Investigating the boosting framework for face recognition

Bas Boom Robin van Rootseler Raymond Veldhuis
University of Twente
Fac. EEMCS, Signals and Systems Group
7522 NB Enschede, The Netherlands

Abstract

The boosting framework has shown good performance in face recognition. By combining a set of features with Adaboost, a similarity function is developed which determines if a pair of face images belongs to the same person or not. Recently, many features have been used in combination with Adaboost, achieving good results on the FERET database. In this paper we compare the results of several features on the same database and discuss our solutions on some of the open issues in this method. We compare the boosting framework with some standard algorithms and test the boosting algorithm under difficult circumstances, like illumination and registration noise.

1 Introduction

Face recognition can be used in applications such as identity authentication for credit card or passport, access control and video surveillance. However, most automatic face recognition systems still only work under constrained conditions. Recently, several papers propose a new framework for face recognition, which combines a set of features into a classifier using a boosting algorithm. In this paper we investigate this new framework to determine if it can deal with the challenges in face recognition, like noisy registration, illumination and expressions.
In several papers [1, 2, 3, 4, 5] the boosting framework is used for face recognition. These papers use simple features like rectangle features, Gabor features or Local Binary Patterns (LBP) to make a function which can evaluate the similarity between two face images. This function learns the differences between face image pairs of the same person (intrapersonal) and image pairs of different persons (extrapersonal). The Adaboost algorithm constructs this similarity function by selecting a combination of features. This function is able to separate intrapersonal and extrapersonal image pairs. To the outcome of this similarity function we can apply a threshold for face verification. This outcome can also be used for face identification to find the most similar face image in a gallery.
The boosting framework is built up out of several parts, which allow different settings. For instance, this framework allows us to use different features, but also a combination of those features. Further more, different versions of the Adaboost algorithm [6, 7] can be used. Because we use image pairs we have to deal with large amounts of training data, given a database of N images per class with K individuals, the total number of pairs is $\binom{KN}{2}$, where only a small part is of the same individual. Several resampling methods are already proposed to solve this problem. In this paper we discuss these settings of the framework. We use the results on the FERET database as comparison to the other paper and we also experiment with more challenging datasets, trying to find the limits of this face recognition approach.
This paper is organized in the following way: In section 2 we explain in detail our method, which includes the features, the Adaboost algorithm and the resampling

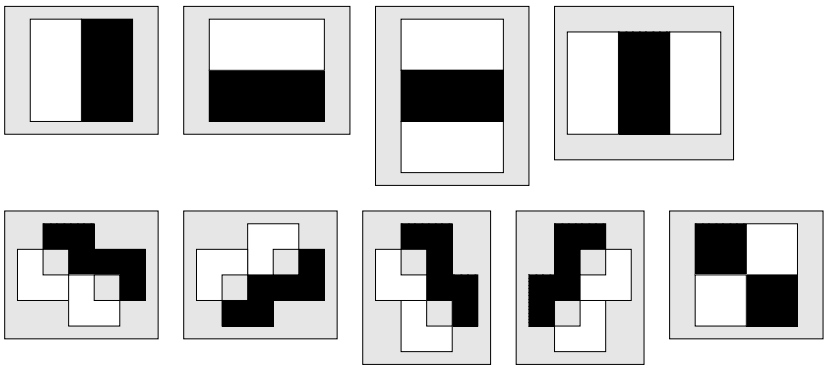


Figure 1: Rectangle Features use by Adaboost

method we use. Section 3 describes the experiments and results on the the different datasets. In the final section, we present the conclusions and some future work.

2 Method

The boosting framework can roughly be divided into three parts: the features, the Adaboost algorithm and the resampling method. In this section, we describe the different parts of the framework we used in our implementation.

2.1 Features

One of the main issues in face recognition is what features to use to represent a face. In the boosting framework we can use different features and also combinations of these features. In [1] Jones and Viola use the rectangle features also used in face detection [8]. In [3, 4] Gabor features are used in combination with Adaboost and [5] uses the local binary pattern (LBP) as features. We first decided to try a combination of the rectangle features and LBP. This because the rectangle features are very fast in computation, while the LBP features are less sensitive to illumination and achieved the best result in combination with Adaboost [5].

2.1.1 Rectangle Features

Rectangle features are computed by summing the pixelvalues in black areas and substract these with the sum of the pixelvalues in the white area. Although, there are already all kind of different rectangle features, especially in the area of face detection [9], we only use the features given in Figure 1. By adding more features the time to train Adaboost increases, so there is always a trade off between training time and a richer feature set. One of the main advantages of the rectangle features is that the computation can be sped up by using the integral images [8], making it one of the fastest features to compute.

2.1.2 Local Binary Pattern

The Local Binary Patterns (LBP) are introduced in [10] and have shown to be a robust feature in face recognition [11]. They can also be used as preprocessing step in face recognition to remove illumination [12]. The standard LBP, shown in Figure 2 gives the 3×3-neighbors the value 0 if they are smaller than the center pixel value and 1

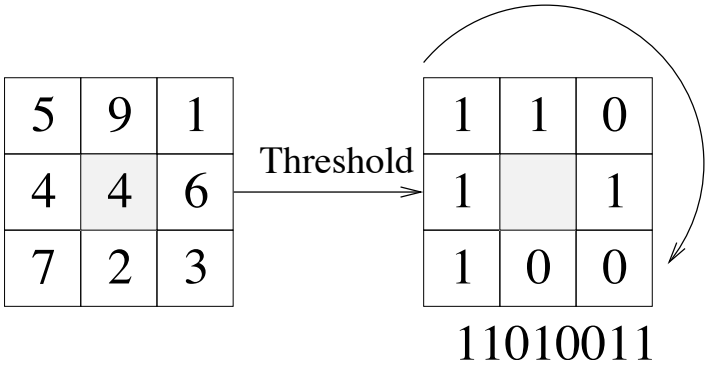


Figure 2: Local Binary Features

otherwise. This gives us a 8-bit string, which can be used to represent the texture at that point. It is also possible to select neighborhoods of bigger sizes, this allows to capture larger scale structures. Because we work on relative small face images of a resolution of 45×45 we only calculate LBPs in neighborhoods of 1 pixel. In this method, we compare in a certain region the histograms of different the LBP values.

2.2 Adaboost

Adaboost is the machine learning algorithm which is used to combine the features given in section 2.1. In the boosting framework we train a face similarity function which determines if the two faces belong to the same person or to a different person. The face similarity function is given in below:

$$F(I_1, I_2) = \sum_{t=1}^{T} f_t(I_1, I_2) \tag{1}$$

In this equation, I_1 and I_2 are the face images which are compared to see if they belong to the same person. The function f_t represents a weak classifier used by Adaboost. The final classifier is a weighted sum over all the selected weak classifiers. A weak classifier f_j is given below, where α and β are given by Adaboost.

$$f_j(I_1, I_2) = \begin{cases} \alpha & \text{if } |\phi_j(I_1, I_2)| < t_j \\ \beta & \text{otherwise} \end{cases} \tag{2}$$

In this Formula ϕ_j is the feature output for the given image pair and t_j is the feature threshold. This means that for every feature ϕ_j, we first determine the optimal threshold t_j, by minimizing the weighted training error, Equation 5. Determining the optimal threshold for each feature consumes most of the training time. To speed up this process we sometimes also estimate the threshold by taking the weighted means of the outputs of all positive and negative examples, and take the value in the middle as threshold.

There are multiple versions of the Adaboost algorithm, we use Real Adaboost, described by Schapire and Singer in [6], which almost always outperforms the older Discrete Adaboost. The main difference between these versions is the calculation of α and β. α and β are given in Equations 3 and 4, for a further explanation see [6] or [1].

$$\alpha = \frac{1}{2} \log\left(\frac{\sum_{i:y_i=+1 \wedge \phi_j(x_i)<t_j} w_i}{\sum_{i:y_i=-1 \wedge \phi_j(x_i)<t_j} w_i}\right) \tag{3}$$

$$\beta = \frac{1}{2}\log(\frac{\sum_{i:y_i=+1\wedge\phi_j(x_i)\geq t_j} w_i}{\sum_{i:y_i=-1\wedge\phi_j(x_i)\geq t_j} w_i}) \tag{4}$$

There are some other boosting algorithm which can be used, like GentleBoost and LogitBoost [7], but it depends on the dataset which boosting algorithm works best. To be able to make a comparison with [1] we choose to use the Real Adaboost algorithm. To each example $x_i = (I_1^i, I_2^i)$, a label $y_i \in \{+1, -1\}$ and a weight w_i are assigned. On each round Adaboost selects the classifier with the smallest error ϵ_j with respect to weights:

$$\epsilon_j = \sum_{i:y_i=+1\wedge\phi_j(x_i)\geq t_j} w_i + \sum_{i:y_i=-1\wedge\phi_j(x_i)<t_j} w_i \tag{5}$$

Note that we do not minimize the t_j on each round of the Adaboost algorithm, this would lead to much more computation without significant improvement in the performance. The Real Adaboost algorithm is given in Figure 3.

- Given examples $(x_1, y_1), ..., (x_N, y_N)$ where $N = A + B$. A is the number of examples with $y_i = +1$ and B is the number of examples with $y_i = -1$
- $w_{1,i} = \frac{1}{2A}$ for those examples with $y_i = +1$ and $w_{1,i} = \frac{1}{2B}$ for those examples with $y_i = -1$
- Let R be the number of rounds to boost before resampling
- For $t = 1, ..., T$
  - Normalize the weights, $w_{t,i} = \frac{w_{t,i}}{\sum_{k=1}^{N} w_{t,i}}$
  - For each ϕ_j find the ϕ_j which minimizes the error, ϵ_j, in Equation 5
  - Choose α and β according to Equations 3 and 4
  - Update the weights:

$$w_{t+1,i} = w_{t,i} e^{-f_t(x_i)y_i} \tag{6}$$

  - If t is a multiple of R then resample.
- The final strong classifier is:

$$F(x) = \sum_{t=1}^{T} f_t(x) \tag{7}$$

Figure 3: Adaboost algorithm

2.3 Resampling

The number of positive examples is small in comparison to the number of negative examples. The resampling method corrects this imbalance by giving Adaboost a small negative subset of $\mathcal{M}$ examples to train on. To make use of the entire training set of $\mathcal{N}$ examples, the resampling method selects after a number of boosting rounds ($R = 40$) a new subset of $\mathcal{M}$ examples to train the next boosting rounds. In our paper, we use a similar resampling approach as described in [1]. We initialize the weight w_j for all the negative examples x_j in the training set to $\frac{1}{2B}$. After a number of boosting rounds the weights of negative examples used in Adaboost have changed. We take all the weights and define a vector with the cumulative weights:

$$c_k = \sum_{i=1}^{k} w_i \text{ for } k = \{1, 2, ..., \mathcal{N}\} \tag{8}$$

The total weight of all negative examples is given by $c_{\mathcal{N}}$. We generate a random number r_s (uniformly distributed) on the interval $[0, c_{\mathcal{N}}]$. The random number r_s is associated with example x_j for which $c_j < r_s < c_{j+1}$ holds. We repeat this procedure until we have $\mathcal{M}$ unique examples. If an example is selected it gets a weight 1, if the same example is selected once more we increase the weight with 1 without adding the example twice, finally we normalize the total weight to 0.5.

This resampling method will choose more examples with larger weights while trying to preserve the overall distribution. This method also causes that mutual information is included in the first boosting iterations after the resampling step. This can be solved by calculating weights for all the examples by applying the strong classifier (Equation 9), before selecting the different examples.

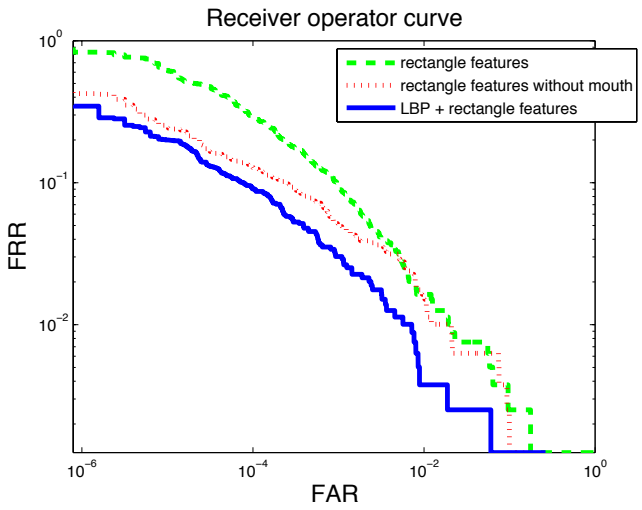
$$w_i = e^{-F(x_i)y_i} \tag{9}$$

3 Experiments and Results

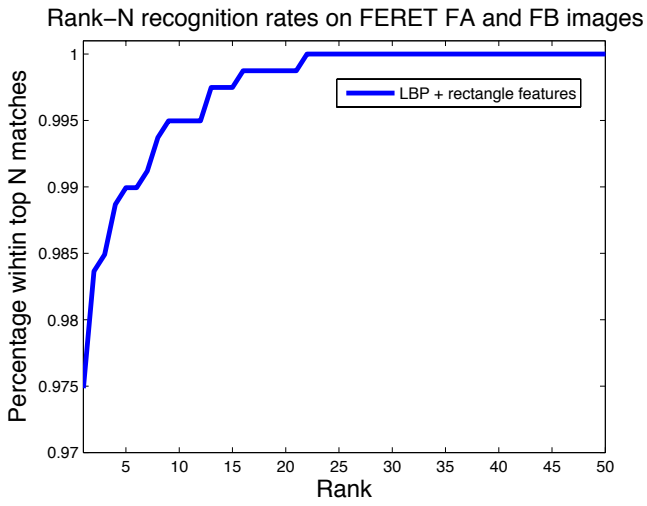
We performed several experiments with this boosting framework, where we varied different settings. We experimented on two face databases, namely the FERET and FRGCv1 dataset.

3.1 FERET

We first tested the boosting framework on the FERET database, which makes it comparable to the results in other papers on the boosting framework. We use the FA and FB images, where we have 1196 FA images and 1195 FB images. All subjects have exactly one image as gallery image and one image as probe image, except for one person. We use one third of image pairs (396) of the FERET as training set and the rest is used as gallery and probe set. The resolution of the images is 45×45 pixels, running Adaboost for 120 rounds giving us 120 features. We perform experiments with only rectangle features and with a combination of LBP and rectangle features. These experiments gave results which are comparable to the results reported in [1]. Our results are shown in Figure 4. Figure 4(a) shows the results of different features, where we tested with rectangle features and with a combination of LBP and rectangle features (dashed and solid lines). We also experimented with a different region of interest ignoring the mouth region getting even better results with only the rectangle features



(a) ROC on the FERET



(b) Rank-N recognition rates on the FERET

Figure 4: ROC and Rank-N recognition rates on the FERET FA and FB images of the boosting framework

training ↓ test →	no noise	3% noise	5% noise
no noise	1.9	4.2	7.8
3% noise	2.0	2.9	-
5% noise	2.3	-	5.5

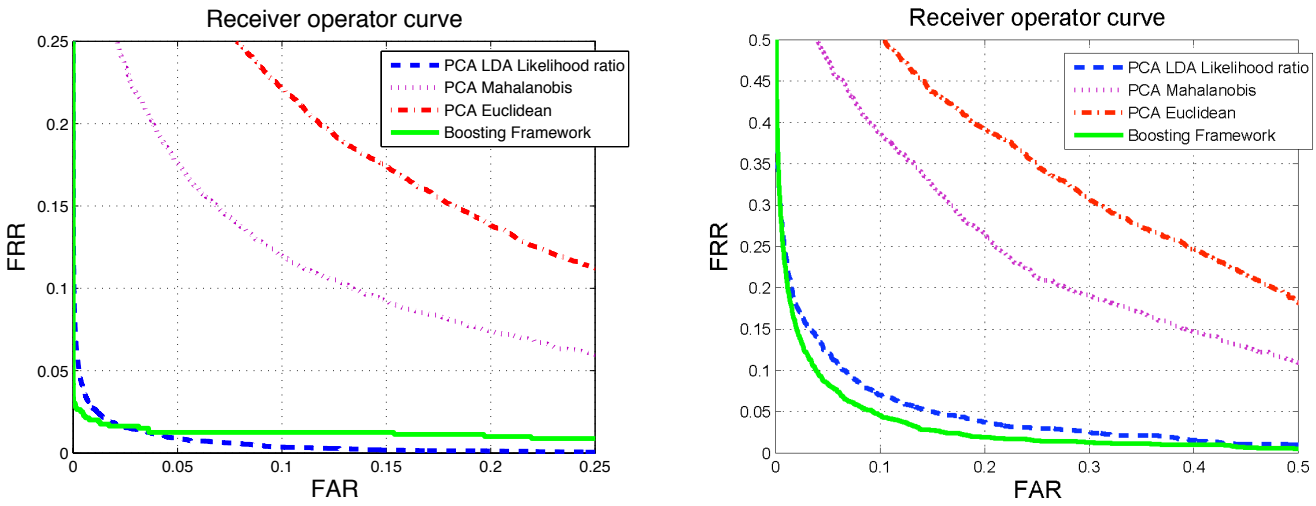
Table 1: Robustness to registration errors (EER in %)

(dotted line). Our implementation achieves a rank-1 recognition rate of 97.5% and with EERs of approximately 1%. This is similar to what other papers report which use the boosting framework. Using LBP in combination with rectangle features improves the results, where the first features selected by Adaboost are LBP features. We found out that other small improvements in the boosting framework are not measurable on the FERET database. For this reason we experimented with this algorithm under more difficult circumstances.

One assumption of this algorithm is that the registration of the face images has to be correct. In practise it is very hard to realize good registration. To simulate mistakes in registration we added gaussian noise to the landmarks of the FERET database. Results of these experiments are given in Table 1 showing that registration noise has of course a negative effect but is not disastrous. These results are reached after 40 rounds of Adaboost, without resampling.

3.2 FRGC

Because of the excellent results on the FERET database, we tested our method on a more challenging dataset. We used two subsets of the FRGC version 1 database, one subset with images taken under controlled conditions and another subset contains images taken under uncontrolled conditions, most cause by illumination. We divided both these datasets randomly into 3 subsets, giving the training set at least 2 images



(a) Controlled dataset

(b) Uncontrolled dataset

Figure 5: ROC on the subsets of the FRGCv1 database

for each person. The FRGC dataset taken under controlled conditions has a training set containing 1363 face image, a gallery set containing 1237 images and a probe set containing 1161 face images. The FRGC dataset taken under uncontrolled conditions has a training set containing 775 face image, a gallery set containing 550 images and a probe set containing 486 face images. The FRGC database contains 275 individuals. We compared our results with the PCA Euclidean distance, PCA Mahalanobis distance and our log-likelihood ratio based approach described in [13]. For PCA we use 100 components and for LDA we use 50 components.

The FRGC database is a more challenging database, which can be seen in Figure 5. Our method outperforms the normal PCA approaches and gives slightly better results in EER than the log-likelihood ratio classifier on the controlled set of faces 5(a). During this experiment, we combined 280 weak classifiers with Adaboost using both rectangle features and LBPs. For the uncontrolled set of images we combined 500 rectangle features and LBPs, results are shown in Figure 5(b). On the uncontrolled set of faces the difference between the log-likelihood ratio and the boosting framework is even bigger, which indicates that the boosting framework is more robust to illumination.

4 Conclusions

The boosting framework is a strong method to recognize faces. Although we achieve the excellent results using the boosting framework, improvement can be made on the details of this method, like the features selection, resampling approach, version of Adaboost. We have shown that combinations of features improve the results, although it takes more time to train the algorithm. In this paper we show that this method is also promising under more difficult circumstances. This has been demonstrated by the performance of the boosting framework under bad registration. The experiments on the FRGC version 1 database also show that the algorithm performs well on more difficult conditions. In the future we hope to focus more on the illumination problems, in which this algorithm can play an important role. Also the use of cascading structures in combination with the boosting framework can speed up recognition on large databases.

References

[1] Micheal J. Jones and Paul Viola, "Face recognition using boosted local features," in *International Conference on Computer Vision*, 2003.

[2] Guo-Dong Guo, Hong-Jiang Zhang, and Stan Z. Li, "Pairwise face recognition," *iccv*, vol. 02, pp. 282, 2001.

[3] Peng Yang, Shiguang Shan, Wen Gao, Stan Z. Li, and Dong Zhang, "Face recognition using ada-boosted gabor features," *fgr*, p. 356, 2004.

[4] Lei Zhang, Stan Z. Li, Zhi Yi Qu, and Xiangsheng Huang, "Boosting local feature based classifiers for face recognition," in *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 5*, Washington, DC, USA, 2004, p. 87, IEEE Computer Society.

[5] Guangcheng Zhang, Xiangsheng Huang, Stan Z. Li, Yangsheng Wang, and Xihong Wu, "Boosting local binary pattern (lbp)-based face recognition," in *Chinese Conference on Biometric Recognition*, 2004, vol. SINOBIOMETRICS 2004, pp. 179–186.

[6] R. E. Shapire and Y. Singer, "Improving boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37:3, pp. 297–336, 1999.

[7] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," Tech. Rep., Dept. of Statistics, Stanford University, 1998.

[8] Paul A. Viola and Michael J. Jones, "Rapid object detection using a boosted cascade of simple features.," in *CVPR (1)*, 2001, pp. 511–518.

[9] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," *DAGM'03, 25th Pattern Recognition Symposium*, pp. 297–304, 2003.

[10] Timo Ojala, Matti Pietikainen, and David Harwood, "Comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, pp. 51–59, 1996.

[11] T Ahonen, A Hadid, and M Pietikinen, "Face recognition with local binary patterns," in *Computer Vision - ECCV 2004*, 2004, pp. 469–481.

[12] Guillaume Heusch, Yann Rodriguez, and Sebastien Marcel, "Local binary patterns as an image preprocessing for face authentication," *fgr*, pp. 9–14, 2006.

[13] R.N.J. Veldhuis, A.M. Bazen, W. Booij, and A.J. Hendrikse, "Hand-geometry recognition based on contour parameters," in *Proceedings of SPIE Biometric Technology for Human Identification II*, Orlando, FL, USA, March 2005, pp. 344–353.