EURASIP Journal on Applied Signal Processing 2005:13, 2061–2071
© 2005 David Masip et al.

Feature Extraction Methods for Real-Time Face Detection and Classification

David Masip
Centre de Visió per Computador (CVC), Departamento de Informática, Universitat Autònoma de Barcelona, Bellaterra 08193, Spain
Email: davidm@cvc.uab.es

Marco Bressan
Centre de Visió per Computador (CVC), Departamento de Informática, Universitat Autònoma de Barcelona, Bellaterra 08193, Spain
Email: marco.bressan@xrce.xerox.com

Jordi Vitrià
Centre de Visió per Computador (CVC), Departamento de Informática, Universitat Autònoma de Barcelona, Bellaterra 08193, Spain
Email: jordi@cvc.uab.es

Received 22 December 2003; Revised 29 November 2004

We propose a complete scheme for face detection and recognition. We have used a Bayesian classifier for face detection and a nearest neighbor approach for face classification. To improve the performance of the classifier, a feature extraction algorithm based on a modified nonparametric discriminant analysis has also been implemented. The complete scheme has been tested in a real-time environment achieving encouraging results. We also show a new boosting scheme based on adapting the features to the misclassified examples, achieving also interesting results.

Keywords and phrases: face detection, face recognition, boosting, feature extraction.

1. INTRODUCTION

As computers become faster and faster, new applications dealing with human faces become possible. Examples of this applications are face recognition applied to surveillance systems, gesture analysis applied to user-friendly interfaces, or gender recognition applied to reactive marketing. We will propose here a global face detection and recognition framework, which has achieved good results in an uncontrolled environment. Usually, working under uncontrolled conditions is one of the hardest problems of computer vision, for example, in applications where illumination presents strong changes, or where we have to deal with objects under unpredictable movements. We have tested the system in a real environment, with no restrictions on scale and illumination, achieving real-time satisfying performance.

According to [1] face detection schemes can be classified into four different categories, although some methods can belong to more than one category.

(1) Knowledge-based methods, where some rules or relationships between features are encoded. Kotropoulos and Pitas followed this approach using projection profiles to locate the face [2]. We have used a similar technique to locate the eyes once the face has been detected using our method.

(2) Feature-invariant approaches, where the idea is to detect the facial features first, such as eyes, mouth, eye brows, and group them into candidate faces [3].

(3) Template-matching methods, where there is a predefined face pattern that is correlated with the image. Point distribution models (PDMs) have also been used for this purpose [4].

(4) Appearance-based methods, where the goal is to train a classifier that learns the features of the faces from a training set with face and nonface images. Many classic techniques such as principal component analysis [5], Gaussian mixture models [6], neural networks [7], hidden Markov models [8], support vector machines [9], and probabilistic models [6] have been applied in this approach.

Many methods of face recognition have also been proposed. Basically they can be divided into holistic template-matching-based systems, geometrical local-feature-based schemes, and hybrid schemes [10].

(i) The holistic methods use the whole image as a raw input to the learning process. Examples of this techniques are principal component analysis [5], independent component analysis [11], or support vector machines [12] applied to face recognition.

(ii) In the feature-based schemes, some structural features are extracted, such as eyes, mouth, and their local appearance, position, or relative relationship are used for training the classifier. The most successful technique is the elastic bunch graph matching presented in [13] where the authors use Gabor wavelets to extract the basic features for the graph-matching scheme.

(iii) Hybrid methods try to use the best of the holistic and feature-based approaches combining local features and the whole face to recognize. An example of hybrid methods is the use of eigenfeatures [14], which extends the idea of eigenfaces to specific regions of the face such as mouth, nose, or eyes.

Among the holistic methods, appearance-based methods are the most successful. They are commonly implemented following these steps.

(1) *Image preprocessing* [15], where usually an illumination correction is performed, followed by the localization of some parts of the face for geometrical alignment that makes the feature-based approaches more accurate. Usually the center of the eyes are located, and faces are warped in such a way that distance between eyes remains stable within subjects.

(2) *Feature extraction*. Dimensionality reduction techniques have shown important advantages in some pattern recognition tasks and face processing is not an exception. Usually we achieve a compression of the input data, reducing the storage needs. In other cases there is also an improvement of the classification results due to reduction of the noise present in the most part of the natural images. Principal component analysis is perhaps one of the most spread dimensionality reduction techniques [5, 16]. The goal in PCA is to find a linear projection that preserves the maximum amount of input data variance. Another approach is to take into account the labels of the input data points, and try to find the linear projection that best discriminates the space in different classes. Fisher linear discriminant analysis is an example of this kind of techniques [17, 18]. Nevertheless, the algorithm has some limitations, because the dimensionality of the resulting space after the projection is upper bounded by the number of classes, and there is also a Gaussian assumption in the input data. In this paper we will introduce the use of nonparametric discriminant analysis [19] for face recognition, a technique that overcomes the drawbacks of FLD, and a modification of the original NDA algorithm that increases its performance.

(3) *Feature classification*. Once the proper features are extracted, any classifier can be applied. The most common ones are the nearest-neighbor classifier using either Euclidean or angle distance, and neural networks.

Most of these methods have been successfully used in artificial environments, but do not perform well in many real-world situations as several independent tests have documented [20].

In the next section, we introduce the face detection method that is based on a classifier combination technique, the AdaBoost algorithm. We also will suggest a modification of the algorithm. Then we will focus on the classification engine for face recognition, and the discriminant analysis used as feature extraction will be presented. In Section 4 we will show the performed experiments. First we have tested the detection scheme using images extracted from two public face databases. Also the recognition layer has been tested in the real environment. We finalize the paper with the conclusions.

2. FACE DETECTION

Today, the most promising approach for face detection is an appearance-based method that is based on the classification, using ensemble methods, of an overcomplete set of simple image features. This approach was first proposed by Viola and Jones [21] and developed by Lienhart [22]. The main characteristics of this approach are a cascade architecture, an overcomplete set of rectangle features, and an algorithm based on AdaBoost for constructing ensembles of rectangle features in each classifier node. Much of the recent work on face detection following Viola-Jones has explored alternative boosting algorithms such as FloatBoost [23], GentleBoost [22], and asymmetric AdaBoost [24]. In the next section we present the AdaBoost approach, that also constitutes the central part of our detector.

2.1. Classifier combination and AdaBoost

Sometimes it is possible to improve the performance of a poor classifier by combining multiple instances of it, that will be called weak classifiers, in a more powerful decision rule. In the literature, we can find three different approaches to do it: boosting [25], bagging [26], and random subspace methods [27]. The difference between these algorithms is the way how they combine the classifiers. The main idea of random subspace methods is to use only a subset of the features of each vector. Different classifiers are created using random subspaces by sampling the original-data feature space. The final decision rule is usually a majority voting among the different classifiers. The RSM method is specially useful when dealing with high-dimensional data and reduced training sets.

Bagging combines the results of different classifiers by sampling the training set. A new classifier is trained using each subset of the training vectors, and then every test vector is classified using all the classifiers, and a final decision rule is performed (weighted majority voting, etc.). Bagging is often useful when we have outliers in the training set, because they are isolated in some subsets and do not alter the global results.

The idea behind boosting is similar to bagging, but the classifiers are trained in a serialized way. At each training step a new classifier is created. These classifiers are called weak classifiers. The training samples are classified using it, and a set of weights are modified according to the classification results (the weights of the misclassified examples are increased). Then a new classifier is trained taking into account these weights. The algorithm is iterated a certain number of

```
Given the training samples $X_1, X_1, \ldots, X_n$

(1) Initialize the weight vector to 1 $\forall W_{i=1,\ldots,n}$.
(2) For $s = 1, 2, \ldots, S$ do:
      (i) Resample the training data samples $\mathbf{X}$ according to their actual weights, obtaining the samples $\mathbf{H}$.
     (ii) Train a classifier $C_s(\mathbf{H}_y)$.
    (iii) Classify the samples $\mathbf{X}$ using $C_s$.
     (iv) Compute the probability of the error of classification taking into account the weights as follows:
              $\varepsilon_s = \frac{1}{N}\sum_{i=1}^{N} W_i^s \xi_i^s,$   (1)
          where
              $\xi_i^s = \begin{cases} 1, & \text{if } \mathbf{X}_i \text{ was wrongly classified in the step } s, \\ 0, & \text{otherwise.} \end{cases}$   (2)
      (v) Compute $\zeta_s$ as
              $\zeta_s = \frac{1}{2}\log\left(\frac{1-\varepsilon_s}{\varepsilon_s}\right).$   (3)
     (vi) If $\varepsilon_s < 0.5$ set $\mathbf{W}_i^{s+1} = \mathbf{W}_i^s \exp(\zeta_s \xi_i^s)$ for each training vector $i$, and then normalize the weights in a
          way that $\sum_{i=1}^{N} \mathbf{W}_i^{s+1} = N$.
    Otherwise restart the algorithm.
(3) Finally the classifiers obtained are combined using a weighted majority voting using the coefficients $\zeta_s$
    that encode a measure of the error in each step. The final decision rule for each new test vector is
              $O(x) = \sum_s \zeta_s L_s > 0,$   (4)
    where $L$ is the label obtained in the classifier $s$, $L \in (-1, 1)$.
```

Algorithm 1: General AdaBoost algorithm.

steps, and the final decision rule is a weighted combination of the intermediate classifiers.

In this work, we have chosen as the boosting scheme a variant of the AdaBoost algorithm [25] (see Algorithm 1). In the literature we can find some variants of the original algorithm depending on the use of the weights in the learning process. One possible approach is to use a classifier that can accept weights as inputs, or simply multiply the training vectors by their weights. Another approximation is to use the weights to resample the training set in such a way that misclassified examples must have higher probability of being chosen for the next step. We have used this scheme in our experiments.

2.1.1. *Boosting using fixed features*

We propose a detection process that makes use of the boosting scheme of Algorithm 1 where the chosen weak classifier is the naive Bayes classifier. In particular we have assumed a *Bernoulli* distribution on the data [28]. In fact, this assumption is justified because we have used binary data instead of the original images.

For each input image we have extracted a set of fixed features, using a detector of ridges and valleys (see [29] for more details) due to its robustness against changes in the illumination. The final features have been obtained by thresholding the response of the filters, assigning -1 to a pixel when it is situated in a valley, 1 when it is on a ridge, and 0 otherwise (see Figure 1). To convert this ternary representation into a binary one, we have separated the filtered image into two representations where we put to 1 the pixels where there is a ridge/valley and 0 otherwise. Both representations have been vectorized and concatenated (so at the end we have a binary vector with double dimensionality).

As we use a Bayesian classifier, for each target image $\mathbf{X}_i$ we decide that $\mathbf{X}_i$ is a face if $p(\mathbf{X}_i|C_{\text{Faces}}) > p(\mathbf{X}_i|C_{\text{NonFaces}})$ (see the MAP rule [30]). To estimate the conditional probabilities for the faces, we use

$$p(X|C_{\text{Faces}}) = \prod_{d=1}^{D} p_d^{X_d} q_d^{1-X_d}, \tag{5}$$

where $\mathbf{X}_d$ is dth pixel of the image, p_d is the probability of finding a 1 in the pixel d, and q_d the probability of finding a 0 in the pixel d ($p_d = 1 - q_d$). This probabilities p and q can be estimated directly from the training samples by finding the frequencies of the ones and zeros of the face vectors. In a similar way the conditional probabilities for the nonfaces are

$$p(X|C_{\text{NonFaces}}) = \prod_{d=1}^{D} \tilde{p}_d^{X_d} \tilde{q}_d^{1-X_d}, \tag{6}$$

where $\tilde{p}$ and $\tilde{q}$ are obtained as p and q but using the nonface instead of the face samples.

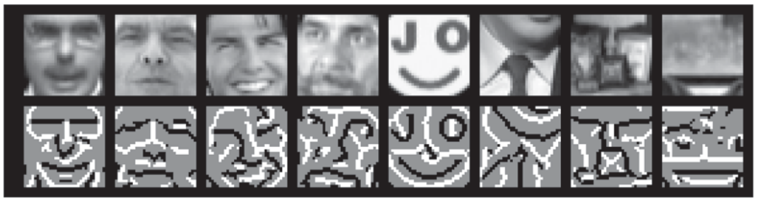


Figure 1: Example of ridges and valleys detection for a subset of face and nonface images.

Face detection requires high detection rates (higher than 90%) and very low false positive rates (lower than 0.0001%) in order to be useful. In order to get this kind of false positive rates, Viola and Jones [21] proposed the use of boosted classifiers in a cascade architecture where each classifier was specialized in classifying a specific subset of nonfaces while keeping high detection rates. We have also followed this approach.

2.1.2. Boosting using adaptable features

Usually the boosting algorithm uses a fixed set of features extracted from the input data $\mathbf{X}$. These features can be in the most simple case the pixel values of the image, or in other cases, the result of some kind of filtering (see [21]) or feature extraction, like the method proposed in the last section.

We also have considered a completely different approach. The idea is to introduce in the boosting process the feature extraction in such a way that at every fixed rate of boosting steps, the features per each image are recomputed in a dynamical way. To compute the new features, the weights of the training vectors are used, so every time the features are more focused in the more difficult examples.

In our experiments, we have used a variant of the nonnegative matrix factorization algorithm [31] to recompute the features and our classifier was a single-layer perceptron. The modified algorithm uses a weighting matrix to focus the algorithm in some specific samples (see [32] for more details).

The basic NMF technique proposed by Lee and Seung [31] tries to find parts in the training objects, providing a part-based set of bases. The key point of the NMF algorithm is the use of the nonnegativity constraint applied to both the set of bases and coefficients that represent each vector in the reduced space. This nonnegativity constraint provides always sparse bases due to the fact that the weighted combination of the bases is always additive, so bases cannot have big regions active because once a big portion of an image is used as a base, it cannot be removed (due to the constraint of additivity).

The general formulation for the NMF algorithm consists of given a set of N D-dimensional data points $\mathbf{X}$ (representing positive-valued pixel images), and finding the nonnegative bases $\mathbf{W}$ that satisfy

$$\mathbf{X}_{ij} \approx (\mathbf{W} * \mathbf{H})_{ij} = \sum_{d=1}^{D} \mathbf{W}_{id}\mathbf{H}_{dj}. \tag{8}$$

Usually we will use a number of bases $R < D$, to obtain an important dimensionality reduction, and a selection of the most useful features. In the case of face images it can be shown that the nonnegativity constraints achieve bases which represent specific parts of faces such as eyes, mouth, and so forth To find this set of bases and coefficients, the following update rules should be iterated:

$$\begin{aligned}
\mathbf{W}_{ij} &\longleftarrow \mathbf{W}_{ij} \sum_{d} \frac{Q_d \mathbf{X}_{id}}{(\mathbf{WH})_{id}} \mathbf{H}_{jd}, \\
\mathbf{W}_{ij} &\longleftarrow \frac{\mathbf{W}_{ij}}{\sum_k \mathbf{W}_{kj}}, \\
\mathbf{H}_{jd} &\longleftarrow \mathbf{H}_{jd} \sum_{i} \mathbf{W}_{ij} \frac{\mathbf{X}_{id}}{(\mathbf{WH})_{id}}.
\end{aligned} \tag{9}$$

We should take into account that matrices $\mathbf{W}$ and $\mathbf{H}$ must be randomly initialized with positive values. Notice also the inclusion of the diagonal matrix Q, which encodes the weights of the samples. We use the same weights obtained at each step of the AdaBoost algorithm to set the matrix Q at each step, and focus the feature extraction to the most difficult examples.

The advantage of this adaptive boosting approach is that at every boosting step it is more specialized in the wrong classified examples, and the features that are being considered are learned, giving more importance to the most difficult examples. This must lead the algorithm to a faster convergence, achieving the same or even better results than the classic boosting but in fewer steps.

3. FACE RECOGNITION

In our scheme we will use a discriminant analysis to project the data into a reduced space followed by a nearest-neighbor classification. The dimensionality reduction achieved in the feature extraction is essential in real-time classification problems, learning invariances in the samples, and yielding a compact representation that reduces the storage and computational needs. Prior to the feature extraction the proper image normalization has been performed, in order to make the system more robust against changes in illumination and in geometric transformations.

3.1. Image preprocessing

The normalization engine has two parts. One takes care of illumination and a second one takes care of geometric

```
(1) Given the matrix $\mathbf{X}$ containing data samples placed as $N$ $D$-dimensional columns, the within-class scatter
    $\mathbf{S}^I$ matrix, and $M$ maximum dimensions of discriminant space.
(2) Compute eigenvectors and eigenvalues for $\mathbf{S}^I$. Make $\mathbf{\Phi}$ the matrix with the eigenvectors placed as
    columns and $\mathbf{\Lambda}$ the diagonal matrix with only the nonzero eigenvalues in the diagonal. $M^I$ is the number
    of nonzero eigenvalues.
(3) Whiten the data with respect to $\mathbf{S}^I$ to obtain $M^I$ dimensional whitened data
        $\mathbf{Z} = \mathbf{\Lambda}^{-1/2}\mathbf{\Phi}^T\mathbf{X}.$    (7)
(4) Compute $\mathbf{S}^E$ on the whitened data.
(5) Compute eigenvectors and eigenvalues for $\mathbf{S}^E$ and make $\mathbf{\Psi}$ the matrix with the eigenvectors placed as
    columns and sorted by decreasing eigenvalue.
(6) Preserve only the first $R = \min\{M^I, M, \mathrm{rank}(\mathbf{S}^E)\}$ columns, $\mathbf{\Psi} = \{\psi_1, \dots, \psi_R\}$ (those corresponding to
    the $R$ largest eigenvalues).
(7) The resulting optimal transformation is $\widehat{\mathbf{W}} = \mathbf{\Psi}^T\mathbf{\Lambda}^{-1/2}\mathbf{\Phi}^T$ and the projected data is, $\mathbf{Y} = \widehat{\mathbf{W}}\mathbf{X} = \mathbf{\Psi}^T\mathbf{Z}$.
```

Algorithm 2: General algorithm for solving the discriminability optimization problem stated in (11).

normalization. Given a candidate face, illumination was normalized based on the local mean and variance.Geometric normalization was quite simplistic. Tilt was not taken into account. Gray value frequencies were obtained for different unidimensional projections and this information was used for obtaining the position of the eyes and face pose. Two peaks in the horizontal projection and one peak in the vertical projection allow the localization of the pixels of the eyes in the image. Yaw and normal face rotations were normalized taking advantage of the almost bilateral symmetry of the faces with respect to the eyes.

Since this scheme results in a very poor approach, an additional threshold was introduced in order to decide the goodness of the normalization. It was determined empirically using the training samples and checking which is the reasonable range of values where the central pixel of the eyes can be located. Actually, this threshold acts as an additional layer of detection. Face candidates yielding values below this threshold were disregarded. Otherwise, we store the position of each eye in the original frame.

3.2. Face classification

As we use NN as classification rule, a proper feature extraction algorithm is needed, and discriminant analysis can be very useful for this task. In this section we will introduce the classic Fisher discriminant analysis to show later how its nonparametric extension can solve the main drawbacks of FLD: Gaussian distribution assumption and reduced dimensionality of the generated subspaces. We will also show our modification of the classic NDA [19], which is expected to improve the NN classification.

3.2.1. Fisher discriminant analysis

The goal of discriminant analysis is to find the features that best separate the different classes. One of the most used criterions $\mathcal{J}$ to reach it is to maximize

$$\mathcal{J} = \mathrm{tr}\,(\mathbf{S}^E\mathbf{S}^I), \tag{10}$$

where the matrices $\mathbf{S}^E$ and $\mathbf{S}^I$, generally represent the scatter of sample vectors between different classes and within a class, respectively. It has been shown (see [33, 34]) that the $R \times D$ linear transform that satisfies

$$\widehat{\mathbf{W}} = \arg \max_{\mathbf{W}^T\mathbf{S}^I\mathbf{W}=\mathbf{I}} \mathrm{tr}\,(\mathbf{W}^T\mathbf{S}^E\mathbf{W}) \tag{11}$$

optimizes the separability measure $\mathcal{J}$. This problem has an analytical solution based on the eigenvectors of the scatter matrices. The algorithm presented in Algorithm 2 obtains this solution [34]. The most widely spread approach for defining the within- and between-class scatter matrices is the one that makes use of only up to second-order statistics of the data. This was done in a classic paper by Fisher [17] and the technique is referred to as Fisher discriminant analysis (FLD). In FLD the within-class scatter matrix is usually computed as a weighted sum of the class-conditional sample covariance matrices. If equiprobable priors are assumed for classes C_k, $k = 1, \dots, K$, then

$$\mathbf{S}^I = \frac{1}{K}\sum_{k=1}^{K} \mathbf{\Sigma}_k, \tag{12}$$

where $\mathbf{\Sigma}_k$ is the class-conditional covariance matrix, estimated from the sample set. The between-class scatter matrix is defined as

$$\mathbf{S}^E = \frac{1}{K}\sum_{k=1}^{K} (\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_0)^T, \tag{13}$$

where $\boldsymbol{\mu}_k$ is the class-conditional sample mean and $\boldsymbol{\mu}_0$ is the unconditional (global) sample mean.

Notice the rank of $\mathbf{S}^E$ is $K - 1$, so the number of extracted features is, at most, one less than the number of classes. Also notice the parametric nature of the scatter matrix. The solution provided by FLD is blind beyond second-order statistics. So we cannot expect our method to accurately indicate which features should be extracted to preserve any complex classification structure.

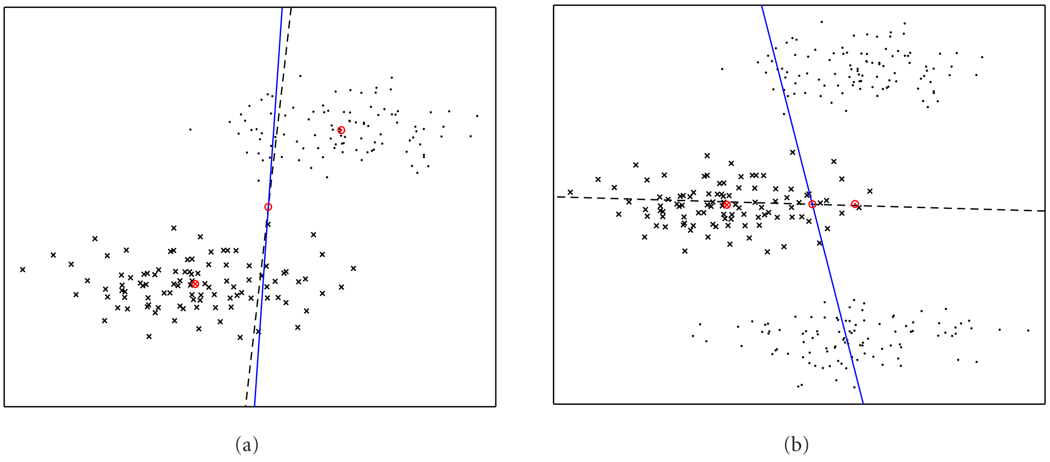


FIGURE 2: First directions of NDA (solid line) and FLD (dashed line) projections, for two artificial datasets. Observe the results in (b) where the FLD assumptions are not met.

3.2.2. *Nonparametric discriminant analysis*

In [19] Fukunaga and Mantock present a nonparametric method for discriminant analysis in an attempt to overcome the limitations present in FLD. In nonparametric discriminant analysis, the between-class scatter $\mathbf{S}^E$ is of nonparametric nature. This scatter matrix is generally full rank, thus loosening the bound on extracted feature dimensionality. Also, the nonparametric structure of this matrix inherently leads to extracted features that preserve relevant structures for classification. We briefly expose this technique, extensively detailed in [34].

In NDA, the between-class scatter matrix is obtained from vectors locally pointing to another class. This is done as follows. The extra-class nearest neighbor for a sample $\mathbf{x} \in C_k$ is defined as $\mathbf{x}^E = \{\mathbf{x}' \in \overline{C_k} / \|\mathbf{x}' - \mathbf{x}\| \le \|\mathbf{z} - \mathbf{x}\|,\ \forall \mathbf{z} \in \overline{C_k}\}$. In the same fashion we can define the set of intraclass nearest neighbors as $\mathbf{x}^I = \{\mathbf{x}' \in L_c / \|\mathbf{x}' - \mathbf{x}\| \le \|\mathbf{z} - \mathbf{x}\|,\ \forall \mathbf{z} \in C_k\}$.

From these neighbors, the extra-class differences are defined as $\mathbf{\Delta}^E = \mathbf{x} - \mathbf{x}^E$ and the intraclass differences as $\mathbf{\Delta}^I = \mathbf{x} - \mathbf{x}^I$. Notice that $\mathbf{\Delta}^E$ points locally to the nearest class (or classes) that does not contain the sample. The nonparametric between-class scatter matrix is defined as (assuming uniform priors)

$$\mathbf{S}^E = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{\Delta}_n^E)(\mathbf{\Delta}_n^E)^T, \qquad (14)$$

where $\mathbf{\Delta}_n^E$ is the extra-class difference for sample $\mathbf{x}_n$.

A parametric form is chosen for the within-class scatter matrix $\mathbf{S}^I$, defined as in (12). Figure 2 illustrates the differences between NDA and FLD in two artificial datasets, one with Gaussian classes where results are similar, and one where FLD assumptions are not met. For the second case, the bimodality of one of the classes displaces the class mean introducing errors in the estimate of the parametric version of $\mathbf{S}^E$. The nonparametric version is not affected by this situation. We now make use of the introduced notation to examine the relationship between NN and NDA. This results in a modification of the within-class covariance matrix which we also introduce.

Given a training sample $\mathbf{x}$, the accuracy of the 1-NN rule can be directly computed by examining the ratio $\|\mathbf{\Delta}^E\| / \|\mathbf{\Delta}^I\|$. If this ratio is more than one, $\mathbf{x}$ will be correctly classified. Given the $M \times R$ linear transform $\mathbf{W}$, the projected distances are defined as $\mathbf{\Delta}_W^{E,I} = \mathbf{W}\mathbf{\Delta}^{E,I}$. Notice that this definition does not exactly agree with the extra- and intraclass distances in projection space since, except for the orthonormal transformation case, we have no warranty on distance preservation. Equivalence of both definitions is asymptotically true. By the above remarks, it is expected that optimization of the following objective function should improve or, at least, not downgrade NN performance:

$$\widehat{W} = \arg \max_{\mathrm{E}\{\|\mathbf{\Delta}_W^I\|^2\}=1} \mathrm{E}\left\{\|\mathbf{\Delta}_W^E\|^2\right\}. \qquad (15)$$

This optimization problem can be interpreted as follows: find the linear transform that maximizes the distance between classes, preserving the expected distance among the members of a single class. Consider that

$$\mathrm{E}\left\{\|\mathbf{\Delta}_\mathrm{W}\|^2\right\} = \mathrm{E}\{(\mathbf{W}\mathbf{\Delta})^\mathrm{T}(\mathbf{W}\mathbf{\Delta})\} = \mathrm{tr}\,(\mathbf{W}^\mathrm{T}\mathbf{\Delta}\mathbf{\Delta}^\mathrm{T}\mathbf{W}), \qquad (16)$$

where $\mathbf{\Delta}$ can be $\mathbf{\Delta}^I$ or $\mathbf{\Delta}^E$. Substituting (16) in (17) we have that this last equation is a particular case of (11). Additionally, the formulas for the within- and between-class scatter matrices are directly extracted from this equation. In this case, the between-class scatter matrix agrees with (14), but the within-class scatter matrix is now defined in a nonparametric fashion:

$$\mathbf{S}_w = \frac{1}{N} \sum_{n=1}^{N} \mathbf{\Delta}_n^I \mathbf{\Delta}_n^{I\,T}. \qquad (17)$$

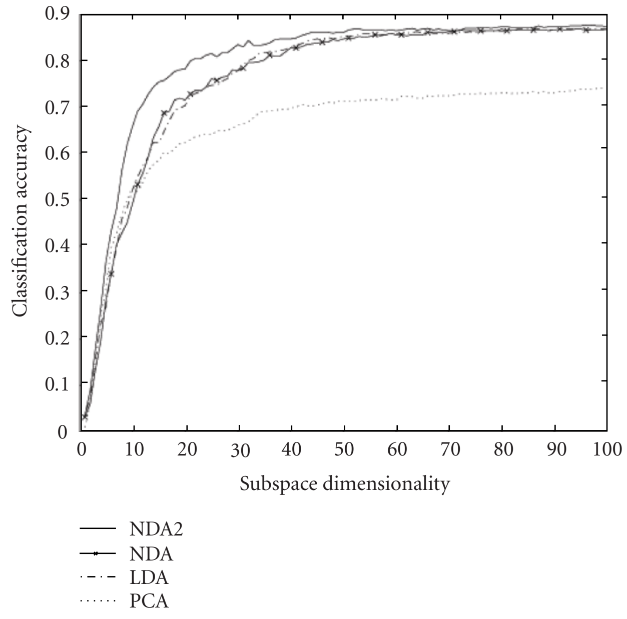


Figure 3: Recognition accuracy of different feature extraction algorithms with different subspace dimensionalities.

Given that we have an optimization problem of the form given in (11) the algorithm presented in Algorithm 2 can also be applied to the optimization of our proposed objective function (15).

To compare the improvement of the modified NDA in a face recognition problem, we have taken 10 samples (5 for training and 5 for testing) of each of the 115 subjects from the AR Face Database [35]. We have compared different feature extraction techniques for the same data set in the face recognition problem (using the 1-NN classifier). In the Figure 3, we show the classification accuracy as a function of the number of features extracted using NDA, principal component analysis, Fisher discrimiant analysis, and the modified NDA (NDA2 in the figure). It can be seen how NDA and the modified NDA outperform the other techniques, the modified algorithm being slightly better in almost all the dimensions. Other comparative studies between NDA and the most used feature extraction techniques using different data sets can be found in [28, 36], where it is shown that in low-dimensional subspaces, NDA performs better than the other techniques. In high-dimensional subspaces in problems with a large number of classes, both NDA and FLD have similar performance. Nevertheless, in this work we have chosen the modified NDA approach, given that it performs better in the general case (low- and high-dimensional subspaces and in low and large class problems).

4. EXPERIMENTS

We have designed a face verification application composed of two parts: the face detection in natural images, and the verification of the identity of the face. In the case of the face detector, we have opted for using a classic boosting scheme

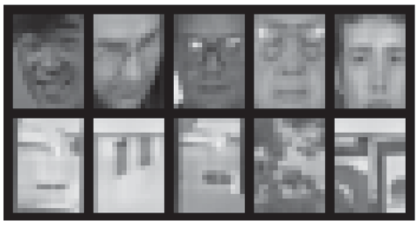


Figure 4: Examples of false negative and false positive images obtained in the boosting scheme. As can be seen, the false positives are images with the structure similar to a face (eyes, nose, and mouth).

instead of using adaptable features. Although the accuracies obtained in the adaptive scheme are higher, the main goal of this experiment was to design a real-time system, and the temporal performance of the adaptive scheme still was not enough to satisfy this real-time restriction.

This real-time requisite has also justified the election of a feature extraction algorithm (to reduce the amount of data storage) in the verification step. Also our face detector is restricted to faces with small rotations (less than 10 degrees), although the set up of the real application allows the capture of enough frontal views of each person.

4.1. Face detection

4.1.1. Face detection using fixed features

The Bayesian classifier defined in Section 2 has been used in a boosting scheme, where we have selected one feature at each step. We have used 6500 training images, with 1500 faces extracted from different public face databases (we have used the XM2VTS [37] and the first image of each subject from the AR Face Database [35]) and 5000 nonface images. To test the performance of the algorithm, a huge set of 28 000 images has been used (26 000 nonfaces and 2000 faces taken from the same databases). The global performance of the classifier in the first cascade level is 2.45% false rejection rate and only a 0.83% false positive rate. If we consider the full cascade with 32 levels (as described in [38]), the final detection rate is close to 94%, while we obtain one false positive every 100 000 samples.

In Figure 4, we show some examples of images wrongly classified by the detector. Some twisted and blurry faces are confused, and also some natural images which present a structure very similar to a face are also misclassified. Also we have tested the face detector using the CMU test database [39] (with 483 labeled faces), obtaining 94.2% of detection rate and just 82 false positives. Notice that these results are similar to the most common techniques used in the state of the art, although our detector is more simple, which allows us to use it in a real environment. A complete comparative study can be found in [1], where it can be seen that the best technique achieves 98% of detection rate, but also obtains a large amount of false detections (12 758). Other techniques such Fisher linear discriminant achieve just 74 false detections, although the detection rate decreases (93.6%). Our purpose

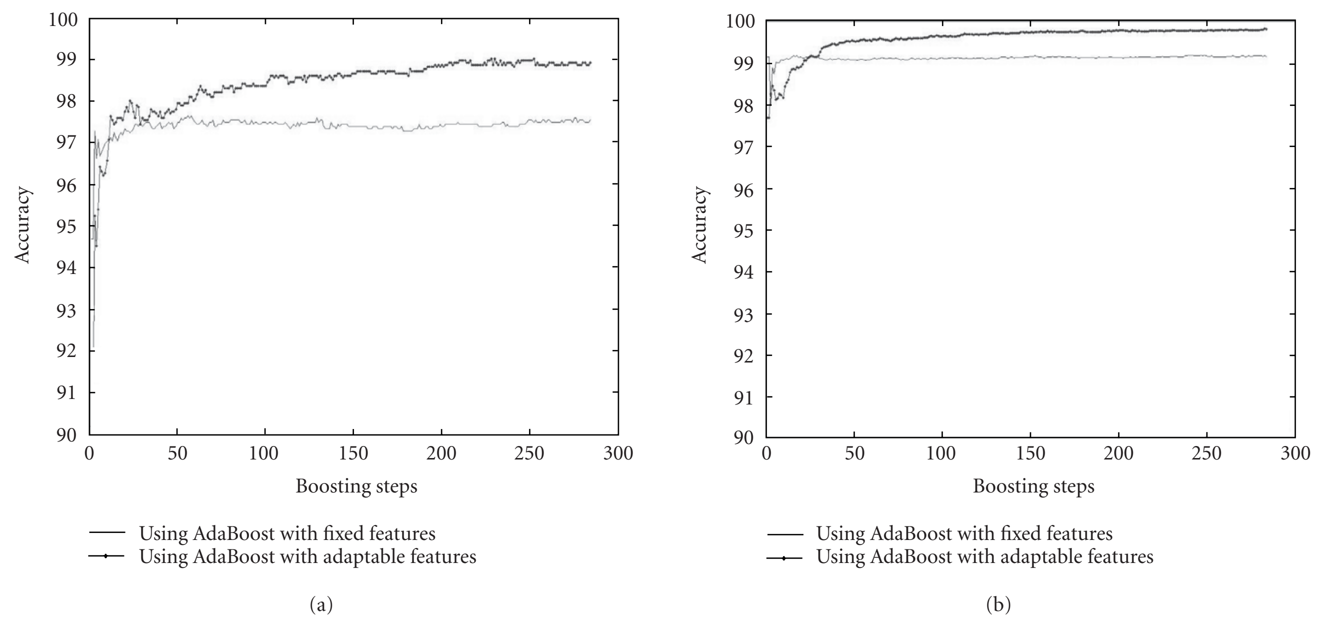


Figure 5: Accuracies obtained in the face and nonface images as a function of the boosting steps. As can be seen, the adaptive feature learning achieves better results.

achieves detection rates close to the other techniques, keeping a reasonable false detection rate and allowing a fast implementation.

4.1.2. Face detection using adaptable features

Although we have discarded its use in the real-time application due to the computational constraints, we also show the results obtained with the adaptable boosting scheme to compare the efficiency of both algorithms and highlight that there is an increase in the detection rates. In fact, the introduction of the feature extraction into the boosting scheme is still an interesting line of research and can be still optimized.

We have trained the boosting scheme using the same 1500 face images as above, and also the 5000 nonface images extracted from random natural images. To test the performance of the algorithm, the same 28 000 images have been used. We have achieved a 99.73% of global accuracy, with only 1.07% of false rejection and 0.2% of false positive ratio. We have considered a correct detection that each subimage has at least a face covering 80% of its surface. In Figure 5, it can be seen how the use of the adaptable feature approach can improve the results can be improved significantly as we detect 1.4% more faces. In a similar way we are able to improve the rejection of nonface images. On the other hand, the computational needs of the adaptable version have also increased, and thus it takes 12 seconds to process a single image in a Pentium 4 2.4 GHZ platform using Matlab 6.0 code.

4.2. Face recognition in an uncontrolled environment

A second experiment was performed using the modified NDA representation for real-time recognition in an uncontrolled environment. We now detail the design of this experiment, which attempts to simulate a surveillance situation. As most face processing applications, ours consisted in three engines: detection, normalization, and recognition. Now we will focus on the recognition stage; in Section 4.1 we have described our detection scheme.

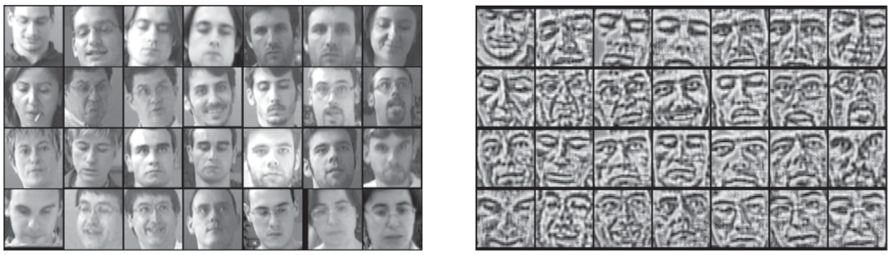
4.2.1. Settings for the face detector

Given a single frame of 576×768 pixels, the image was resized to 288×384 to avoid the effect of interlacing. The set of images, candidates to be a face, is generated as follows: a set of sliding windows of 32×24 pixels has been generated for each frame in such a way that all the possible subwindows from the image are generated. The center of the sliding window is considered every two pixels. The process is repeated at 4 different scales, each time the image is rescaled by a factor of 1.25. Four scales have been shown to be enough for our application, detecting faces up to 64×64 pixels. Detection was performed using the scheme proposed in Section 4.1.1 on each candidate image; 63 000 candidate subimages are generated from each frame. Once the subimage is detected as a face, we take from the original image the square corresponding to the face, and we resize it to a 32×32 image (optimal size for face recognition).

As a face image can appear in more than one sliding window or in multiple scales, a merging step has been added to avoid multiple detections of the same face. We compute the overlap among sliding windows with a detected face in close positions (overlapping more than 80% of their surface). We keep the one that is closer to the mean of the overlapping windows (taking into account the center).

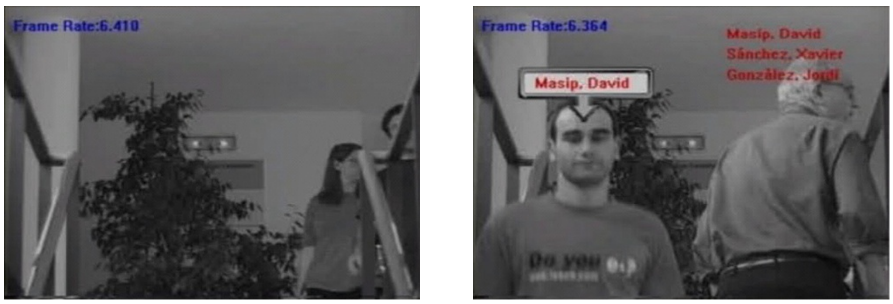
4.2.2. Feature extraction and face recognition

Finally, for the recognition engine, we considered a scheme based on a linear projection using the NDA algorithm



(a) (b)

Figure 6: Example of some faces used in the face recognition experiment, before and after normalization.



(a) (b)

Figure 7: Frames extracted from the human ID at distance application in the computer vision center.

followed by a nearest-neighbor classification. Recognition was performed on the 32×32 faces in the original frame. This is illustrated in Figure 6. In Figure 6a some detected faces from a video sequence are shown. Figure 6b shows these same faces after the normalization.

Given these images, we now detail the feature extraction and classification stages. As a previous step, we have computed a principal component analysis projection matrix using a huge face data set, and the data vectors have been projected to a 224-dimensional subspace that preserves approximately the 97% of the variance, prior to learning the NDA representation. Then 128 NDA components were preserved. Classification was performed using the 5 nearest neighbors' average voting.

All the parameters from this scheme (PCA and NDA dimensionality, number of nearest neighbors, and classifier combination policy) were set by cross-validating the training set.

4.2.3. Experimental setup and results

A Sony EVI D-31 camera was installed looking at a staircase in the Computer Vision Center. This camera was connected to a VCR and four hours of recordings at peek hours were gathered each day every other week, for a total lapse of 6 weeks. The face detector was applied to these videos and the detected images were saved and manually labeled. From the approximately 80 different people detected in all tapes, only those 47 with more than 30 detected faces were included in the gallery. The total number of faces for these 47 subjects was 4176, approximately 88 faces per subject. Ten fold cross-validation on the faces was used to evaluate the performance of our classifier. For this experiment, classification accuracy was 96.83%. We also did a supervised cross-validation, so no samples from the same day were at the same time in training and test sets. Results were very similar, yielding a 95.9% accuracy.

Recognition was also evaluated online, recording the recognition results and video to manually evaluate the online classification accuracy. Recognition rate for approximately 2000 test images belonging to the 47 subjects in the gallery was 92.2%. In this experiment, we also observed that the classifier would greatly benefit from temporal integration which, at the moment, was not implemented. The frame rate of the application with all three working engines and this gallery of the 47 subjects was approximately 15 fps. Also prototype selection techniques applied to the NN classifier could be applied to speed up the system. Figure 7 shows two frames taken directly from the working application. A first frame illustrates the environment in which the experiment took place, and the second frame illustrates the recognizer at work.

5. CONCLUSIONS

In this paper, a real-time face recognition framework has been presented. Two different problems have been solved.

First the problem of the face detection in uncontrolled environments has been solved using a boosting scheme based on the naive Bayes classifier. As it has been shown, this scheme achieves very low false positive ratios in the experimental tests made with two public face databases. Then a face classification scheme based on the NDA feature extraction followed by a nearest-neighbor classification has been performed. The results show very good accuracies in the tests made in a real environment, and also we have achieved a very good performance in terms of time, achieving rates close to 15 frames per second. NDA computational cost is similar to PCA and other linear projection techniques, while the accuracies obtained have been higher in all the tests.

We also show a new approach to boosting. We have introduced the feature learning into the boosting scheme by adapting the features used to the most difficult examples. The algorithm achieves excellent accuracies, but further work is needed to speed up the algorithm in order to be used in a real-time environment.

Another open consideration in the boosting scheme is the fact that we can take benefit from asymmetric classifiers in the boosting process. One desirable property in face detection is not to lose faces in the detection process and this property should be introduced in the internal boosting classifier.

ACKNOWLEDGMENT

This work is supported by MCYT Grant TIC2003-00654 and FP2000-4960, Ministerio de Ciencia y Tecnologia, Spain.

REFERENCES

[1] M. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: a survey," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 1, pp. 34–58, 2002.

[2] C. Kotropoulos and I. Pitas, "Rule-based face detection in frontal views," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '97)*, vol. 4, pp. 2537–2540, Munich, Germany, April 1997.

[3] T. K. Leung, M. C. Burl, and P. Perona, "Finding faces in cluttered scenes using random labeled graph matching," in *Proc. 5th International Conference on Computer Vision (ICCV '95)*, pp. 637–644, Boston, Mass, USA, June 1995.

[4] A. Lanitis, C. Taylor, and T. Cootes, "An Automatic face identification system using flexible appearance models," *Image and Vision Computing*, vol. 13, no. 5, pp. 393–401, 1995, online, available: http://citeseer.ist.psu.edu/article/.lanitis95automatic.html

[5] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 1, pp. 103–108, 1990.

[6] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 696–710, 1997.

[7] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 1, pp. 39–51, 1998.

[8] F. Samaria, *Face recognition using hidden Markov models*, Ph.D. dissertation, University of Cambridge, Cambridge, UK, 1994.

[9] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pp. 130–136, San Juan, Puerto Rico, USA, June 1997.

[10] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: a literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, 2003.

[11] M. S. Bartlett, M. H. Lades, and T. J. Sejnowski, "Independent component representations for face recognition," in *Human Vision and Electronic Imaging III*, vol. 3299 of *Proceedings of SPIE*, pp. 528–539, San Jose, Calif, USA, January 1998.

[12] Z. Li and X. Tang, "Bayesian face recognition using support vector machine and face clustering," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 2, pp. 374–380, Washington, DC, USA, June–July 2004.

[13] L. Wiskott, J.-M. Fellous, N. Kruger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 775–779, 1997.

[14] A. Pentland, B. Moghaddam, and T. Starner, "View-based and modular eigenspaces for face recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pp. 84–91, Seattle, Wash, USA, June 1994.

[15] A. M. Martínez, "Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 748–763, 2002.

[16] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[17] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179–188, 1936.

[18] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 711–720, 1997.

[19] K. Fukunaga and J. Mantock, "Nonparametric discriminant analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 5, no. 6, pp. 671–678, 1983.

[20] P. J. Phillips, P. Grother, R. J. Micheals, D. M. Blackburn, E. Tabassi, and J. M. Bone, "Frvt 2002: Evaluation report," Tech. Rep., March 2003, online, available: http://www.frvt.org/FRVT2002/documents.htm.

[21] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, 2002.

[22] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection of boosted classifiers for rapid object detection," Tech. Rep., Microsoft Research Lab, Intel Labs, Cambridge, UK, 2002.

[23] H. S. Z. Li, Z. Q. Zhang, and H. J. Zhang, "Floatboost learning for classification," in *NIPS 15*, S. Thrun, S. Becker, and K. Obermayer, Eds., Vancouver, Canada, December 2002.

[24] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., pp. 1311–1318, MIT Press, Cambridge, Mass, USA, 2002.

[25] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th International Conference on Machine Learning (ICML '96)*, pp. 148–156, Bari, Italy, July 1996.

[26] M. Skurichina and R. P. W. Duin, "Bagging for linear classifiers," Tech. Rep., Delft University of Technology, Delft, The Netherlands, 2002.

[27] M. Skurichina and R. P. W. Duin, "Bagging, boosting and the random subspace method for linear classifiers," *Pattern Analysis and Applications*, vol. 5, no. 2, pp. 121–135, 2002.

[28] M. Bressan, *Statistical independence for classification of high dimensional data*, Ph.D. dissertation, Universitat Autonoma de Barcelona, Barcelona, Spain, 2003.
[29] A. M. Lopez, F. Lumbreras, J. Serrat, and J. J. Villanueva, "Evaluation of methods for ridge and valley detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 4, pp. 327–335, 1999.
[30] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2001.
[31] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
[32] D. Guillamet, M. Bressan, and J. Vitrià, "A weighted non-negative matrix factorization for local representations," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 1, pp. 942–947, Kauai, Hawaii, USA, December 2001.
[33] P. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall, London, UK, 1982.
[34] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, Boston, Mass, USA, 2nd edition, 1990.
[35] A. M. Martínez and R. Benavente, "The AR face database," Tech. Rep. 24, Computer Vision Center, Barcelona, Spain, 1998.
[36] M. Bressan and J. Vitrià, "Nonparametric discriminant analysis and nearest neighbor classification," *Pattern Recognition Letters*, vol. 24, no. 15, pp. 2743–2749, 2003.
[37] J. Matas, M. Hamouz, K. Jonsson, et al., "Comparison of face verification results on the XM2VTS database," in *Proc. International Conference on Pattern Recognition (ICPR '00)*, vol. 4, pp. 858–863, Barcelona, Spain, September 2000.
[38] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 1, pp. 1063–6919, Kauai, Hawaii, USA, December 2001.
[39] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 1, pp. 23–38, 1998.

David Masip is a Ph.D. student in computer science at the Computer Vision Center (CVC), the Autonomous University of Barcelona (UAB). His research interests concern the development of algorithms of feature extraction for pattern recognition and face classification. He has a degree in computer science and an M.S. degree in computer vision from UAB.

Marco Bressan is a permanent researcher in XRCE's Image Processing Research Group (IPC). He holds a B.A. degree in applied mathematics from the University of Buenos Aires (UBA), an M.S. degree in computer vision from the Computer Vision Centre in Barcelona (CVC), and a Ph.D. degree in computer science and artificial intelligence from the Autonomous University of Barcelona (UAB). As a researcher, he has authored more than 15 refereed publications. His main research interests are in statistical learning and classification; image and video semantic scene understanding; and object detection, recognition, and enhancement, particularly when dealing with uncontrolled environments.

Jordi Vitrià received the Ph.D. degree from the Autonomous University of Barcelona (UAB), Barcelona, Spain, for his work on mathematical morphology in 1990. He joined the Computer Science Department, UAB, where he became an Associate Professor in 1991. His research interests include machine learning, pattern recognition, and visual object recognition. He is the author of more than 40 scientific publications and one book.