Modification of the AdaBoost-based Detector for Partially Occluded Faces

Jie Chen[1] Shiguang Shan[2] Shengye Yang[2] Xilin Chen[2] Wen Gao[1,2]

[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China
[2] ICT-ISVISION Joint R&D Laboratory for Face Recognition, Institute of Computing Technology,
The Chinese Academy of Sciences, Beijing 100080, China
{chenjie, sgshan, syyan, xlchen, wgao}@jdl.ac.cn

Abstract

While face detection seems a solved problem under general conditions, most state-of-the-art systems degrade rapidly when faces are partially occluded by other objects. This paper presents a solution to detect partially occluded faces by reasonably modifying the AdaBoost-based face detector. Our basic idea is that the weak classifiers in the AdaBoost-based face detector, each corresponding to a Haar-like feature, are inherently a patch-based model. Therefore, one can divide the whole face region into multiple patches, and map those weak classifiers to the patches. The weak classifiers belonging to each patch are re-formed to be a new classifier to determine if it is a valid face patch—without occlusion. Finally, we combine all of the valid face patches by assigning the patches with different weights to make the final decision whether the input subwindow is a face. The experimental results show that the proposed method is promising for the detection of occluded faces.

1. Introduction

Over the past ten years, face detection has been thoroughly studied in computer vision research for its wide potential applications, such as video surveillance, human computer interface, face recognition, and face image database management etc. Face detection is to determine whether there are any faces within a given image, and return the location and extent of each face in the image if one or more faces are present [21]. Recently, the emphasis has been laid on data-driven learning-based techniques, such as [1, 4, 8, 9, 10, 12, 13, 16, 20]. All of these schemes can be found in the recent survey by Yang [21]. After the survey, one of the important progresses is the boosting-based method proposed by Viola [17] who uses the Haar features for the rapid object detection, and a lot of related works were then followed [5, 6, 7, 15, 18, 19]. Meanwhile, in [6], Lin *et al.* proposed a framework to detect the occluded faces based on the reinforcement training and cascading with evidence.

One of the challenges associated with face detection is Occlusion [21]—Faces may be partially occluded by other objects. Some examples are shown in Fig 1. It results in the presence or absence of some structural components. In general, the occlusions include occluding by other objects (such as the accessories or other faces), self-occlusion (for example, a profile face), local highlight or underexposure in the face images or poor image quality (such as being blurred very much). All of these cases decrease the systems' performance rapidly.

In this paper, we present a solution to detect partially occluded faces by reasonably modifying the AdaBoost-based face detector, which is easy to be implemented while it can work favorably.

The rest of this paper is organized as follows: In section 2, we detail the method about how to detect the occluded face in our case. Some experiments are carried out in Section 3. In Section 4, we give the conclusion.

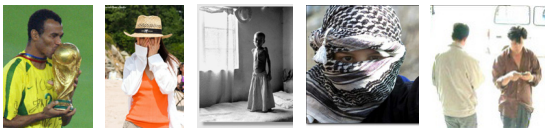


Fig 1. Some occluded face images.

2. Detecting the occluded faces

After training an AdaBoost-based face detector using the Haar Basis functions, the feature statistics demonstrates that most of the learned Haar features locate in the regions near the eyes. As shown in Fig. 2, if some patches, such as 4th, 5th, 6th and 7th, are occluded, it disables the features located in these regions. Consequently, the cascade detector will reject the input window in the first several layers. How to detect these occluded faces? We will detail it in this section.

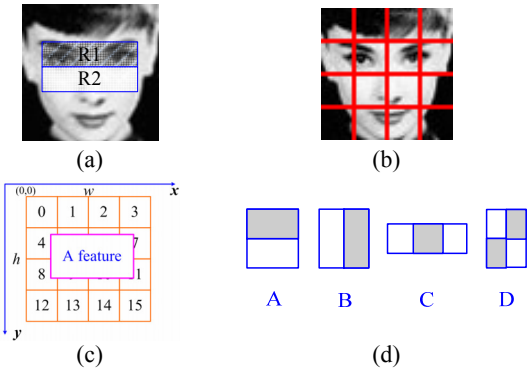


Fig. 2. Sample division and weak classifier mapping. (a) A typical Haar-like feature and a face sample; (b) Divide this sample into patches without overlapping. (c) Map a weak classifier to patches; (d) Some example rectangle features. The feature value is the difference between the sum of the pixels within the white rectangles and the sum of pixels in the grey rectangles; (A) and (B) are two-rectangle features; (C) is a three-rectangle feature, and (D) is a four-rectangle feature.

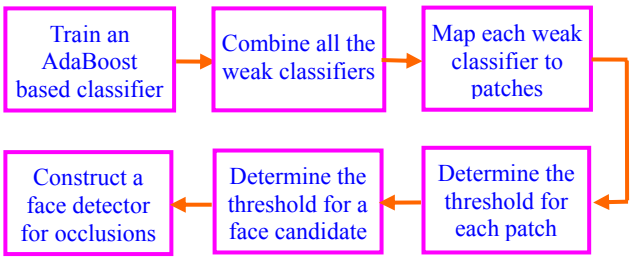


Fig.3. The flow chart of the proposed method.

The flow chart of our algorithm is illustrated in Fig.3. First of all, we train an AdaBoost-based face detector, and combine all the weak classifiers of each layer into a strong classifier only in one stage. Subsequently, we divide the training samples (positive and negative ones)

into patches and map each weak classifier to the patches as illustrated in Fig.2, followed by determining statistically the threshold for each patch to verify if it is a valid face patch. Finally, we decide the threshold for a face candidate by assigning the patches with the different weights to vote whether the input sub-window is a face.

2.1 Sample division and weak classifier mapping

If some face regions are present in an image, one might conclude that there is a face in the given image. Motivated by this intuition, we can divide the input sub-window into patches, and then we search the window to determine whether some possible patches appear.

In detail, we divide evenly the window into $m \times n$ patches $S = \{patch_0, patch_1, \cdots, patch_{mn-1}\}$ without overlapping as shown in Fig. 2 (b). In our case, the size of each sample is 20×20 pixels, so we let m=n=4 and each patch is a 5×5 block.

For the AdaBoost-based cascade classifier, each weak classifier corresponds to a rectangle feature as shown in Fig. 2 (a). In order to determine which patch is present in the input window, we map each weak classifier to patches. Having computed the output of each weak classifier, one can determine which patch appears. As illustrated in Fig.2 (c), the algorithm for mapping a weak classifier to patches is as follows:

Algorithm 1: Map a weak classifier to patches

Step 1: Compute the overlapped area between the rectangle feature and each patch;

Step 2: Find the patch $Patch_j$ $(j = 0, 1, \cdots mn-1)$ with the largest overlapped area $Patch_{MAX}$ for the feature;

Step 3: **For** i=0, 1, …, mn-1 :

If the overlapped area between $patch_i$ and feature is no less than $\xi \times Patch_{MAX}$, then the feature is high correlative to $patch_i$, otherwise, weak correlation.

End For

Because a rectangle feature might overlap with several patches as illustrated in Fig. 2(c), several overlapped areas might equal to $Patch_{MAX}$ and some overlapped areas might approximate $Patch_{MAX}$. Herein, we set a parameter ξ in step 3, which equals to 0.8 in our case.

For example, as shown in Fig 2 (c), the overlapped area ratios between the rectangle feature and 16 patches are {0, 0, 0, 0, 0.11, 0.67, 0.67, 0.22, 0.08, 0.58, 0.58, 0.19, 0, 0, 0, 0}. The overlapped area ratio for $patch_i$ is computed as:

$$ratio_i = \frac{area(overlapped\ between\ the\ feature\ and\ patch_i)}{area(patch_i)}. \quad (1)$$

The value of $Patch_{MAX}$ is 0.67, which corresponding to the patches 5 and 6. So we have $\xi \times Patch_{MAX} = 0.8 \times 0.67 = 0.54$. Consequently, the given feature is mapped to these four patches 5, 6, 9, 10.

Note that a rectangle feature usually consists of several sub-rectangles as shown in Fig 2 (d). When mapping it to patches, all of the sub-rectangles are seen as a whole one. For example, the feature in Fig 2 (c) is composed of two sub-regions R1 and R2 as shown in Fig 2 (a).

2.2 Threshold and weight for each patch

Having mapped each learned feature to patches, how to determine a patch is a valid patch for an input sub-window (Herein, a valid patch means that it is a valid face patch and is not occluded.)? It is described in the Algorithm 2.

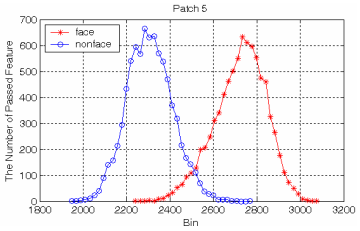


Fig.4. A histogram example for $patch_5$ to determine the threshold θ_5^p according to Minimum Error Rate Criterion, where the x-coordinate denotes the outputs of the all weak classifiers associated to $patch_5$, and y-coordinate denotes the number of the features with $sgn\left(h_t^{p_5}(\boldsymbol{x}) - \theta_t^f\right)$ equal to +1 for each bin.

Let $h_t^{p_i}$ ($i = 0, 1, \cdots, mn-1$, $t = 0, 1, \cdots, N_{p_i} - 1$) be the weak classifier associated to $patch_i$; N_{p_i} is the number of these weak classifiers associated to $patch_i$. By the step 1, we can calculate the outputs of the all $h_t^{p_i}$; and by step 2, we determine $N_{p_i}^w$ —the number of the weak classifiers associated to $patch_i$ and $sgn\left(h_t^{p_i}(\boldsymbol{x}) - \theta_t^f\right)$ equal to +1. After the histograms in step 3 are plotted, θ_t^p is decided according to the Minimum Error Rate Criterion. A histogram example for $patch_5$ to determine the threshold θ_5^p is illustrated in Fig.4. In our case, $N_p = N_n$ =15,000, and N_f =4,630. The x-axis denotes the bins while the y-axis denotes the numbers of the features with $sgn\left(h_t^{p_5}(\boldsymbol{x}) - \theta_t^f\right)$ equal to +1 for the $patch_5$ of faces and non-faces.

Algorithm 2: Determine the threshold for each patch

Input: N_p face samples; N_n non-face samples; N_f weak classifiers learned by the AdaBoost-based algorithm; the learned threshold $\boldsymbol{\theta}^f = (\theta_0^f, \theta_1^f, \cdots, \theta_{N_f-1}^f)$ for each weak classifier $h_t(\bullet)$, $t = 0, 1, \cdots, N_f - 1$;

Output: The threshold for each patch $\boldsymbol{\theta}^p = (\theta_0^p, \theta_1^p, \cdots, \theta_{mn-1}^p)$

Step 1: **For** each weak classifier $h_t(\bullet)$, **Do**:

Compute the outputs for all of the samples(faces and non-faces);

End For

Step 2: **For** the faces and non-face samples, respectively **Do**:

For each weak classifier $h_t(\bullet)$, **Do**:

Let $\boldsymbol{x}$ be an input sample;

Divide $\boldsymbol{x}$ into several patches;

Determine the patches to which the weak classifier $h_t(\bullet)$ is high correlative;

If ($sgn\left(h_t^{p_i}(\boldsymbol{x}) - \theta_t^f\right)$ equal to +1):

Count the feature to the corresponding patches.

End If

End For
End For

Step 3: Plot the histogram for the each patch and determine the threshold $\boldsymbol{\theta}^p$

The weights for each patch can be determined straightforward. For the $patch_i$, its weight ω_i can be calculated as

$$\omega_i = N_{p_i} / N_f \quad , \quad (i = 0, 1, \cdots mn-1). \tag{2}$$

2.3 Threshold for a face candidate

For an input sub-window $\boldsymbol{x}$, as discussed in section 2.2, $N_{p_i}^w$ is computed as:

$$N_{p_i}^w = \sum_{t=0}^{N_{p_i}-1} 0.5\left(1+\mathrm{sgn}\left(h_t^{p_i}(\boldsymbol{x}) - \theta_t^f\right)\right). \tag{3}$$

Whether the $patch_i$ is a valid patch is depended on:

$$H_i(\boldsymbol{x}) = 0.5(1+\mathrm{sgn}(N_{p_i}^w - \theta_i^p)) \quad (i = 0, 1, \cdots mn-1). \tag{4}$$

So, we have:

$$f(\boldsymbol{x}) = \sum_i \omega_i H_i(\boldsymbol{x}), \tag{5}$$

And the label of $\boldsymbol{x}$ is equal to:

$$label(\boldsymbol{x}) = \mathrm{sgn}(f(\boldsymbol{x}) - \theta_{fc}), \tag{6}$$

where θ_{fc} is the threshold for a face candidate. It is determined during the experiments which can be used to adjust the detection rates and false alarms.

2.4 Classification of an input window

The detailed algorithm about how to classify an input window is as follows:

Algorithm 3: Classification of an input window

Input: An input sub-window from an image
Output: The label of the input window

Step 1: Divide the input into patches as illustrated in Fig 2.
Step 2: Compute the outputs for all weak classifiers associated to each patch;
Step 3: Compute the $N_{p_i}^w$ according to Eq. (3)
Step 4: Compute the $H_i(\boldsymbol{x})$ according to Eq. (4)
Step 5: Compute the $f(\boldsymbol{x})$ according to Eq. (5).
Step 6: Label the input window according to Eq. (6).

3. Experiments

In this section, we train an AdaBoost-based classifier using the Haar-like features, and then combine all of the weak classifiers as discussed in section 2. The resulting system is tested on the MIT+CMU frontal and profile face test set and a test set from the internet which includes various occluded faces.

3.1 Train the AdaBoost based classifier

The basic method to make the face detector robust to affine transform is to enlarge the face set by rotating, translating and scaling [2, 3, 8, 11, 14, 15, 18]. All cropped samples are normalized to the size of 20×20 pixels, and then histogram equalization is performed.

The original database that we use consists of 6,000 faces which are collected from internet. By applying the preprocessing mentioned above, we get 15,000 face samples. The non-face class is initially represented by 15,000 non-face samples.

These samples are used to train an AdaBoost-based classifier. Each single classifier is trained using a bootstrapping approach similar to what described in [16] to increase the number of samples in the non-face set on a set of 16,536 images containing no faces. The resulting complete face detection cascade consists of 20 stages with N_f= 4,630 features. For the details of the AdaBoost based classifier, please refer to [17].

Fig. 5. Divide a sample into patches with overlapping.

Besides the classifier based on the non-overlapping pattern, we adopt a different division method to exploit the influence about dividing samples into sub-blocks in the different manner. As shown in Fig. 5, we partition a sample also into $m \times n$ patches. However, every two neighbor blocks overlap with each other by two rows or columns. Likewise, after training an AdaBoost-based cascade classifier, we map each weak classifier to patches as discussed for the non-overlapping case.

3.2. Test on the collected set

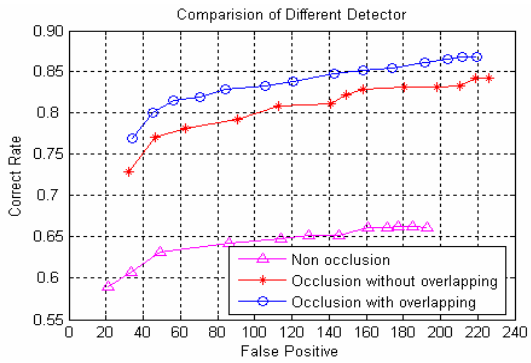
To test the robustness of the resulting systems on various occluded faces, we collect a test set from the web, which consists of 174 images showing 328 faces. The performances of these detectors on this set are compared in Fig. 6 (a). All of these three classifiers are trained on the same training set, where "*Non occlusion*" is the originally trained AdaBoost based detector; "*Occlusion without overlapping*" means that we combine all the weak classifiers by the proposed method, and samples are divided *without* overlapping; "*Occlusion with overlapping*" means that we combine all the weak classifiers by the proposed method and samples are divided *with* overlapping. During the detecting process, all of the input windows are not divided, or divided without overlapping or overlapping, correspondingly for each system.

From these Receiver Operating Characteristic (ROC) curves one can conclude that the detector "*Occlusion without overlapping*" outperforms significantly the originally trained AdaBoost-based classifier. The detector "*Occlusion with overlapping*" improves the detector performance further.

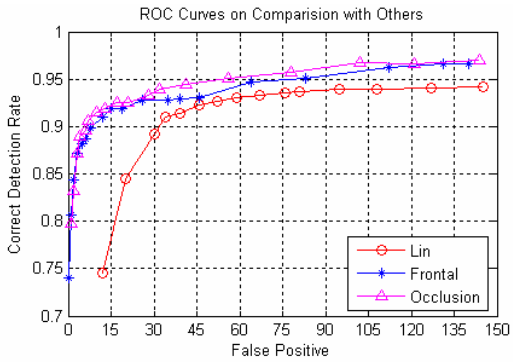
3.3. Test on the CMU face test set

The original AdaBoost-based detector and its derived detector "*Occlusion with overlapping*" are both evaluated on the MIT+CMU frontal face test set, which consists of 130 images showing 507 upright faces [12]. The performances of these two detectors on this set are compared in Fig. 6 (b).

From the ROC curves one can conclude that its derived detector "*Occlusion with overlapping*" still outperforms the original AdaBoost-based detector. However, in this test set, most of the faces being not occluded, the improvement of performance is not as remarkable as the collected set mentioned in section 3.2.



(a)



(c)

Fig.6. The experimental results comparison. (a) Results for the different methods on the collected test set; (b) Comparison with others on MIT+CMU frontal face test set.

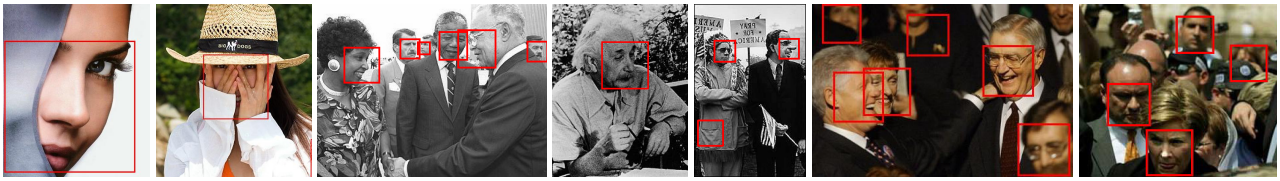


Fig.7. Outputs of our face detector. Note several false alarms are shown in the third row.

In Fig. 6 (b), we also address the performance comparison of the proposed method with the reported results in [6]. It can be found that our method compares favorably with Lin, especially for the low numbers of false alarms. However, different criteria (e.g. number of training examples involved and the number of scanned windows during detection etc.) can be used to favor one over another.

Fig. 7 shows some examples of the detected faces from the two testing sets: the CMU frontal and profile face test set and the collected test set from web. From these images, one can conclude that our detector displays robust generalization performance for various occlusions.

4. Conclusion

In this paper, we present a solution to detect partially occluded faces by reasonably modifying the AdaBoost-based face detector. The only inheritance from the trained cascade by AdaBoost is the weak classifiers and their corresponding thresholds. Neither the trained cascade classifier is used as a detector for occlusions in our case, nor the weight for each weak classifier trained by AdaBoost. This method can be easily implemented and work well. The future work is to regroup the weak classifiers in cascade to improve the detection speed.

Acknowledgements

This research is partially sponsored by Natural Science Foundation of China under contract No.60332010, “100 Talents Program” of CAS, the program for New Century Excellent Talents in University (NCET-04-0320).

Reference

[1] R. Féraud, O. Bernier, J. Viallet, and M. Collobert. A Fast and Accurate Face Detector Based on Neural Networks, in *PAMI* 2002, vol. 23, no. 1, pp. 42-53.

[2] C. Garcia and M. Delakis. Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection, in *PAMI* 2004, pp. 408-1423.

[3] B. Heisele, T. Poggio, and M. Pontil. Face Detection in Still Gray Images. *CBCL Paper #187*. MIT, Cambridge, MA, 2000.

[4] R. L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, Face detection in color images, in *PAMI* 2002, pp. 696–706,

[5] S. Li and Z. Zhang. Floatboost learning and statistical face detection, in *PAMI* 2004, pp.1112 – 1123,

[6] Y. Y. Lin, T.-L. Liu, and C.-S. Fuh. Fast Object Detection with Occlusions, in *ECCV* 2004, pp. 402–413.

[7] C. Liu, H. Y. Shum. Kullback-Leibler Boosting, in *CVPR* 2003.

[8] C. J. Liu. A Bayesian Discriminating Features Method for Face Detection, in *PAMI* 2003. pp. 725-740

[9] E. Osuna, R. Freund, and F. Girosi, Training support vector machines: An application to face detection, in *CVPR* 1997, pp. 130–136.

[10] C. P. Papageorgiou, M. Oren, and T. Poggio, A general framework for object detection, in *ICCV* 1998, pp.555–562.

[11] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection, In *CVPR* 2001.

[12] H. A. Rowley, S. Baluja, and T. Kanade. Neural Network-Based Face Detection, in *PAMI* 1998. pp. 23-38.

[13] H. Schneiderman and T. Kanade. A Statistical Method for 3D Object Detection Applied to Faces, in *CVPR*2000, pp. 746-751.

[14] H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection, In *CVPR* 2004., P29-36

[15] J. Sun, J. Rehg, and A. Bobick. Automatic cascade training with perturbation bias, In *CVPR* 2004, pp. 276 -283

[16] K. K. Sung, and T. Poggio. Example-Based Learning for View-Based Human Face Detection, in *PAMI* 1998.

[17] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features, in *CVPR* 2001.

[18] J. Wu, J. Rehg, and M. Mullin. Learning a rare event detection cascade by direct feature selection, In *NIPS*, 2003.

[19] R. Xiao, M. J. Li, H. J. Zhang. Robust Multipose Face Detection in Images, *IEEE Trans on Circuits and Systems for Video Technology*, pp. 31-41. 2004.

[20] M. H. Yang, D. Roth, and N. Ahuja. A SNoW-Based Face Detector, in *NIPS* 2000, MIT Press, pp. 855-861.

[21] M. H. Yang, D. Kriegman, and N. Ahuja. Detecting Faces in Images: A Survey, in *PAMI* 2002, pp. 34-58.