is the feature value image. The feature value image is shown in figure 8(b). For each window, in the case of this example, only the one feature is used for classifying it, the other features contained in it calculated by the other neighboring window' classification are not used. This is wasteful and less computation efficient, so feature-centric method is proposed to improve the utilization ratio of the calculated features.

In feature-centric method, firstly, feature value image (the middle image shown in figure 9,) is computed by scanning the upper feature in each position of the image. It is just the same one as in figure 8(b).Then the 'feature-centric' classifier is run on the feature value image and needs no feature calculation operation. Figure 9 illustrates this procedure. As to learning, the 'feature-centric' classifier is learned from all the features belonging to the window. In fact, the features are of the same size because they are collected by shift one specific feature on the image. Of course, any size can be used to build the 'feature-centric' classifier. But it is better to pick out the most effective one. In this paper we use a greedy search to find the best size, which is 3*3 in our experiments.

Utilizing all the calculated features in the window also incurs more classification operations because each feature is used to build a classifier and the classifier is run when classifying. But the extra classification operations are deserved because much more discriminating power is brought, especially when the classification operation is very simple and effective. In theory, any learning algorithm can be used to build a classifier for window-centric and feature-centric method. But considering the simplicity and effectiveness of classification operation, we use RealBoost learning [30] algorithm to learn a linear classification function, which can be expressed as:

$$c(x) = \sum_{i=1}^{T} h(l_i(x)), \qquad (5)$$

where c is the classification function, x is the sample window, h is the weak classifier function, l_i is the feature calculation function of feature i, T is the total feature number. For RealBoost, the classification operation h includes one look-up-table for feature value, one look-up-table for confidence and one addition. In figure 8 and figure 9, the linear classifier of window-centric and feature-centric method are also shown respectively. In figure 9, for feature-centric detection method, the classifier contains all the features containing in the window. The total feature number is denoted by N in the classification function.

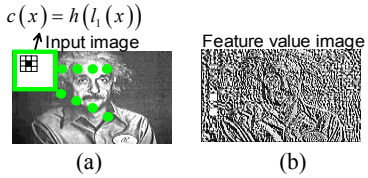

Figure 8: Windows-centric method with the classifier only contains one feature and the feature value image.

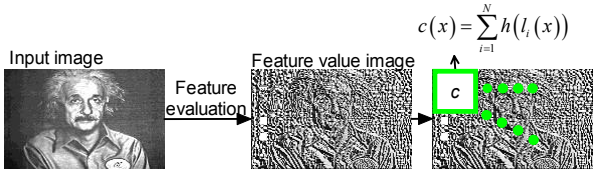

Figure 9: Feature-centric method with one size of LAB features.

3.2. Feature-centric cascade

In this section, we modify the 'feature-centric' classifier to a cascade for the sake of computational efficiency. In feature-centric method, all features containing in the window are used to construct one whole classifier. But 'feature-centric' classifier is frequently fairly strong. Scanning it as a whole at each position of the input image is not computationally smart. To promote the computation efficiency, it is better to further divide it into a cascade. The cascade learned from a 'feature-centric' classifier is called feature-centric cascade. Of course, it is run in a feature-centric way.

Obviously feature-centric cascade reduce computation cost.

An example and its computation cost analysis are given here. The setup of it is as the example in section 3.1.

Assuming the classification window size is 24*24, the feature is 3*3 LAB feature, so there are 256 features in a window. Because the other processes are of the same for feature-centric method and feature-centric cascade, so the numbers of classification operation number averaged on each window of these two methods represent their difference in computation cost. For feature-centric method, all 256 classification operations incurred by these 256 features are operated on each candidate window. So the mean classification operations for each window are 256 times. But for a feature-centric cascade, because some windows are rejected gradually with stage increasing, the mean classification operations for each candidate window must be less than 256 times. In our experiments, the classification operation number is less than 15 for frontal face detection.

The process of building a feature-centric cascade from a 'feature-centric' classifier is similar to Viola and Jones' work [11]. The features of the 'feature-centric' classifier and the feature-centric cascade are illustrated in figure 10. In the figure, l_i is the ith LAB feature picked out by RealBoost, N is the total feature number of a feature-centric classifier. The numbers in the arrowed arcs denote the stage number. The features covered by the arrow arcs are features belonging to the corresponding stages.

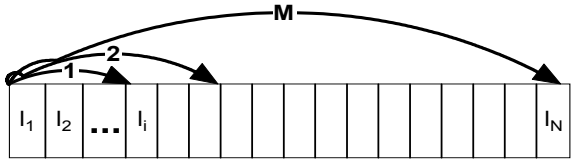

Figure 10: Feature-centric cascade.

3.3. Learning a further window-centric cascade

After learning the feature-centric cascade which rejects most simple non-faces efficiently, a window-centric cascade is learned based all sizes of LAB features to further reject those difficult non-faces. The learning procedure is similar to that of feature-centric cascade except that all sizes of LAB features are used. Of course, window-centric cascade is run in a window-centric way.

3.4. Multi-view face detection

Till now, we have presented the method to build a detector for one view of faces. In this section we extend the method to multi-view face detection. To construct a multi-view face detector, we first divide all faces into 5 categories according to left-right rotate off plane, and then continue to split each category into 3 views, each of which takes charge of 30° rotation in plane. Besides, each view covers [-30°, +30°] up-down rotation off plane for robustness. The 15 different views are illustrated in figure 11.

We build a feature-centric cascade and a window-centric cascade for each view. For detection, the procedure is illustrated in figure 12. Given the input image, we first compute the feature value image. Then for each view, feature-centric cascade is firstly run based on the calculated feature image, and then the window-centric cascade is run on the raw image.

Note that for multi-view face detection the feature value image is shared by all the feature-centric cascade of 15 views. This speeds up the detector largely.

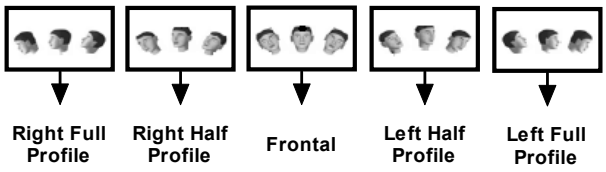

Figure 11: Multi-view face categories, each rectangle include 3 rotations in plane.

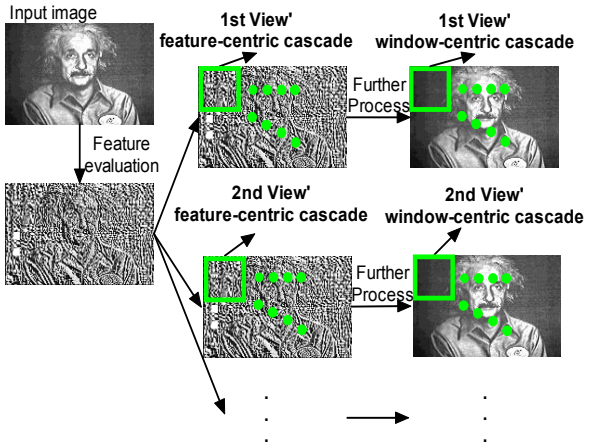

Figure 12: Feature-centric cascades for multi-view detection.

4. Experiments

In this section, we evaluate LAB features and the proposed detection method in frontal face and multi-view face detection. Section 4.1 gives the experimental setup firstly. Then in section 4.2, a quantitative analysis of LAB feature is given. Section 4.3 and section 4.4 report the performance of the proposed method on frontal and multi-view face database respectively.

4.1. Experimental setup

For frontal face detection, 23,608 frontal face samples are collected from various sources, such as WEB, FERET, and BioID. Most faces in the sample set have the variation of up-down off plane rotation within range of [-30°, 30°]. Totally 236,080 24×24 grayscale face samples are generated from the original 23,608 face images with manually labeled eyes by following transformation: mirroring, in plane rotation of -12°,-6°, 0°, and 6° 12°.

For multi-view face detection, the left full profile faces and the left half profile faces are mostly collected from 700 video clips captured by us. The scene of these video clips is in a room with normal lighting conditions. Each video clip contains one person. The faces in these video clips are about 100*100 in size. Totally 24,000 left full profile faces and

60,000 left half profile faces are collected. . By rotation in plane of -12°,-6°, 0°, and 6° 12°, the left full profile and the left half profile faces are 120,000 and 300,000 respectively. The face samples of other views can be generated by horizontal flip and rotation in plane.

As for the negative samples, 30,000 images without faces are collected for generating non-faces.

From the description above, one can see our training face set is quite large. We use Matrix-Structural Learning (MSL) [28] to learn from the training sets, which is a cascade learning method to deal with enormous training set. In cascade learning, Minimum detection rate and maximum false alarm rate of feature-centric and window-centric cascade are both set to 0.9999 and 0.4 respectively. The non-face samples used to train a feature-centric classifier are 60,000. The training non-face samples for each window-centric stage are fixed to 10,000. For the positive bootstrap in MSL, the starting face sample set size is 2,000. At each positive bootstrap, maximally 500 new samples are added.

To detect faces with various scales, test images are down-sampled with a coefficient of 0.8. In the later sections, if there is no specific mention, the experiments are conducted on a common PC with a 3.20GHz Pentium IV processor.

4.2. Evaluation of LAB feature on efficiency on frontal face samples

To build a feature-centric cascade, firstly, we search the most efficient size. The classifiers learned from different sizes of LAB features are investigated according to their classification accuracies. 3*3 is picked out as the most effective size.

Secondly, we conduct experimental comparisons with Haar features. For 3*3 LAB features, when sample size is 24*24, there are totally 256 features. In contrast, Haar feature set consists of quite a larger number of features, which is 31,728 to be exact. The feature types are as shown in figure 1. The bin number for Haar feature is set to 40 empirically.

The feature numbers, which mainly determine the computational cost, of a set of classifiers learned from 3*3 LAB features and Haar features are presented in table 1. The classifiers are learned by adjusting the target false alarm rate with the detection rate fixed to 0.9999. In table 1, 'FAR' denotes the false alarm rate. From table 1 one can see that feature numbers of LAB classifiers is always less than the corresponding ones of Haar classifiers by a half despite Haar features used to learn these classifiers are much more than 3*3 LAB features.

FAR	0.4	0.1	0.05	0.01	0.005	0.001	0
Haar	13	34	48	68	72	80	89
LAB	7	18	22	30	31	35	39

Table 1: Feature numbers of the classifiers learned from Haar and 3*3 LAB features at different false alarm rates.

4.3. Experiment on frontal face detection

A frontal face detector is trained based on the proposed method. The detection speed is about 30ms.for a 320*240 image. The face detector processes faces from 24*24 to 240*240.

In fact, the computation speed is about 20 times faster than Haar features at the same accuracy. Here, we also try to analyze the total operations in both detectors. As shown in Table 1, the LAB feature number in a feature-centric cascade is nearly a half of the Haar feature number at the same accuracy. Exactly, the LAB cascade processes about 15 LAB features for each window averagely, which includes 15 look-up-tables and 15 additions (for computing confidence). Including the feature evaluations (about 17 additions) in advance for each window, totally, LAB cascade needs 32 additions and 15 look-up-tables for each window. In contrast, Haar cascade processes nearly 30 Haar features, which include 30 Haar feature evaluations (each feature evaluation takes at least 7 additions using integral image), 30 divisions (for variance normalization), 30 look-up-tables and 30 additions (for computing confidence). So, totally Haar cascade needs at least 240 additions, 30 divisions and 30 look-up-tables. Evidently, the proposed method is overwhelming in speed.

To get a sense of the classification accuracy, the face detector is tested on CMU+MIT dataset comprising 130 images containing 507 faces. The ROC curves are given in figure 13. Obviously, our method gets the best results.

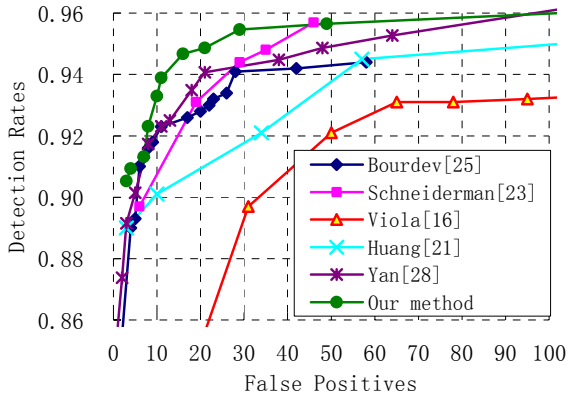

Figure 13: ROC curves on CMU+MIT frontal face set.

4.4. Experiment on multi-view face detection

We also trained a multi-view detector to investigate the proposed method. The multi-view face detector requires 80 ms on a 320x240 image. The detection processes faces from 24*24 to 240*240. It is reasonable that the multi-view face detector' detection time does not increase linearly with multiple classifiers because the preceding procedures (integral image computation, feature value image computation etc) are shared by all the feature-centric cascades of 15 categories.

To get a sense of the classification accuracy, the multi-view face detector is tested on CMU profile face test set comprising of 208 images containing 441 faces. Figure 14 shows the ROC curve. For comparison, the results of the previous methods are also listed. From the figure, one can see that our results are better than [23] but a little worse than [19]. Considering the profile face training samples in our experiment are lacking in varieties, the results are acceptable. Note that our method gets

a much higher detection speed than [23]. As to [19], the detection speed is close. In fact it is difficult to say which method is faster only from the results of the related literature because the programming is also very important. By the way, the frontal face detector and two half profile face detector without rotation in plane have been re-programmed to a 600M Hz DSP as a whole multi-view face detector. It surprisingly gets a detection speed of 30ms on a 320*240 image to run them simultaneously.

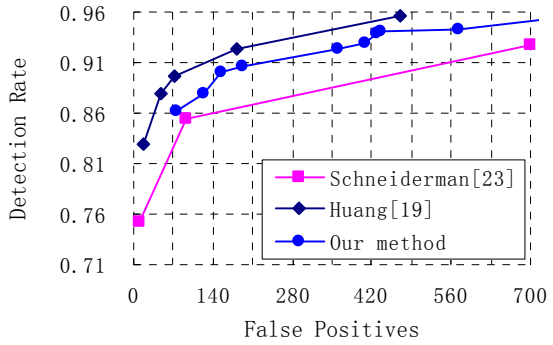

Figure 14: ROC curves on CMU profile face set.

5. Conclusion and future works

We propose a novel type of feature, called LAB feature, accompanying which a feature-centric cascade is also presented. In LAB feature, Haar feature is binarized and assembled together in a mode similar to LBP to model local texture modes. In some sense, LAB feature bridges the gap between Haar features and LBP operator. We show that LAB feature is not only robust to lighting variations (thus avoiding time-consuming lighting correction operator), but also very discriminative for face/non-face classification. Furthermore, it also facilitates using the feature-centric cascade to further improve the face detection speed. Extensive experiments verified the above observations.

There are still many open questions related to LAB features and feature-centric cascade, such as: Is there other more efficient features in assembled binary Haar features rather than LAB features? If any, how to find them out? If more sizes of features are used for feature-centric method, what will happen? Can the detection speed be promoted further? It is also interesting to apply LAB feature to other object detection, such as pedestrian, car.

Acknowledgements

This paper is partially supported by National Natural Science Foundation of China under contract No.60332010, No.60772071, and No.60673091; Hi-Tech Research and Development Program of China under contract No.2006AA01Z122 and No.2007AA01Z163; 100 Talents Program of CAS; and ISVISION Technology Co. Ltd.

References

[1] M.H. Yang, D. Kriegman, N. Ahuja. Detecting faces in images: a survey. IEEE TPAMI, 2002, 24 (1): 34~58.

[2] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. CVPR1997.

[3] R.L. Hsu, M. Abdel-Mottaleb, A.K. Jain. Face detection in color images. IEEE TPAMI, 2002, 24 (5): 696–706.

[4] C.A. Waring, X. Liu. Face detection using spectral histograms and svms. IEEE T.SMC, Part B, 2005, 35 (3): 467–476.

[5] K. C. Yow, R. Cipolla. A probabilistic framework for perceptual grouping of features for human face detection. AFGR 1996.

[6] B. Heiselet, T. Serre, M. Pontil, T. Poggio. Component-based face detection. CVPR 2001.

[7] H. Schneiderman, T. Kanade. A statistical method for 3d object detection applied to faces and cars. CVPR 2000.

[8] H. A. Rowley, S. Baluja, T. Kanade. Neural network-based face detection. IEEE TPAMI, 1998, 20 (1): 23–38.

[9] C. Garcia, M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. IEEE TPAMI, 2004, 26 (11):1408–1423.

[10] D. Roth, M. Yang, N. Ahuja. A snow based face detector. NIPS 2000.

[11] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. CVPR 2001.

[12] S.Z. Li, L. Zhu, Z. Q. Zhang, A. Blake, H. J. Zhang and H. Shum. Statistical learning of multi-view face detection. ECCV 2002.

[13] C. Liu and H.Y. Shum. Kullback-Leibler boosting. CVPR 2003.

[14] S.W. Lyu. Infomax Boosting. CVPR 2005.

[15] J. Sun, J. M. Rehg and A. F. Bobick. Automatic cascade training with perturbation bias. CVPR 2004.

[16] P. Viola. Fast and robust classification using asymmetric AdaBoost and a detector cascade. NIPS 2001.

[17] J. X. Wu, M. D. Mullin, and J. M. Rehg. Linear asymmetric classifier for cascade detectors. In ICML 2005.

[18] J. X. Wu, J. M. Rehg, M. D. Mullin. Learning a rare event detection cascade by direct feature selection. NIPS 2004.

[19] C. Huang, H.Z. Ai, Y. Li and S.H. Lao. High-performance rotation invariant multi-view face detection. IEEE TPAMI, 2007, 29(4): 671-686.

[20] R. Lienhart and J. Maydt. An extended set of Haar features for rapid object detection. ICIP 2002.

[21] C. Huang, H.Z. Ai, B. Wu and S. H. Lao. Boosting nested cascade detector for multi-view face detection. ICPR 2004.

[22] R. Lienhart, L. Liang, and A. Kuranov. A detector tree of boosted classifier for real time object detection and tracking. ICME 2003.

[23] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. CVPR 2004.

[24] R. Xiao, L. Zhu, and H. J. Zhang. Boosting chain learning for object detection. ICCV 2003.

[25] L.Bourdev and J. Brandt. Robust object detection via soft cascade. CVPR 2005.

[26] H.T. Luo. Optimization design of cascaded classifiers. CVPR 2005.

[27] J. Sochman, J. Matas. WaldBoost–Learning for time constrained sequential detection. CVPR 2005.

[28] S.Y. Yan, S.G. Shan, X.L. Chen, W. Gao and J. Chen. Matrix-Structural Learning (MSL) of cascaded classifier from enormous training set. CVPR 2007.

[29] T. Ojala, M. Pietikainen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. Pattern Recognition, 1996, 29(1):51–59.

[30] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rate predictions. Machine Learning, 1999, 37(3): 297-336.