Master thesis

Face recognition in controlled environments using multiple images

José Gómez Martil

ETSETB – UPC
March 2009

“I beg your pardon, I didn't recognize you - I've changed a lot.”
“Be yourself; everyone else is already taken.”
-Oscar Wilde-

Agradecimientos

Este trabajo ha sido posible gracias a la paciencia de mucha gente, en especial de mi familia: Paquita, Justo y Raúl. Gracias a Alberto por preocuparse tanto. También quisiera agradecer la atención de la gente del D5, Ramón, Verónica, Carlos, Albert, Joan, Jaume y todos los que me ayudaron puntualmente o con los que simplemente pasamos buenos ratos. Y gracias a Ferran por aguantar hasta el final y conseguir que este proyecto salga a la luz. Finalmente agradecer el esfuerzo del lector, ya sea por obligación, curiosidad o interés, del que espero que encuentre este trabajo comprensible y útil.

Table of contents

List of figures

List of tables

1 Introduction

Face recognition has become one of the most studied applications in various fields such as biometrics, image processing, pattern recognition or computer vision during the past years. One of the reasons for this development has been the increasing need of security applications.

This interest in face recognition, and biometric systems in general, can be seen in the actions of three main parts. Firstly it is shown in the research community, for example with approximately 140 articles published about face recognition and its related tools among the different IEEE and Elsevier journals in 2007 and about 180 in 2008. Besides, about 770 articles can be found in the IEEE Conference Proceedings in 2007 and nearly 800 in 2008, including specific conferences as the IEEE International Conference on Automatic Face and Gesture Recognition (IEEE FG 2008). Secondly, it is shown in the development of different commercial solutions and the growth of the biometric market, with companies like L-1 Identity Solutions, with a market capitalization of US$1270 million in September 2008 (about US$425 million in February 2009). Finally, it is shown on the side of the clients, which require reliable products based on face recognition. Some of the main customers are different government agencies interested in security applications. For that reason, there are some initiatives to try independent evaluations of prototype face recognition algorithms and also commercial systems, like the Face Recognition Vendor Test (FRVT) 2006 [1], the last of a series of evaluations like FERET or FRVT 2000 and 2002. These last evaluations were conducted by the U.S. National Institute of Standards and Technology (NIST).

The use of different biometric systems is becoming usual in our society. In their objective of determining the identity of one person, several characteristics can be analyzed. For instance, there are physical features as fingerprints, iris, retina, face, hand geometry, hand veins, odour or DNA and psychological features as gait or signature. Every technique has its own peculiarities, but they can be compared following the next criteria [2] as we can see in Table 1:

- Universality, which indicates how commonly the characteristic is found in every person.
- Distinctiveness, which reflects if the characteristic is different enough among the different people.
- Permanence, which refers to the invariance in time of the characteristic.
- Collectability, if the characteristic can be easily acquired and measured.
- Performance, which shows the accuracy, speed, and cost (resources) required.
- Acceptability, which indicates how people are prepared to accept the use of that technique.
- Circumvention, which reflects how the system behaves if someone is trying to cheat on it.

	Universality	Distinctiveness	Permanence	Collectability	Performance	Acceptability	Circumvention
DNA	H	H	H	L	H	L	L
Ear	M	M	H	M	M	H	M
Face	H	L	M	H	L	H	H
Facial thermogram	H	H	L	H	M	H	L
Fingerprint	M	H	H	M	H	M	M
Gait	M	L	L	H	L	H	M
Hand geometry	M	M	M	H	M	M	M
Hand vein	M	M	M	M	M	M	L
Iris	H	H	H	M	H	L	L
Keystroke	L	L	L	M	L	M	M
Odour	H	H	H	M	H	L	L
Palmprint	M	H	H	M	H	M	M
Retina	H	H	M	L	H	L	L
Signature	L	L	L	H	L	H	H
Voice	M	L	L	M	L	H	H

Table 1. Comparison of various biometric systems (high, medium and low are denoted by H, M and L respectively)

Since it is a natural act for human beings, there has been a strong motivation to create reliable face recognition systems. However, as we can see in Table 1, using the face as a biometric identifier has its advantages and disadvantages. But, although we can find problems with its distinctiveness, permanence or circumvention, its collectability and acceptation make this type of recognition especially suitable for non-intrusive applications; that is, applications that do not require a direct collaboration from the observed people.

For that reason, there are many initiatives to use face recognition systems for security applications in airports, borders or crowd control. Moreover, we can consider other uses like the intelligent interactions in the known as *smart* or *controlled environments*. One example of this was the CHIL (Computers in the Human Interaction Loop) project. This was a European project with the objective of creating environments in which computers serve humans who interact with other humans in an indirect and unobtrusive way; that is, without having to attend to the machines themselves [3]. To perform different experiments about the project, UPC built a “smart room”; that is, a meeting room which included different sensors and a set of subsystems to perform a wide range of operations, including face recognition. This is an example of a controlled environment where we can test our solutions more efficiently than in a real setting, as we can control many of the main problems of this kind of systems, like pose variation or lighting.

In any case, face recognition can be used in many ambits in addition to the typical security or surveillance applications, like access control in mobile phones, computers or ATMs, physical access control through smart doors, prevention of voter fraud or photo labelling, like recently introduced in Google’s Picasa, Apple’s iPhoto or Microsoft’s Windows Live Photo Gallery.

However, although in some applications the face detection part is considered together with the recognition part, we will not consider it. Even though it is a very important stage of the whole application, it is enough complex and differentiated to be considered independently. For that reason we will focus only on face recognition.

So we have to design a part of a broad system considering that there is a face detection system previous to the face recognition. As a result of this, the input data to our system are a set of faces from different people, possibly with pose, light and size variation, and previously labelled from a tracking system.

With these premises in mind, we can see that we are dealing with a pattern recognition problem; that is, we will try to assign a label to the given elements, which are described with a set of features or characteristics [4] [5]. There are two main categories for pattern recognition problems: supervised and unsupervised learning. In unsupervised learning, the inputs are not labelled and the system has to find the intrinsic structure of the data, whereas in supervised learning the inputs have some preassigned labels, as in our case. Therefore, we can analyze our system dividing it into the different parts of a classic unsupervised learning problem; that is, data collection, feature selection, selection of a representation model, classifying, training and testing. In Fig. 1 we can see a block diagram of these different stages and how they will be translated to our problem, as we will see next.

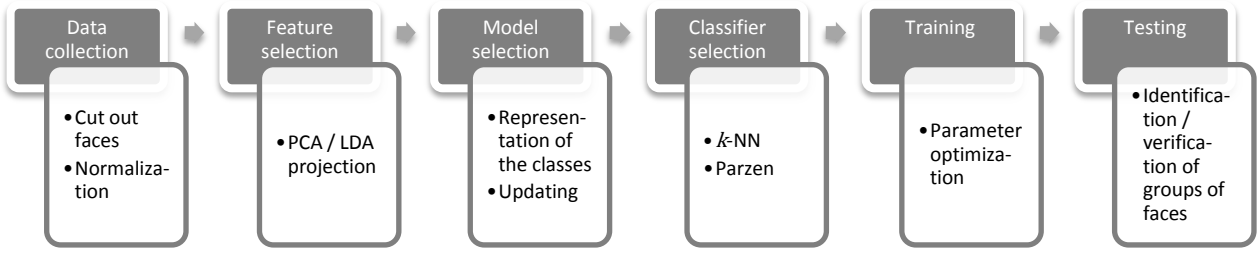


Fig. 1. Design stages of a classical pattern recognition system and its application to our face recognition problem

Our objective is the design of a face recognition system for its use in controlled environments. For that reason, different tasks will be performed, that chronologically can be stated as:

- The study of the basis of pattern recognition, and especially the statistical pattern recognition.
- The study of the main feature selection techniques, focusing in principal component analysis (PCA), and the basic classifiers, as the nearest neighbour and Parzen classifiers.
- The study of the main combinations techniques, as we will try to take advantage of the availability of sets of faces to make better-founded decisions
- The evaluation of the main problems, like pose variation or light changes.
- The choice and posterior adaptation of these different general tools to our needs, like the use of modifications of the Parzen classifier.
- The design of the main algorithm, focusing on the differences between the two working modes, identifying (when we try to recognize a person among the different people in our

system) and verifying (when we know a priori the *id* of the person and we try to assure its identity), and the use of single images or sets of faces.

- The implementation of the algorithms in C, trying to become familiar with the development environment and considering the possible optimization but maintaining the clarity of the code.
- The testing of the whole application using faces from different databases and from real data; that is, captured in real scenarios and in non optimal conditions.

In principle, the study and evaluation of different basic techniques was the starting point of this project, so there are not many innovations. However, this work was taken as a base work, to study properly these fundamental tools for posterior developments and also as a way to get base results for later comparisons. As a consequence of this, there were no specific previous requirements to meet, with regard to accomplish determined recognition rates, but we will try to get the best results in performance and resources. Besides, there were other objectives to consider, as the development of the software as a modular architecture using the UPC's SoftImage platform, which contains different libraries with many image processing tools. This environment is continuously updated with new contributions from people of the UPC Image Processing Group, so some parts of this work will be included in it.

This master thesis is organized as follows. In section 2 we will see a brief survey on pattern recognition, focusing on the statistical approach, whereas in section 3 we will analyze the main face recognition systems. Next, in section 4, we will detail some of the main techniques used in face recognition systems, and in section 5 we will analyze the characteristics of our problem and the solutions applied in it. In section 6, we will test the different algorithms explained in section 5, and in section 7 we will see the conclusions and future work. Finally we will find a section with the main references used in this work and an appendix with some implementation remarks.

2 Pattern recognition

Pattern recognition [4] -the act of taking in raw data and making an action based on the "category" of the pattern- is as natural for human beings, for example recognizing faces, voices or characters, that only when it is tried to be applied in an automatic system the difficulty of the process becomes patent.

Basically our objective is matching every pattern with the class it belongs to. A pattern is the information that characterizes and differentiates the element to recognize. The patterns should be extracted from the same data source, for example we cannot compare voice with face patterns, but we can fuse them to generate new patterns. Although they are slightly different concepts, in this work we will refer to the patterns as samples, features, characteristics or vectors. A class refers to a group of patterns with similar characteristics among them and different with the patterns of the other classes. In this work we will talk about classes or categories.

A pattern classification system can be divided in different stages. Previously we have seen the design steps in Fig. 1 and now in Fig. 2 we can see a general approach to the working stages.

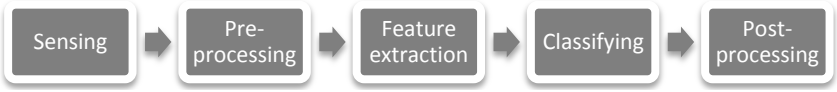


Fig. 2. Stages of a general pattern classification system

- Sensing: it consists in the acquisition of the pattern to analyze using some transducer (cameras, microphones, etc), to have a representation in a digital format.
- Pre-processing: at this stage different tasks can be performed, like the segmentation of the object to study from the background, noise removal, normalizations, etc.
- Feature extraction/selection: it consists in a stage where the data are reduced representing them with different qualitative or quantitative features or characteristics. Although most of the times feature selection and feature extraction are considered equivalent, extraction refers to creating new features using combinations or transformations of the existing feature set, whereas selection refers to selecting a subset of the available features without transformations.
- Classifying: at this stage the features are analyzed to determine to what class or category the object belongs. The features should discriminate among the different classes to recognize and be similar to the objects of the same class.
- Post-processing: this part evaluates the cost of the decision of the classifier and tries to reduce the risk of errors using different tools, basically from the context of our observations.

In real applications, these stages may not be so differentiated, merging different functions in some stages. Furthermore, the working mode may not be only unidirectional as posterior stages can report different results to adapt the behaviour of earlier sections.

Pattern classification can be divided into two main categories as we have commented in the introduction: supervised and unsupervised learning. In unsupervised learning the inputs are

not previously labelled and the system has to find the intrinsic structure of the data; that is, establish the organization of the classes based on the properties of the data. On the other hand, in supervised learning the inputs have some preassigned labels, as in our case, so there are some available patterns previously classified to use as a training set.

Then, according to the nature of the features, we can define two main approaches, the syntactic and the statistical classification. Syntactic (or structural) pattern recognition is based on the characterization of the inherent structure of the qualitative features. For that reason, the complex patterns can be decomposed using a hierarchical structure in simple subpatterns, until every pattern can be represented with the interrelationships among the simplest subpatterns called primitives. This approach can be explained with an analogy with the syntax of a language [6], where we can consider the complex pattern as the sentences, the primitives as the alphabet and the relations among the primitives as the grammar. Then, every complex pattern may be described with a limited number of primitives and their relations. In contrast, in the statistical pattern recognition approach the patterns are characterised using the statistics of the quantitative features. As this is the type of solution used in our case, we will comment it with more detail next.

2.1 Statistical pattern recognition

In statistical pattern recognition every pattern is represented by a set of d features arranged as a d-dimensional vector; that is, $\boldsymbol{x} = (x_1, x_2 \dots x_d)^T$, and considered as a point in a d-dimensional space, $\boldsymbol{x} \in \mathbb{R}^d$. This space is called the feature space, where each axis corresponds to one of the features. Besides, we expect that the vectors of the same class will form a relatively compact cluster and sufficiently separated from the other classes. Each one of these areas in our feature space can be described by a probability density function (or mass if the features are discrete) for every class, which can be obtained from a set of training samples. Then, every pattern vector $\boldsymbol{x}$ from the class ω_i (one of the c classes $\omega_1, \omega_2, \dots, \omega_c$) is viewed as an observation obtained randomly from the class-conditional probability density function $p(\boldsymbol{x}|\omega_i)$. Now, depending on the available information about these functions, we can think about different approaches to design a classifier. These approaches are summarized in Fig. 3 [6], where from left to right and top to bottom we can see the possible solutions when the a priori knowledge about the problem decreases.

As we can see, if all the class-conditional probability density functions are known, then the optimal Bayes decision rule can be used to design the classifier. However, as these functions are not usually available, they should be estimated from the existing data. Depending if the form of the density functions is known but some of the parameters are not, a parametric solution can be used. In contrast, if the form is not known, we should use a nonparametric technique. This latter kind of solutions represents the more general approach for a supervised learning problem, and seems suitable for our problem. For that reason we can try to estimate the density functions (for example using the Parzen window approach) or try to directly construct the decision boundaries (for example, with the k-nearest neighbour approach). These solutions are examples of another type of division, the geometric approach where the decision boundaries are obtained from the training data, and the probabilistic density-based approach, where the density functions are estimated and then the decision boundaries are set using some discriminant functions; that is, the functions that determine if

a sample belongs to a class or another. In any case, under certain circumstances the two approaches are equivalent as we will see later.

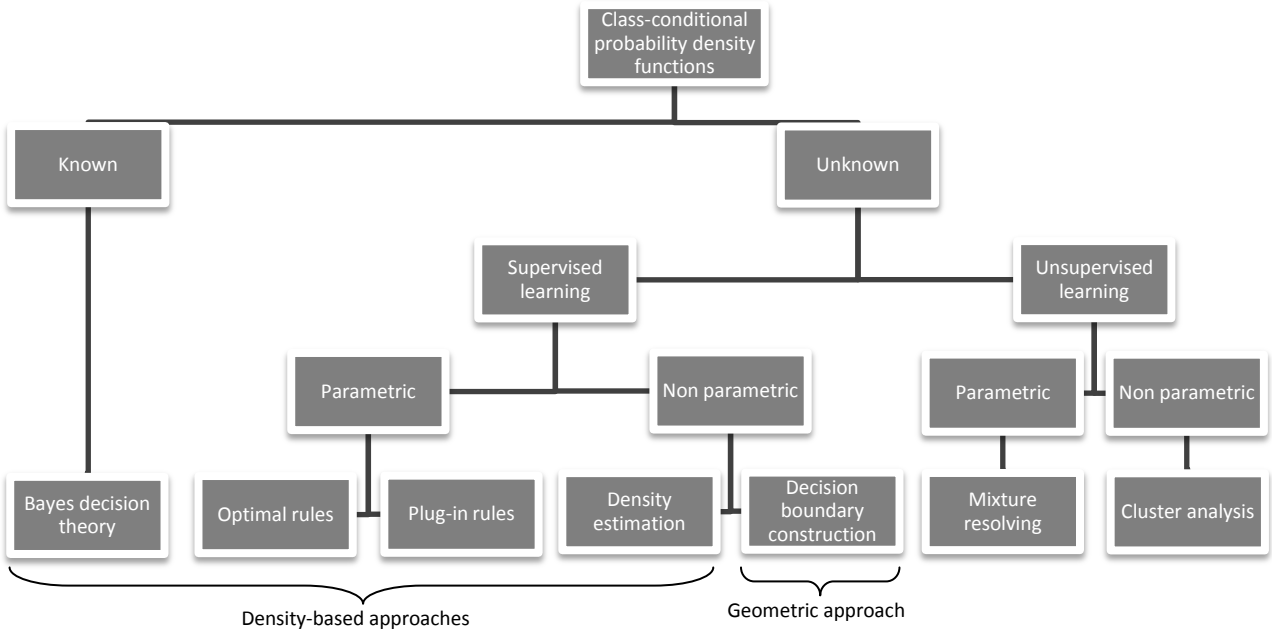


Fig. 3. Approaches in statistical pattern recognition

In the next subsections the main techniques used in feature selection and classification, the two main stages of a statistical pattern recognition problem, are summarized. Besides, some of the general problems associated to the general process will be commented.

2.2 Feature extraction

Feature extraction is one of the most challenging parts of our statistical pattern classification system. Our goal is to find the set of numerical features that best represent our data, because if the features are well representative and discriminative among the classes, the work of the classifier becomes easier. Furthermore, it is desirable to these features to be invariant to translation, rotation and size changes of the objects to classify, although these issues can be treated in the pre-processing stage as well. For that reason, the selection of the features is a critical point, mainly because it depends totally on the characteristics of the patterns to recognize, whereas the classifiers are more standardized tools.

Another of the objectives of feature extraction is dimensionality reduction. With this reduction we are looking for the simplification of the data representation and consequently making the work of the classifier easier. Besides, we have to take care of not decreasing the discrimination power of the features if we reduce too much the data.

Considering these premises, feature extraction tries to transform, linearly or non-linearly, the vectors $\boldsymbol{x}$ in the original d-dimensional space ($\boldsymbol{x} \in \mathbb{R}^d$) into an m-dimensional subspace, with $m \leq d$. Among the linear techniques, the most known and used ones are [4] [6] principal component analysis (PCA), linear discriminant analysis (LDA) and independent

component analysis (ICA). PCA, also known as the Karhunen-Loève transform (KLT), is a linear transformation that permits representing one pattern with the combination of a set of significant eigenvectors (those with the highest eigenvalues) obtained from the covariance matrix of the set of patterns. ICA is similar to PCA except that it is designed to consider the distribution of the components as non-Gaussian. Finally, LDA, also known as Fisher's linear discriminant (FLD), uses the class information of the samples to try to maximize the interclass separation while minimizes the intraclass variation.

About the non-linear transformations, it is common to use the kernel trick. The basic idea of this method is to map the original non-linear observations into a higher-dimensional space, and then use a linear transformation. The most known application is the Kernel PCA (KPCA), but we can find more non-linear versions of various linear techniques. Furthermore, we can find other tools as multidimensional scaling (MDS) or neural networks as the self-organizing map (SOM) or Kohonen map (see [4] [6] [7] for an overview about all these techniques).

2.3 Classification

The choice of the classifier is the next challenging stage of our system. As we have commented previously, the election of the classifier is not as dependant of the application like feature selection, and in many cases it is done by the availability of the algorithm. We can find three main approaches in the design of the classifiers depending on the concept they are based on: similarity, geometry or probability (see [4] [6] [7] for an overview about all these techniques). In any case, there are other possible taxonomies.

The first type of classifiers is based on similarity; that is, similar patterns should correspond to the same class. One of the possible techniques is template matching; that is, the pattern to be recognized is matched against the stored templates and then a similarity measure like a correlation is estimated. Another option is the use of a minimum distance classifier, where we should select a metric (e.g. Euclidean) and a prototype for the class, for example the mean of the samples of every class. Then the class with a minimum distance between the sample and the prototype is chosen.

The second type is based on geometry. These classifiers try to estimate the decision boundaries; that is, the hyperplanes that separates the classes, directly by optimizing some error criteria. Examples of this approach are support vector machines (SVM) or neural networks as the single-layer perceptron.

Finally, we have the probabilistic approach. Depending on the available information, we have different solutions as we could see in Fig. 3. If we do not have any information, we can only make a decision based on the a priori class probability $p(\omega_i)$, obtained from a large enough number of random samples, and then choose the class with the higher probability,

$$p(\omega_i) > p(\omega_j) \quad 1 \leq i, j \leq c \,, i \neq j$$

Anyway, our objective is to obtain the posterior class conditional densities $p(\omega_i|\boldsymbol{x})$; that is, the probability that the sample $\boldsymbol{x}$ belongs to the class ω_i. The way to obtain these densities tells the difference among the techniques, but once they are obtained the Bayes decision

rule is applied to get the minimum error,

$$p(\omega_i|\boldsymbol{x}) > p\big(\omega_j|\boldsymbol{x}\big) \quad 1 \leq i,j \leq c\,, i \neq j\,, \boldsymbol{x} \in \omega_i$$

Besides, in most of the cases we will only be able to estimate the class conditional densities $p(\boldsymbol{x}|\omega_i)$ but using the Bayes' theorem we can get the posterior probabilities,

$$p(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)p(\omega_i)}{p(\boldsymbol{x})}$$

and as the denominator does not depend on the class, we can simplify the expression to obtain the known as the minimum error classifier,

$$p(\boldsymbol{x}|\omega_i)p(\omega_i) > p\big(\boldsymbol{x}|\omega_j\big)p(\omega_j) \quad 1 \leq i,j \leq c\,, i \neq j\,, \boldsymbol{x} \in \omega_i$$

We can see that an optimal classifier can be designed if we have the prior probabilities $p(\omega_i)$ and the class conditional densities $p(\boldsymbol{x}|\omega_i)$. However, in a real application these probabilities are not available and the classification should be done using only some general knowledge about the problem and the training samples. If we know a parametric form of the class conditional densities, the work is reduced to find the parameters that characterize the distribution (for example, a Gaussian with its mean and variance). Two methods are commonly used [4], the maximum likelihood method that looks for the parameters that are best supported by the training data, and the Bayesian estimation, where the parameters are considered random variables with a known a priori density that the training data converts into a posteriori.

The multivariate Gaussians distributions are commonly used in the parametric estimations. Depending on the assumptions about the covariance matrices we can find different solutions, if these matrices are considered equal for every class, the Bayes plug-in rule (with plug-in we are referring to substitute the real probabilities with the estimations obtained from the data) provides a linear classifier. If they are different, a quadratic classifier is obtained.

About the non parametric solutions we can find two main approaches, the k-nearest neighbour (k-NN) classifier and the Parzen classifier. The k-NN classifier is based on the idea of finding the classes of the k-nearest neighbours vectors (based on a distance metric, usually the Euclidean) of the vector to classify. After that, it calculates which the most found class is on that k neighbours and assigns that class to the test sample. With Parzen, a window function is imposed on every training sample to get a density estimate of each of the classes (that is why is also known as Parzen window method or kernel density estimation). It is common to use a Gaussian function (another types of functions can be used, as uniform or triangular ones). Then the classifier assigns the class which gives the highest probability for that test pattern based on the density functions of all the classes computed during training.

In addition to the classifiers from these three groups, we can find other classifiers like decision trees (such as the commonly used CART and C4.5) and different neural networks solutions.

Classifiers can also be seen from another point of view using the discriminant functions. So a classifier can be determined with a set of c (one for every class) discriminant functions $(g_i(\boldsymbol{x})\colon \mathbb{R}^n \to \mathbb{R}\,, 1 \leq i \leq c)$ that provides a score that the sample $\boldsymbol{x}$ belongs to every class, and typically $\boldsymbol{x}$ is labelled as a member of the class with the highest score. So the discriminant functions partition the $\mathbb{R}^n$ space into c decision regions or classification regions, which are delimited by the classification boundaries.

With this approach we can use estimations of the posterior probability densities as discriminant functions to obtain the Bayes decision rules: $g_i(\boldsymbol{x}) = p(\boldsymbol{x}|\omega_i)p(\omega_i)$. Besides, we can use linear or quadratic functions, use kernel functions to obtain the Parzen classifier and distance based functions to obtain the k-NN method.

2.4 Known problems

One of the most common problems in pattern recognition systems is the known as "curse of dimensionality" [7]. It is common to represent our data in a high dimensional space (for example an image can be represented as a point in a space with dimension the size of the image), and we could think that the high dimensionality of the data only would affect on the time and resources we will have to use, but there are more subjects to deal with. In summary, the problem is that as we increase the number of dimensions of our feature space the volume of the space increases exponentially and for that reason, the potential complexity of the probability density functions is higher. It has been seen that, for a fixed amount of training data, the performance of the system improves when the number of features grows until a certain point when the performance decreases. This effect is closely related to the small sample size problem, which appears when the number of available samples, which is usually limited, is low compared to the dimension of the space. For that reason, the ratio between the number of training samples and the dimension of the space has to be high enough to assure that our samples can be well represented (it is generally accepted to use a ratio higher than 10 when possible). As we can suppose, the dimensionality reduction made at the feature extraction stage will be an important process in our system to reduce this effect.

Another of the significant problems is the choice of an appropriate model for the representation of our classes. This model should not be confused with the features themselves. For example a model for a class can be the centroid of the different elements of the same class.

Finally, we can find various issues about the generalization in classifiers; that is, the performance of the classifier with new samples. These results can be low due to the curse of dimensionality, because the number of parameters is too high, or because the classifier has been too optimized for the training samples and not for the new ones, which is known as overtraining.

3 Face recognition

In this section we are going to summarize some of the main techniques used in face recognition. Although this can be considered as a pattern recognition problem, in the literature it is common to find that the separation among the different parts seen before is not as definite and the vision of the systems is more global.

Face recognition systems can be divided into two main groups depending on the nature of the input data, still images or video. The use of still images is the most frequent, while the use of video is not as developed, as in many occasions the still image methods are extrapolated and the characteristics of the video are not considered, as we will see later.

3.1 Face recognition from still images

There is a great amount of face recognition systems and most of them use a mixture of techniques. For that reason, it is difficult to make a clear separation in groups. However, it is common to use a classification based on the type of features as an analogy of the psychology behind humans to recognize faces [8]: using a holistic, a feature-based or a hybrid approach. The holistic methods use the whole face as the input data to our system whereas in the feature-based methods some local features (e.g. eyes, mouth or nose) and their locations or statistics are used to classify the faces. In the hybrid approach, both characteristics are used.

There have been several studies about the psychophysics and neurological aspects of human face recognition. For example, about considering face recognition as a dedicated process, which seems true, or about face perception as a holistic or feature-based process. The studies suggest that both data are used, first considering a global idea of the face which is refined with some features, especially if they are not usual, as big ears or a hooked nose.

3.1.1 Holistic methods

About the holistic methods, the most used ones are those based on the linear face extraction techniques, like PCA, LDA or ICA, and classifiers from the nearest neighbour or nearest mean type. However, there is a large number of possible variations, so we show some of the techniques used as the basis for most of them.

PCA represents one face with the combination of a set of significant eigenvectors obtained from the covariance matrix of the training faces. Therefore, we can represent a high dimension vector (which represents the luminance of the pixels of the face as a column vector) in a low dimensional space, known as the face space or eigenspace [9], where we expect that the vectors of the same person form a class as a compact cluster. Then it is common to use a classifier based on the distance (normally Euclidean) of the vector to analyze to the mean vector of every class.

Similar approaches are proposed using LDA or ICA. LDA [10] tries to maximize the distance among the different classes, not only focusing on the best representation of the faces. A variation of the technique proposed in [11] is commonly used, where firstly PCA is used to

obtain a dimensionality reduction and then the LDA projection is made. Finally ICA is similar in concept to PCA, but it is more appropriate for non-Gaussian distributions as it do not rely on the second order property of the data.

The intrapersonal/extrapersonal Bayesian classifier tackles the problem from a different point of view. Now the space generated by the subtraction of two images is considered instead of the face space. For that reason, every difference of faces can only be considered in one of these two classes: the intrapersonal class, generated by images of the same person, or the extrapersonal, generated by the difference of images from different people. Furthermore, we suppose that these two classes follow Gaussian distributions [12] [13]. Then, the class conditional probabilities are estimated and a similarity measure is made, following either the MAP (maximum a posteriori) or ML (maximum likelihood) rule, to classify the images.

Finally other systems are suggested, like the combination of PCA and SVM or neural networks solutions as the probabilistic decision based neural networks (PDBNN).

3.1.2 Feature-based methods

Concerning the feature-based methods, we can find techniques based on the geometry of the faces, for example, using the distance between the eyes, the distance from the eyes to the mouth, etc.

Other solutions use hidden Markov models (HMM). An HMM is a statistical model in which the system is considered as a Markov process with unknown parameters, and the goal is to estimate the hidden parameters from the known ones, and use these parameters for the pattern recognition task. These models use strips of pixels from the face or its KL projection coefficients for data representation.

Finally we can find the elastic bunch graph matching (EBGM), where the local representation is made with the wavelet (usually Gabor) coefficients for different scales and rotations based on fixed bases (called jets [14]).

3.1.3 Hybrid methods

The hybrid methods take advantage of both global and local features. For example, we can find the application of PCA to the local features to obtain eigenfeatures [15] such as eigeneyes or eigenmouth, in addition to the eigenface. With these new elements the recognition results are improved, especially when there are occlusions. Other solutions use LFA (local feature analysis), which constructs kernels detecting local structures of a face, combining it with PCA. Finally we can find different methods based on 3D models, which are especially suitable to minimize the effect of pose but are considerably more complex than the previous solutions.

3.2 Face recognition from video

A priori, face recognition from video is preferable over using still images since we have more available information as the temporal variation. However, although using video is common in surveillance and access control applications, this type of techniques have not been as developed until recently as the recognition from still images. This is due to some of the characteristics of the frequently used videos, like their low quality, as it is common to use low-end cameras; the low quality of the faces, in terms of their small size and also variations in pose and lighting; and finally because a great amount of resources are needed, specially storage capacity. In any case, motion information is widely used in face detection and face tracking.

We can find three main groups of face recognition systems based on video information. The first group is formed with algorithms based on the extrapolation of the techniques used in still images to video face recognition. This kind of techniques is the most extended, and is based on two main ideas: the election of frames where the face is in better conditions according to pose and lighting (as there are many available frames, the rest are discarded for recognition tasks); and the use of some type of voting or combination among the results obtained from a continuous recognition, that as we will see next is the approach used in our system. In the second group we can find the systems that use both the spatial and the temporal information, for example tracking some facial features. Finally we have the multimodal approach, where different inputs are used in addition to face, like gait or voice.

Besides this classification, we can consider other two types of systems based on the nature of the training data. So we can find still-to-video face recognition when there are still images for the training, and video-to-video when the gallery (the training data that form our models) and the probe data are videos.

Now we can see some of the techniques proposed in the literature. In [16] a probabilistic framework is proposed for both the still-to-video and video-to-video cases. In [17] a video-to-video system is proposed which models the faces as a linear dynamical system using an autoregressive and moving average (ARMA) model. Then it uses the concept of subspace angle to estimate the distance between the probe and gallery video. In [18] a probabilistic algorithm that learns from reduced sets of images to recognize later in videos is used. Furthermore, to try to reduce the effect of pose variation and the occlusion, it uses a division of the face images into six zones which are characterized by a mixture of Gaussians. Finally, in [19] a solution based on the use adaptive Hidden Markov Models (HMM) is proposed. During the training process, the statistics of training video sequences of each subject, and the temporal dynamics, are learned by an HMM. During the recognition process, the temporal characteristics of the test video sequence are analyzed over time by its corresponding HMM. Then the likelihood scores provided by the HMMs are compared, and the highest score provides the identity of the test video sequence.

3.3 Known problems

Most of the techniques that we have seen perform well in controlled situations, but their behaviour in an uncontrolled environment can be different. As we have introduced

previously, there are two main issues in face recognition in real applications: the pose variation problem and the illumination variation problem.

It is usual to test face recognition systems with frontal and uniformly illuminated images from different databases, but when there are significant changes in pose and lighting conditions the performance decreases in general. Besides, other common problems are expression changes, the changes caused by aging and the facial hair or occlusions, for example due to objects like glasses, scarves, etc.

The change introduced by illumination can be even greater than the difference between two individuals, so the systems that have to deal with these circumstances should introduce some techniques to minimize its effect. In [8], the main techniques are grouped into four approaches. Firstly we can find the heuristic methods, where different tools are used as the pre-processing of the faces with contrast normalization and histogram equalizations. In addition to this, one of the most used techniques in PCA approaches is to discard the most significant components (usually the first three) as they mostly represent the illumination changes, as verified in [11]. Secondly we have the image comparison approach, which tries to find a proper representation for the faces; that is, features as invariant to lighting changes as possible, and classifiers that try to minimize this effect too. Next we can find the class-based approach, where the light source is modelled as a Lambertian surface (that is, a surface that reflects light isotropically), and uses simple images or sets of images under different illumination directions to generate normalized faces. Finally there are various techniques that represent the faces using 3D models to synthesize virtual normalized faces, as in the last approach. These methods are more accurate but have higher computational requirements.

Regarding pose variation, similar approaches to the previous ones can be found in the literature. For example, if there is a training set with controlled rotated faces (e.g. profile images, at 45 degrees, etc.) the eigenface approach can be extended to create view-based eigenspaces as proposed in [15]. Besides, we can find solutions that try to generate frontal faces using 2D transformations or using 3D models, which can also be used to minimize the illumination effects.

We can anticipate that according to the characteristics of our application we will not take into account the illumination problem, as we are dealing with controlled environments which are mostly illuminated with uniform artificial lights. However, due to its simplicity we will test the effect of discarding the first eigenvectors in PCA when we will evaluate the different parameters of the system. Moreover, with regards to the pose variation problem, we will use a view-based approach. In any case, we will detail these aspects next, when we will analyze the characteristics of our models.

4 Description of the main techniques

In this section we will discuss some of the main techniques which seem suitable to be used in our problem. These are known tools used in the state-of-the-art face recognition technology. We have introduced most of them previously when we have seen the parts of the pattern recognition problem and now we are going to develop them, focusing on feature selection and on the classifiers. Finally, in the next section when we will see the general architecture of our system, we will comment what techniques have been used in our system and the reasons for their choice.

4.1 Feature selection

In this point we are going to comment the main techniques that we can find in the literature for feature selection: PCA (principal component analysis), LDA (linear discriminant analysis), also known as FLD (Fisher linear discriminant), and the Bayesian intrapersonal/extrapersonal. All these techniques are based on the idea of image space; that is, the application of the feature space concept from statistical pattern recognition previously commented in section 2.1, where the input data are images. So any image with width w and height h can be represented as a point in a d-dimensional image space ($\mathbb{R}^d, d = w \times h$). Therefore, applying it to our case, a face can be considered as a column vector $\boldsymbol{x}$ (every element of the vector is the intensity value of the pixel, which are read from the face by lines and put into the column vector) and consequently as a point which belongs to that image space ($\boldsymbol{x} \in \mathbb{R}^d$). Then we expect that these high dimension vectors represent properly the faces; that is, that the faces of the same person form relatively compact clusters, so we can apply some classification techniques later. Anyway it is common to try to reduce this high dimension previously, to make the task of the classifier easier.

4.1.1 Principal component analysis (PCA)

PCA (principal component analysis, also known as the Karhunen-Loève transform (KLT)) is a technique that obtains the dimensionality reduction of the data, the faces in our case, extracting their main features or components. PCA is a linear transformation that permits representing one image in a low dimensional space, known in our application as face space or eigenspace. So the first component corresponds to the lineal combination of the data with maximum variance, and the nth component corresponds to another lineal combination with maximum variance but orthogonal to the previous $n-1$. To perform the PCA analysis we need to know the statistics of our samples, so the number of these samples should be higher than the dimension of the samples to perform a correct analysis [9].

So, we start with a set of M training images $\boldsymbol{x}_1, \dots, \boldsymbol{x}_M$ of dimension N (again, every image is arranged as a column vector of size $N = w \times h$, where w and h are the width and the height of the image respectively). These vectors can be grouped in a $N \times M$ matrix $X = (\boldsymbol{x}_1 \dots \boldsymbol{x}_M)$. Then we can find the average image as $\overline{\boldsymbol{x}} = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{x}_i$.

The main objective of PCA is to find a set of vectors with the biggest projection over each one of the images; that is, the vectors that best distribute the face images in the image

space. In other words, we are looking for the orthogonal vectors that best represents the samples $\boldsymbol{x}_i$ minimizing the mean error between $\boldsymbol{x}_i$ and $\overline{\boldsymbol{x}}$, and these vectors are the eigenvectors $\boldsymbol{\varphi}_i$, with their associated eigenvalues λ_i, of the covariance matrix C of the samples, $C = WW^T$ where $W = (\boldsymbol{\omega}_1, \boldsymbol{\omega}_2 \cdots \boldsymbol{\omega}_M)$ and $\boldsymbol{\omega}_i = \boldsymbol{x}_i - \overline{\boldsymbol{x}}$; that is,

$$C = WW^T = \sum_{i=1}^{M} (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$$

Therefore, C will be an $N \times N$ matrix, and for instance, with an image of 128×128 pixels, we will obtain a 16384×16384 matrix, clearly too high to be easily manipulated. Nevertheless, it is known by algebra that we could estimate λ_i and $\boldsymbol{\varphi}_i$, estimating the eigenvalues μ_i and eigenvectors $\boldsymbol{\psi}_i$ of the $M \times M$ matrix $C' = W^T W$ [9]:

$$C'\boldsymbol{\psi}_i = \mu_i \boldsymbol{\psi}_i$$
$$W^T W \boldsymbol{\psi}_i = \mu_i \boldsymbol{\psi}_i$$
$$W(W^T W \boldsymbol{\psi}_i) = \mu_i (W \boldsymbol{\psi}_i)$$
$$C(W \boldsymbol{\psi}_i) = \mu_i (W \boldsymbol{\psi}_i)$$

So, we can obtain $M - 1$ eigenvectors $\boldsymbol{\varphi}_i$ with their associated eigenvalues λ_i, from $W\boldsymbol{\psi}_i$ (normalizing them) and μ_i, respectively; that is,

$$\boldsymbol{\varphi}_i = W \boldsymbol{\psi}_i$$
$$\lambda_i = \mu_i$$

Then, the eigenvectors are put in decreasing order (according to their eigenvalue) and only L of the first M eigenvectors are kept to form the projection matrix $\phi = (\boldsymbol{\varphi}_1 \boldsymbol{\varphi}_2 \cdots \boldsymbol{\varphi}_L)$. Besides, these eigenvectors should be normalized to unit energy previously ($\overline{\boldsymbol{\varphi}}_i = \boldsymbol{\varphi}_i / |\boldsymbol{\varphi}_i|$). These eigenvectors are selected as they represent the directions of maximum variance; that is, the principal components.

When we apply these tools to faces we can talk about eigenfaces when we are referring to the eigenvectors, and the projection space is called face space. So every face can be represented as a linear combination of eigenfaces, which at the same time are a linear combination of the original images. In Fig. 4 we can see examples of eigenfaces. At the top we can see the mean image at the left, followed by the first five eigenfaces, whereas at the bottom we can see the eigenfaces corresponding to the eigenvectors with lower eigenvalues, which are clearly less discriminant than the first ones, and at first glance, do not seem to capture any relevant information, mostly noise, so that is the reason why they are commonly discarded without a significant loss for the representation of the faces.

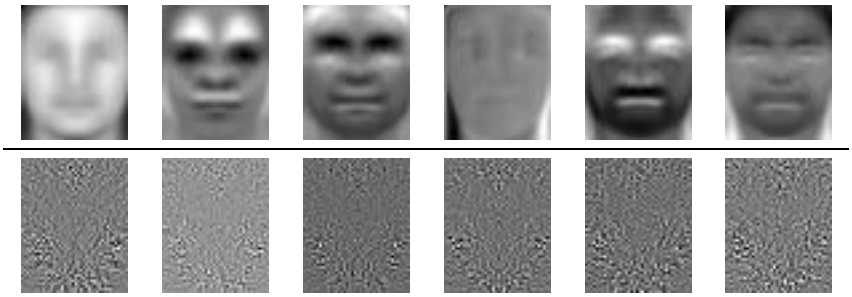


Fig. 4. Examples of mean face and eigenfaces

Finally, we can get the feature vectors $\boldsymbol{y}$ projecting the original image into the face space with the trained PCA matrix, as a common change of base:

$$\boldsymbol{y} = \phi^T \boldsymbol{x}$$

We can see that we are going from an N-dimensional image space to an L-dimensional face space, with $L < N$. For that reason we are dealing with a lossy transformation system. In our case with this effect we obtain the desired dimensionality reduction, but this technique can be used for other applications such as image compression.

In addition to this, we could see that images should be pre-processed before the projection is made. First, the images should be scaled according to the PCA projection matrix; that is, scale the images to the size of the training images used to create the matrix. Next we should normalize the images in mean, subtracting the mean image obtained from the training set, and normalize them to norm one.

4.1.2 Linear discriminant analysis (LDA)

LDA (linear discriminant analysis) [10] is also a linear projection like PCA but tries to maximize the distance among the different classes, not only focusing on the best representation of the samples. This algorithm is also known as Fisher linear discriminant (FLD) and in contrast to PCA, it uses the class specific information.

This method selects a projection matrix Ψ in such a way that the ratio between the between-class scatter matrix (S_B) and the within-class scatter matrix (S_W) is maximized. These matrices are defined as

$$S_B = \sum_{i=1}^{c} N_i (\overline{\boldsymbol{x}}_i - \overline{\boldsymbol{x}})(\overline{\boldsymbol{x}}_i - \overline{\boldsymbol{x}})^T$$

$$S_W = \sum_{i=1}^{c} \sum_{\boldsymbol{x} \in X_i} (\boldsymbol{x} - \overline{\boldsymbol{x}}_i)(\boldsymbol{x} - \overline{\boldsymbol{x}}_i)^T$$

where $\overline{\boldsymbol{x}}$ is the mean vector of all the sample set, $\overline{\boldsymbol{x}}_i$ is the mean vector of the set of samples X_i of the ith class, N_i is the number of samples of the ith class and c is the total number of classes.

So, if S_W is a non singular matrix, the optimal LDA projection [11] is the matrix with orthonormal columns which maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples and the determinant of the within-class scatter matrix of the projected samples; that is,

$$\Psi_{opt} = \arg\max_{\Psi} \frac{|\Psi^T S_B \Psi|}{|\Psi^T S_W \Psi|} = (\boldsymbol{\psi}_1 \boldsymbol{\psi}_2 \dots \boldsymbol{\psi}_m)$$

where $\boldsymbol{\psi}_i$ is the set of generalized eigenvectors of S_B and S_W corresponding to the higher eigenvalues λ_i,

$$S_B \boldsymbol{\psi}_i = \lambda_i S_W \boldsymbol{\psi}_i$$

Another way to maximize this relation [22] [23] is try to maximize the ratio $\frac{det\,|S_B|}{det\,|S_W|}$. The advantage of using this simplification is that if S_W is non-singular, the ratio is maximized when the column vectors of the projection matrix Ψ are the eigenvectors of $S_W^{-1} S_B$. With this criterion we can see that there are at most $c-1$ eigenvectors different from zero, so, the LDA space has a maximum dimension of $c-1$. Besides, at least $N+c$ samples (the dimension of the images plus the number of classes) are required to guarantee that S_W is not singular, which is quite hard to accomplish commonly. To solve this problem a variation of the technique proposed in [11] is commonly used, where an intermediate PCA projection is used to obtain a dimensionality reduction and then the LDA projection is made, so the feature vector will be obtained as

$$\boldsymbol{y} = \Psi_{LDA}^T \phi_{PCA}^T \boldsymbol{x}$$

So, the LDA projection matrix will be formed with the $c-1$ eigenvectors with the highest eigenvalue of $S_W^{-1} S_B$.

Therefore, the original N-dimensional image vector is projected into an intermediate L-dimensional face space, and finally is projected to a F-dimensional LDA space, being $N > L > F$.

4.1.3 Intrapersonal/extrapersonal Bayesian

The intrapersonal/extrapersonal Bayesian technique uses a different approach. This method is based on the use of a space generated by the subtraction of two images instead of using an image as a point in the face space. This subtraction is defined as the signed arithmetic difference between each pixel of the two images ($\Delta = I_1 - I_2$).

For that reason, every difference of faces only can be considered in one of two classes: the intrapersonal class (Ω_{I}), generated by pairs of images of the same person, or the extrapersonal (Ω_{E}), generated by the difference of pairs of images from different people.

In addition to this, we will suppose that these two classes follow a Gaussian distribution [12] [13], and to reduce the dimension of the image difference Δ a solution based on PCA is

employed where the image space is divided into two subspaces, the principal subspace F which contains the M principal components whereas the rest are included in an orthogonal subspace $\bar{F}$, which corresponds to the residual error called distance from feature space (DFFS). The components that are included in F are known as the distance in feature space (DIFS), a Mahalanobis distance for Gaussians. With these concepts in mind we can arrive to the following expression:

$$p(\Delta|\Omega) = \frac{e^{-\frac{1}{2}\sum_{i=1}^{M}\frac{y_i^2}{\lambda_i}}}{(2\pi)^{M/2}\prod_{i=1}^{M}\lambda_i^2}\frac{e^{-\frac{1}{2\rho}\varepsilon^2(\Delta)}}{(2\pi\rho)^{\frac{N-M}{2}}} = p_F(\Delta|\Omega)\hat{p}_{\bar{F}}(\Delta|\Omega)$$

where

- Δ is the difference image from which we want to estimate its Ω class conditional probability
- M is the dimension of the principal subspace F
- N is the dimension of the image espace
- y_i are the principal components (the coefficients from the projection into the principal subspace F)
- λ_i are the eigenvalues associated to the eigenvectors of F
- $\varepsilon^2(\Delta)$ is the residual error or DFFS
- $p_F(\Delta|\Omega)$ is the true density in F
- $\hat{p}_{\bar{F}}(\Delta|\Omega)$ is the estimated density in $\bar{F}$
- The optimal value for ρ is the mean of the eigenvalues from $\bar{F}$: $\rho = \frac{1}{N-M}\sum_{i=M+1}^{N}\lambda_i$

Once the class conditional probabilities are estimated, $p(\Delta|\Omega_{\mathrm{E}})$ and $p(\Delta|\Omega_{\mathrm{I}})$, we can use the Bayes theory to obtain the posterior probability to find to which of the two classes belongs the difference image. Then we can follow either a MAP or a ML strategy to classify the images. With the MAP rule we have to generate c (the number of classes) difference images $(\Delta_i, 1 \leq i \leq c)$ among the image to analyze and one of every class, and classify the image as belonging to the class i with higher posterior probability. On the other hand, with the ML rule only the intrapersonal class is considered and a similarity measure (S_{ML}) is commonly used for classification. This measure can be considered as an indirect estimation for the a priori probability $p(\Delta_i|\Omega_{\mathrm{I}})$ and can be simplified to

$$S_{ML}(\Delta) = \sum_{i=1}^{M}\frac{y_i^2}{\lambda_i} + \frac{\varepsilon^2(\Delta)}{\rho}$$

4.2 Classifying

Now the problem is how to classify the feature vectors in different classes to help us on the recognition of the faces. One of the first ideas about this problem is the one commented in [9]. This simple method consists in finding the face class k that minimizes the Euclidian distance between the vector $\boldsymbol{x}$ and the vector $\boldsymbol{x}_k$ which describes the kth class. The vectors $\boldsymbol{x}_k$ are calculated by averaging the feature vectors of the training faces of the same person. So, for each new face to be identified, the vector $\boldsymbol{x}$ is calculated and also the distances to

each known class. If the minimum distance is lower than a fixed threshold, we could classify the face to that class, and if it is greater, it may be classified as unknown and optionally used to begin a new face class. Finally, the eigenfaces are recalculated to add the new faces classified as known to the model. However, this approach, yet simple about the recognition part, do not seem the most appropriate in our approach to the problem, because we are interested in keeping a general face space and do not change the PCA projection matrix. With this we are looking for a generic and universal face space, where all the faces are represented in the same way and the discrimination among the different classes only depends on the characteristics of the models and the classifiers.

For this reason, we can consider the use of another type of classifiers, also taking into account that the density function of the groups of samples is unknown. Therefore, the use of nonparametric techniques to estimate the densities and to build the classifiers seemed the more appropriate ones. Among the different procedures we could use, we will focus on two of them: the k-NN and the Parzen classifiers.

One type of nonparametric methods tries to estimate the density functions $p(\boldsymbol{x}|\omega_j)$ from the sample patterns (like the Parzen classifier), whereas other type tries to estimate the posterior density functions $p(\omega_j|\boldsymbol{x})$ or directly the decision functions (as the k-NN classifier). This can also be seen as the use of a probabilistic or a geometric approach. However, we will see that these two techniques are more similar in their basis than it might look at first sight, as we could see in the next argument extracted from [4] and [26].

So most of the nonparametric techniques try to estimate an unknown probability density function $p(\boldsymbol{x}|\omega_j)$ based on the fact that the probability P that a vector $\boldsymbol{x}$ is in a region $R \subset \mathbb{R}^d$ is $P = \int_R p(\boldsymbol{u})d\boldsymbol{u}$. Therefore we can see that P is a smooth or averaged version of the density function $p(\boldsymbol{x})$, so we can estimate the density value from P. Assuming that we have N samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N$ independently and identically distributed (i.i.d.) from the considered distribution, the probability that k of the N samples fall in R is given by the binomial law,

$$P_k = \binom{N}{k} P^k (1-P)^{N-k}$$

and the value of P can be estimated as the proportion of samples that fall in R,

$$P \approx \frac{k}{N}$$

If $\boldsymbol{x}$ is a point in the region R and we consider that this region is so small that p is approximately constant in it, then P can be estimated as,

$$P \approx \int_R p(\boldsymbol{u})d\boldsymbol{u} \approx p(\boldsymbol{x}) \int_R d\boldsymbol{u} = p(\boldsymbol{x})V_R$$

where V_R is the volume enclosed by R in $\mathbb{R}^d$. If we compare these two last equations we obtain

$$\frac{k}{N} \approx p(\boldsymbol{x})V_R$$
$$p(\boldsymbol{x}) \approx \frac{k}{NV_R}$$

Theoretically if an unlimited number N of samples is available and the region R can be reduced to a point ($V_R \to 0$), we can obtain an accurate estimate of the probability density $p(\boldsymbol{x})$. However, in practice the number of samples is limited, and the volume V_R cannot be made arbitrarily small, so the estimation of the density will have a certain smoothing.

Then we are going to see the conditions needed to assure the convergence of the estimation in the theoretical case of having an infinite number of samples. First we form a sequence of regions $R_1, R_2, \ldots$, containing $\boldsymbol{x}$, the first with one sample, the second with two, etc. If V_N is the volume of R_N, k_N is the number of samples that fall in R_N, and $p_N(\boldsymbol{x})$ is the Nth estimate of $p(\boldsymbol{x})$, then

$$p_N(\boldsymbol{x}) = \frac{k_N/N}{V_N}$$

So, to make sure that $p_N(\boldsymbol{x})$ converges to $p(\boldsymbol{x})$, we should require that

$$\lim_{N\to\infty} V_N = 0$$
$$\lim_{N\to\infty} k_N = \infty$$
$$\lim_{N\to\infty} k_N/N = 0$$

There are two approaches to create sequences that satisfy these conditions. The first one is to specify an initial region V_N and shrink it as a function of N like $V_N = 1/\sqrt{N}$, which is basically the Parzen approach. On the other hand, we can specify k_N as a function of N like $k_N = \sqrt{N}$, and let the volume V_N grows until it encloses k_N neighbours of $\boldsymbol{x}$, which corresponds to a k_N-nearest neighbour approach.

4.2.1 k-NN classifier

The k-NN (k-nearest neighbour) is one of the simplest classification techniques, and it is based on the idea of finding the classes of the k-nearest neighbour vectors. Then the most represented class on those k neighbours is found and it is assigned to the test sample.

Anyway, we have seen that it can be interpreted from another point of view; that is, fixing k and N in $p(\boldsymbol{x}) \approx \frac{k}{NV_R}$, and letting V_R be variable. If we suppose that we place a volume V_R (if we use the Euclidean distance it will be a hypersphere) around $\boldsymbol{x}$ which captures k samples, and k_i of them are labelled as belonging to ω_i, then the obvious estimate for the class conditional probability $p(\boldsymbol{x}|\omega_i)$ is,

$$p(\boldsymbol{x}|\omega_i) \approx \frac{k_i}{N_i V_R}$$

where N_i is the number of elements from the total that belong to the class ω_i. Then, according to the Bayes theory, and considering that $p(\omega_i) \approx N_i/N$

$$p(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)p(\omega_i)}{p(\boldsymbol{x})} \approx \frac{\frac{k_i}{N_i V_R}\frac{N_i}{N}}{\frac{k}{NV_R}} = \frac{k_i}{k}$$

That is, the minimum error will be obtained by assigning $\boldsymbol{x}$ to the class with the highest posterior probability, the class most represented among the k nearest neighbours. The region R and its volume V_R are specific for each $\boldsymbol{x}$ and are dependent on the metric, but the classification only depends on the number of samples k_i and not on V_R.

In Fig. 5 we can see an example of the partition of the space into cells obtained using the minimum error Bayes decision rule in a 1-NN classifier, where the points closer to a training vector are labelled by the category of that point. These are known as Voronoi tessellations.

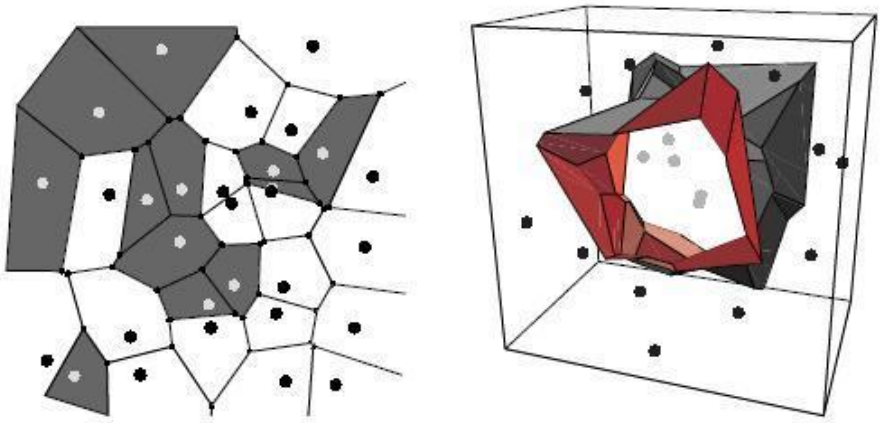


Fig. 5. Examples of a Voronoi tessellation in a 2D and in a 3D space, obtained from [4]

However, we will use a simple implementation. We will calculate the distance of the sample to analyze to the model samples, and then we will look to the most repeated category of the k nearest neighbours. Nevertheless, we are interested in obtaining an estimation of the posterior probability, because with this information we can apply the same criteria that we use with Parzen about the updates, and it also will be useful when we combine the results for groups of images, as we will see later in sections 5.4 and 5.7.1. For that reason, a simple way to obtain the probability is given in [5], which a priori should be more accurate than the previous $p(\omega_j|\boldsymbol{x}) = k_j/k$, where if k is the number of neighbours, $\boldsymbol{x}^{(i)}$ is the ith nearest neighbour of $\boldsymbol{x}$, and $d(\boldsymbol{x}, \boldsymbol{x}^{(i)})$ is the distance between $\boldsymbol{x}$ and $\boldsymbol{x}^{(i)}$, then

$$P(\omega_j|\boldsymbol{x}) = \frac{\sum_{\boldsymbol{x}^{(j)} \in \omega_j} \frac{1}{d(\boldsymbol{x}, \boldsymbol{x}^{(j)})}}{\sum_{i=1}^{k} \frac{1}{d(\boldsymbol{x}, \boldsymbol{x}^{(i)})}}$$

As we could suppose this classifier has different drawbacks. For example, the usual high dimensionality of the data leads to a costly implementation. To solve this problem, some techniques can be used [4], like the use of partial distances (calculating distances in r of the d dimensions, to make it more efficient), the use of search trees (preestructuring the data can make the search of the closest samples faster) or the use of editing or pruning (eliminating "useless" prototypes during training, for example the prototypes that are surrounded by samples of the same category). For these reasons we can find many different variations of the nearest neighbour classifier or the k-NN version. In any case, we will first use the standard solution and only if the resources needed are too high we will evaluate the suitability of using these tools.

4.2.2 Parzen classifier

The Parzen classifier, which is also referred as Parzen window or kernel density estimation method, is designed to estimate the class conditional probabilities and it can be modelled again basing us on the expression $p(\boldsymbol{x}) \approx \frac{k}{NV_R}$, and now fixing N and V_R, and being k found from the data.

The Parzen approach can be introduced assuming that R_N is a d-dimensional hypercube, with side h_N, so the volume of the region is $V_N = h_N^d$ (the subscript N refers to the total number of samples used). We can approximate the number of samples that fall in the hypercube using the window function (also called kernel function or Parzen window) $K(\boldsymbol{u})$, $\boldsymbol{u} = [u_1, \dots, u_d]^T \in \mathbb{R}^d$, such as

$$K(\boldsymbol{u}) = \begin{cases} 0 & if\ |u_j| \leq 1/2 \quad j = 1, \dots, d \\ 1 & otherwise \end{cases}$$

This function defines a unit hypercube centered at the origin. So, $K((\boldsymbol{x} - \boldsymbol{x}_i)/h_N)$ is equal to one if $\boldsymbol{x}_i$ falls within the hypercube of volume V_N centered at $\boldsymbol{x}$, and is zero for points outside the hypercube. So, the number of samples in the hypercube is determined as

$$k_N = \sum_{i=1}^{N} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h_N}\right)$$

and the probability density function (pdf) can be estimated as

$$p_N(\boldsymbol{x}) \approx \frac{k_N}{NV_N} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{V_N} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h_N}\right)$$

We can see that we could generalize this result by using different window functions as, in short, we are estimating the probability $p(\boldsymbol{x})$ simply as an average of functions that depends on the distance of $\boldsymbol{x}$ to the samples $\boldsymbol{x}_i$. To be sure that our estimation is a valid density function we can force the window function to be a density function; that is, be nonnegative and integrate to one. For this reason, it is common to use a Gaussian function, but other types of functions can be used, as uniform or triangular ones.

The effect of the window width h_N on the estimation of the probability $p_N(x)$ is quite significant, so it requires further attention. If we define

$$\delta_N(\boldsymbol{x}) = \frac{1}{V_N} K\left(\frac{\boldsymbol{x}}{h_N}\right)$$

then

$$p_N(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \delta_N(\boldsymbol{x} - \boldsymbol{x}_i)$$

It is clear that h_N affects on the width of $\delta_N(\boldsymbol{x})$ and as $V_N = h_N^d$, it affects on the amplitude too. If h_N is very large, the amplitude of δ_N will be small and the value of $\delta_N(\boldsymbol{x} - \boldsymbol{x}_i)$ will be closer to $\delta_N(0)$ although the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_i$ is relatively high. For that reason, the estimation of $p_N(\boldsymbol{x})$ will be made with the superposition of N broad, slowly changing functions and as a result of this we will have a smooth and biased estimation of $p(\boldsymbol{x})$. On the other hand, if h_N is very small, the amplitude of δ_N will be higher, especially when $\boldsymbol{x}$ and $\boldsymbol{x}_i$ are close. For that reason, now the estimation of $p_N(\boldsymbol{x})$ will be made with the superposition of N narrow functions and as a result of this we will have a spiky and "unbiased" estimation of $p(\boldsymbol{x})$.

That is, when h_N is too large, the estimation of the pdf will have a lower resolution, but if it is too small, we will have a large statistical variability. So with a limited number of samples we should look for an optimal value according to this trade-off. However, in the theoretical case of having an unlimited number of samples, the value of h_N should decrease to zero as N increases, so the estimation $p_N(\boldsymbol{x})$ would be closer to $p(\boldsymbol{x})$. In Fig. 6 we can see an example of the Parzen estimation of a univariate normal density using different widths and numbers of samples, extracted from [4]. Here we can see the effect of the width h and the number of samples (the vertical axes had been scaled), and it is interesting to notice how the estimation will be the same that the true generation pdf, regardless of the value of h, when the number of samples is infinite.

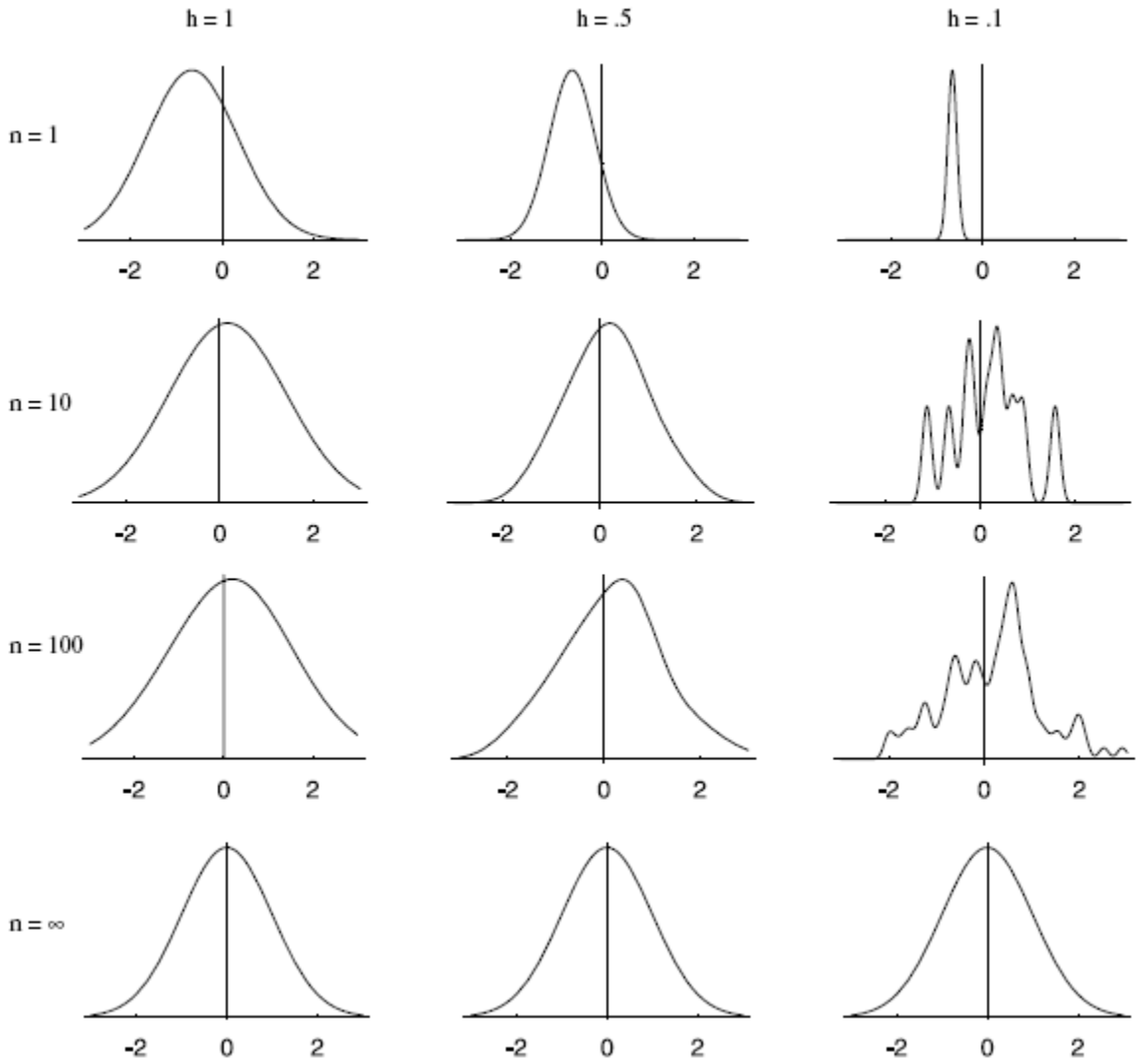


Fig. 6. Example of Parzen-window density estimations for different window widths and number of samples.

In any case, the choice of the optimal bandwidth h is crucial to obtain a good estimation, so different techniques can be found to select it [27] [28]. However the optimal value of h depends both on the design sample size and on the distribution of the pattern vectors. For example, if we have two Gaussian distributions with equal covariance matrices, the optimal decision boundary is a hyperplane and h should be very large. In contrast, if we have complex multimodal distributions then the decision boundary will be nonlinear, so we should use a low value for h even for a reduced number of samples.

For that reason it is common to find an optimal value for h evaluating the performance for different values of h and choosing that value which provides the best results. So our first idea was to fix the same value for all the classes, and vary it until we find an optimal value. It is common [29] to initially use a set of ten values, $0.001, 0.01, 0.03, 0.1, 0.3, 1, 3, 10, 100, 1000$, and then refine the search. The second idea was to use the simplest method to select the optimal kernel bandwidth proposed in [27], using a different value for every class as $h_k = 1.06\sigma_k N_k^{-1/5}$, where σ_k is the standard deviation of the class k, and N_k is the number of samples of the class. Finally, and we will see that seems to provide the best results when we test the Parzen and the modified Parzen classifiers in sections 6.1.7 and 6.1.8, we will use a method which defines the width as a

multiple of the standard deviation of the class; that is, $h_k = c\,\sigma_k$, where we vary the value of c.

Then, once we have the estimation of all the class conditional probability density functions, the Parzen classifier assigns the class which gives the highest probability for that test pattern. In our case, we will use a Gaussian kernel function. So, if we suppose that $\boldsymbol{x}_1^k$, $\boldsymbol{x}_2^k$, … , $\boldsymbol{x}_{N_k}^k$ are the N_k training samples of class ω_k, the class-conditional probability density function (pdf) will be estimated as

$$p(\boldsymbol{x}|\omega_k) = \frac{1}{N_k}\sum_{j=1}^{N_k}\left[\frac{1}{(2\pi)^{n/2}\,h_k^n\,|\hat{\Sigma}_k|^{1/2}}\,exp\left\{-\frac{1}{2h_k^2}(\boldsymbol{x}-\boldsymbol{x}_j^k)^T\,\hat{\Sigma}_k{}^{-1}(\boldsymbol{x}-\boldsymbol{x}_j^k)\right\}\right]$$

where $\hat{\Sigma}_k$ is the sample covariance matrix of class ω_k. Then, using the Bayes formula, we will estimate the posterior probability as

$$p(\omega_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_k)\,p(\omega_k)}{p(\boldsymbol{x})} =$$

$$= \frac{\frac{1}{N_k}\sum_{j=1}^{N_k}\left[\frac{1}{(2\pi)^{n/2}\,h_k^n\,|\hat{\Sigma}_k|^{1/2}}\,exp\left\{-\frac{1}{2h_k^2}(\boldsymbol{x}-\boldsymbol{x}_j^k)^T\,\hat{\Sigma}_k{}^{-1}(\boldsymbol{x}-\boldsymbol{x}_j^k)\right\}\right]\frac{N_k}{N}}{\frac{1}{N}\sum_{k=1}^{L}\left(\sum_{j=1}^{N_k}\left[\frac{1}{(2\pi)^{n/2}\,h_k^n\,|\hat{\Sigma}_k|^{1/2}}\,exp\left\{-\frac{1}{2h_k^2}(\boldsymbol{x}-\boldsymbol{x}_j^k)^T\,\hat{\Sigma}_k{}^{-1}(\boldsymbol{x}-\boldsymbol{x}_j^k)\right\}\right]\right)}$$

$$= \frac{\sum_{j=1}^{N_k}\left[\frac{1}{(2\pi)^{n/2}\,h_k^n\,|\hat{\Sigma}_k|^{1/2}}\,exp\left\{-\frac{1}{2h_k^2}(\boldsymbol{x}-\boldsymbol{x}_j^k)^T\,\hat{\Sigma}_k{}^{-1}(\boldsymbol{x}-\boldsymbol{x}_j^k)\right\}\right]}{\sum_{k=1}^{L}\left(\sum_{j=1}^{N_k}\left[\frac{1}{(2\pi)^{n/2}\,h_k^n\,|\hat{\Sigma}_k|^{1/2}}\,exp\left\{-\frac{1}{2h_k^2}(\boldsymbol{x}-\boldsymbol{x}_j^k)^T\,\hat{\Sigma}_k{}^{-1}(\boldsymbol{x}-\boldsymbol{x}_j^k)\right\}\right]\right)}$$

where N is the total number of training samples, L is the total number of classes, and we suppose that $p(\omega_k) = \frac{N_k}{N}$.

One of the problems about the Parzen classifier is that it is time consuming and requires more storage capacity than other classifiers because it has to use all the training samples to compute the density estimate for a given class and for a given test pattern. This problem can be tackled using the reduced Parzen classifier [30], since using this technique, we can select a given number of representative samples whose Parzen density estimation closely matches that of the entire sample set. However, this will not be a difficulty in our application as we commonly have the small sample size problem; that is, the number of training samples is reduced according to data dimensionality. Thus, to solve this problem we can modify the classifier using the bootstrap and Toeplitz techniques to improve the results as introduced in the modified Parzen classifier [31].

4.2.3 The modified Parzen classifier

The modified Parzen classifier is designed to reduce the effect of the high dimensional space and the small sample size problem when we are estimating the sample covariance matrices,

needed when we use Gaussian kernel functions. The use of Gaussian kernel functions requires the estimation of the covariance matrix, but in practice it is estimated from a limited number of training samples, so if the ratio of the training sample size to the dimensionality is significantly small (we would need a minimum number of samples equal to the dimension of the space), the matrix becomes singular and it is impossible to invert it, so we will not be able to use the classifier. For that reason, the Toeplitz technique is used to reduce the estimation error of the covariance matrix, and the bootstrap technique is used to minimize the influence of the samples which distort the distribution, by resampling the data set and as a result of this, smoothing the density function.

Now we are going to present the definition of the technique as proposed in [31]. If we define the sample covariance matrix for a class (the subscript k of the class is omitted for simplicity) as,

$$\hat{\Sigma} = \begin{bmatrix} \hat{\sigma}_1^2 & \hat{c}_{12} & \cdots & \hat{c}_{1n} \\ \hat{c}_{12} & \hat{\sigma}_2^2 & & \\ \vdots & & \ddots & \\ \hat{c}_{1n} & & & \hat{\sigma}_n^2 \end{bmatrix}$$

where $\hat{\sigma}_i^2$ is the sample variance of $\boldsymbol{x}_i$ and $\hat{c}_{ij}$ is the sample covariance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which can be expressed as $\hat{c}_{ij} = \hat{\rho}_{ij}\hat{\sigma}_i\hat{\sigma}_j$, and $\hat{\rho}_{ij}$ is the sample correlation coefficient between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Then, $\hat{\Sigma} = \hat{\Gamma}\hat{\mathrm{R}}\hat{\Gamma}$, where

$$\hat{\mathrm{R}} = \begin{bmatrix} 1 & \hat{\rho}_{12} & \cdots & \hat{\rho}_{1n} \\ \hat{\rho}_{21} & 1 & & \\ \vdots & & \ddots & \\ \hat{\rho}_{n1} & & & 1 \end{bmatrix} \text{ and } \hat{\Gamma} = \begin{bmatrix} \hat{\sigma}_1 & & & 0 \\ & \hat{\sigma}_2 & & \\ & & \ddots & \\ 0 & & & \hat{\sigma}_n \end{bmatrix}$$

Assuming the Toeplitz form for the correlation matrix, the correlation coefficient between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ depends only on $|i-j|$, so

$$\hat{\mathrm{R}}_{\mathrm{T}} = \begin{bmatrix} 1 & \hat{\rho} & \cdots & \hat{\rho}^{n-1} \\ \hat{\rho} & 1 & & \vdots \\ \vdots & & \ddots & \hat{\rho} \\ \hat{\rho}^{n-1} & \cdots & \hat{\rho} & 1 \end{bmatrix} \text{ and } \hat{\mathrm{R}}_{\mathrm{T}}^{-1} = \frac{1}{1-\hat{\rho}^2}\begin{bmatrix} 1 & -\hat{\rho} & 0 & \cdots & 0 \\ -\hat{\rho} & 1+\hat{\rho}^2 & & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & & 1+\hat{\rho}^2 & -\hat{\rho} \\ 0 & \cdots & 0 & -\hat{\rho} & 1 \end{bmatrix}$$

and

$$\left|\hat{\mathrm{R}}_{\mathrm{T}}\right| = (1-\hat{\rho}^2)^{n-1} \text{ where } \hat{\rho} = \frac{1}{n-1}\sum_{i=1}^{n-1}\frac{\hat{c}_{i,i+1}}{\hat{\sigma}_i\hat{\sigma}_{i+1}}$$

Then, as we are interested in the inverse of the covariance matrix and the determinant, we can estimate them as

$$\hat{\Sigma}_{\mathrm{T}}^{-1} = \hat{\Gamma}^{-1}\hat{\mathrm{R}}_{\mathrm{T}}^{-1}\hat{\Gamma}^{-1}$$

$$= \frac{1}{1-\hat{\rho}^2}\begin{bmatrix} \frac{1}{\hat{\sigma}_1} & & & 0 \\ & \frac{1}{\hat{\sigma}_2} & & \\ & & \ddots & \\ 0 & & & \frac{1}{\hat{\sigma}_n} \end{bmatrix}\begin{bmatrix} 1 & \hat{\rho} & 0 & \cdots & 0 \\ -\hat{\rho} & 1+\hat{\rho}^2 & & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & & 1+\hat{\rho}^2 & -\hat{\rho} \\ 0 & \cdots & 0 & -\hat{\rho} & 1 \end{bmatrix}\begin{bmatrix} \frac{1}{\hat{\sigma}_1} & & & 0 \\ & \frac{1}{\hat{\sigma}_2} & & \\ & & \ddots & \\ 0 & & & \frac{1}{\hat{\sigma}_n} \end{bmatrix}$$

$$\left|\hat{\Sigma}_{\mathrm{T}}\right| = \left|\hat{\Gamma}\right|\left|\hat{\mathrm{R}}_{\mathrm{T}}\right|\left|\hat{\Gamma}\right| = \left(\prod_{i=1}^{n}\hat{\sigma}_i\right)^2 (1-\hat{\rho}^2)^{n-1}$$

We can see that with the Toeplitz approach we only need $n+1$ parameters: $\hat{\rho}$ and $\hat{\sigma}_i, i = 1, \dots, n$. Therefore, the computational cost is reduced in contrast to the classical estimation of the covariance matrix, and in addition to this we obtain directly the inverted matrix.

About the bootstrap technique, its objective is generate a sample set $Y_{N_k} = \{\boldsymbol{y}_1^k, \boldsymbol{y}_2^k, \dots, \boldsymbol{y}_{N_k}^k\}$; that is, sampling with replacement from the original data set X_{N_k}, to reduce the effect of the outlier samples. Firstly, we will randomly select a training sample $\boldsymbol{x}_{s_0}^k$ from X_{N_k}. Then the r nearest neighbour samples $\boldsymbol{x}_{s_1}^k, \boldsymbol{x}_{s_2}^k, \dots, \boldsymbol{x}_{s_r}^k$ of $\boldsymbol{x}_{s_0}^k$ are selected using the Euclidean distance. Next the bootstrap sample is computed $\boldsymbol{y}_1^k = \sum_{i=0}^{r}\omega_i \boldsymbol{x}_{s_i}^k$, where ω_i is a weight such as $\omega_i = \frac{\Delta_i}{\sum_{c=0}^{r}\Delta_c}, 0 \le i \le r$. Δ_i is chosen from a uniform distribution on [0,1]. Finally we repeat this algorithm N_k times.

Finally if we use these techniques, the class-conditional density estimation will be

$$p(\boldsymbol{x}|\omega_k) = \frac{1}{N_k}\sum_{j=1}^{N_k}\left[\frac{1}{(2\pi)^{n/2}\, h_k^n\, |\hat{\Sigma}_{k_T}|^{1/2}} exp\left\{-\frac{1}{2h_k^2}(\boldsymbol{x}-\boldsymbol{y}_j^k)^T\, \hat{\Sigma}_{k_T}{}^{-1}(\boldsymbol{x}-\boldsymbol{y}_j^k)\right\}\right]$$

4.2.4 The reduced Parzen classifier

The reduced Parzen classifier [30] has the objective of obtaining a reduced sample set representative of a larger sample set; that is, the selection of a given number of samples whose Parzen density estimation closely matches that of the entire sample set. Theoretically the Parzen classifier needs all the sample set, in our case all the vectors in the initial models, as the prototype set, and this can be too time consuming for large number of samples. Nevertheless, this will not be a problem in our case as we will commonly have the small sample size problem. Anyway we can take advantage of this idea as a possible method to discard those vectors that are redundant for the estimation of the densities that, as we will see next in sections 5.4 and 5.7.1, could be useful for the model updates.

First we can consider that we have N samples from a density function $p(\boldsymbol{x})$, and we want to select r samples $(r < N)$, such that the Parzen density estimation for the N sample set and the r sample set is as close as possible.

If we assume a Gaussian kernel, the Parzen density estimation at $\boldsymbol{x}$ for N samples is (again, we skip the subscript k of the class for simplicity, but now we note with N or r according to the number of samples)

$$\hat{p}_N(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} k\,(\boldsymbol{x} - \boldsymbol{x}_i)$$

where

$$k(\boldsymbol{x} - \boldsymbol{x}_i) = \frac{1}{(2\pi)^{n/2}\, h^n |\Sigma|^{1/2}} exp\left\{-\frac{1}{2h^2}(\boldsymbol{x} - \boldsymbol{x}_i)^T \Sigma^{-1}(\boldsymbol{x} - \boldsymbol{x}_i)\right\}$$

and n is the dimensionality, Σ is the class covariance matrix and h is the kernel width. In an analogue way, if we have r representatives $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r$ the density estimation will be

$$\hat{p}_r(\boldsymbol{x}) = \frac{1}{r}\sum_{i=1}^{r} k\,(\boldsymbol{x} - \boldsymbol{y}_i)$$

The similarity between $\hat{p}_r(\boldsymbol{x})$ and $\hat{p}_N(\boldsymbol{x})$ can be estimated using the entropy; that is,

$$\int \ln\left[\frac{\hat{p}_r(\boldsymbol{x})}{\hat{p}_N(\boldsymbol{x})}\right]\hat{p}_N(\boldsymbol{x})\, d\boldsymbol{x}$$

The entropy expression can be rewritten as $E\{\ln[\hat{p}_r(\boldsymbol{x})/\hat{p}_N(\boldsymbol{x})]\}$, where the expectation is taken with respect to $\hat{p}_N(\boldsymbol{x})$. Then, if we replace the expectation by the sample mean of the samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, we can approximate the last equation by

$$J = \frac{1}{N}\sum_{i=1}^{N}\left[\ln\left(\hat{p}_r(\boldsymbol{x}_i)\right) - \ln\left(\hat{p}_N(\boldsymbol{x}_i)\right)\right]$$

or

$$J = \frac{1}{N}\sum_{i=1}^{N}\left[\ln\left(\frac{1}{r}\sum_{j=1}^{r} k\,(\boldsymbol{x}_i - \boldsymbol{y}_j)\right) - \ln\left(\frac{1}{N}\sum_{j=1}^{N} k\,(\boldsymbol{x}_i - \boldsymbol{x}_j)\right)\right]$$

To find the best r representatives from the existing samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, we would like to maximize J over all possible r elements subsets of the original N element set. To avoid the search over the possible $\binom{N}{r}$ subsets, the next procedure is followed, searching the maximum J by replacing one element of the representative set by the best candidate not yet selected.

1. Select randomly an initial subset of r samples from the N data set. The r sample set is called STORE and the remaining $N - r$ samples TEST.
2. For each element $\boldsymbol{x}_t$ in TEST, compute the change in J that results if the sample is transferred to STORE, that is

$$\Delta J_1(\boldsymbol{x}_t) = J_{r+1}(\boldsymbol{x}_t) - J_r$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left[\ln\left(\frac{1}{r+1}\sum_{j=1}^{r} k\left(\boldsymbol{x}_i - \boldsymbol{y}_j\right) + k(\boldsymbol{x}_i - \boldsymbol{x}_t)\right)\right.$$
$$\left. - \ln\left(\frac{1}{r}\sum_{j=1}^{r} k\left(\boldsymbol{x}_i - \boldsymbol{y}_j\right)\right)\right]$$

3. Pick the element $\boldsymbol{x}_t$ corresponding to the largest ΔJ_1 and call it $\boldsymbol{x}_t^*$.
4. For each element $\boldsymbol{x}_s$ in STORE, compute the change in J that results if the sample is transferred to TEST; that is,

$$\Delta J_2(\boldsymbol{x}_s) = J_r(\boldsymbol{x}_s) - J_{r+1}$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left[\ln\left(\frac{1}{r}\sum_{j=1}^{r} k\left(\boldsymbol{x}_i - \boldsymbol{y}_j\right) + k(\boldsymbol{x}_i - \boldsymbol{x}_t^*) - k(\boldsymbol{x}_i - \boldsymbol{x}_s)\right)\right.$$
$$\left. - \ln\left(\frac{1}{r+1}\sum_{j=1}^{r} k\left(\boldsymbol{x}_i - \boldsymbol{y}_j\right) + k(\boldsymbol{x}_i - \boldsymbol{x}_t^*)\right)\right]$$

5. Pick the element $\boldsymbol{x}_s$ corresponding to the largest ΔJ_2 and call it $\boldsymbol{x}_s^*$.
6. The change of J due to these two operations is $\Delta J = \Delta J_1(\boldsymbol{x}_t^*) + \Delta J_2(\boldsymbol{x}_s^*)$. To maximize J we would like to have $\Delta J > 0$. If $\boldsymbol{x}_s^*$ satisfies $\Delta J > 0$, then transfer $\boldsymbol{x}_s^*$ to TEST, transfer $\boldsymbol{x}_t^*$ to STORE and go to step 2.
7. Otherwise, find the element $\boldsymbol{x}_t$ corresponding to the next largest ΔJ_1 and call it $\boldsymbol{x}_t^*$.
8. If $\boldsymbol{x}_t^*$ exists, go to step 4.
9. Otherwise, stop.

5 Architecture of the system

At this point we have seen the common stages of a pattern recognition problem and the particular case of face recognition systems. Besides we have seen the basic techniques used in face recognition and others that seem suitable for our system. Now we are going to analyze the characteristics of our application and consequently choose the most appropriate tools for it. This analysis will be performed according to the design stages of a pattern recognition system (as we saw in Fig. 1): data collection, feature choice, model selection, classification, training and testing.

Briefly, we can summarize the solutions used in our system in the use of holistic techniques for feature extraction, a dynamic model for the representation of the classes, non-parametric techniques for classification and sets of images to perform better-founded decisions.

5.1 Previous considerations

Our objective was the design of a face recognition system applied to controlled environments. As we have commented in the introduction, the CHIL project was one example of this type of environments, so we will take into account the main aspects of the general architecture of this project and specifically from the video subsystem as the starting point to determine the characteristics of the input data. However, the general considerations about these data can be extrapolated to other controlled environments. For that reason, if we focus on the video subsystem we can see that we have different video sequences obtained from a set of cameras. In the case of the UPC "smart room", we are interested in the six fixed cameras (four on the corners and two on the middle of the room) and the pan-tilt-zoom (PTZ) camera (a moving camera with zoom capability which is commonly pointing to the entrance of the room). With these cameras we can obtain video sequences of the people in the room. As we have pointed previously, there will be a face detection system previous to the face recognition stage, so we will deal with sets of faces from different people, possibly with pose, light and size variation, and previously labelled from a tracking system.

The main characteristics of the obtained faces will be:

- Reduced size, because we are extracting faces from low or medium-resolution video sequences. In order to have an idea about the order of magnitude of the sizes, we can obtain faces of a minimum size about 10x15 pixels for the fixed cameras to a maximum about 40x60 pixels for the PTZ (see Fig. 7).
- Availability of sets of faces, as we can obtain different samples from the video.
- Pose variation, because we obtain faces from moving people in the room. However, and due to the location of the cameras it is usual to obtain larger and nearly frontal images with the PTZ camera, whereas with the lateral cameras we get smaller and nearly frontal or profile faces. Nevertheless, we have faces in other poses, but we will focus on these two cases.
- Low illumination variation, as we are in a controlled environment which is uniformly illuminated with artificial light.

Fig. 7. Examples of faces obtained from the CHIL room (from the PTZ above and from the lateral cameras below)

With these premises in mind, now we can analyze the different design stages of our system.

5.2 Data collection / Pre-processing

This stage requires a great amount of time and resources because the availability of a big and representative data set will help in the training and testing parts to obtain a robust system.

For that reason we will use different known databases to perform the main simulations of the system, while it will be tested later with "real" data. Although in most of these databases there are audio captures and video sequences we will use only still images extracted from those videos. First we will use the XM2VTS database [20], which is formed by images of 295 different people, with both frontal and profile views, of 720x576 (width x height) pixels. These images were recorded in four sessions uniformly distributed in a period of five months to capture the variability of the people in a relatively large time lapse, and with a uniform illumination and a blue background to easily segment the head. In any case, the faces have been cut manually to get an accurate version, obtaining 1180 frontal faces, with an average size of 205x303 pixels, and 1161 left profile faces and 795 right profile faces with an average size of 146x299 pixels. We can see that these faces are significantly larger than the ones available in a real application, so we should reduce them. In Fig. 8 we can see examples from this database.

Fig. 8. Sample images (frontal and profile) from the XM2VTS database

The frontal faces will be delimited by the forehead and the chin in vertical, and the two ears,

but excluding them (or the hair if they are covered), in horizontal. For the profiles, the width is determined by the nose and the ear or the hair. In Fig. 9 there are examples of the two cases.

Fig. 9. Examples of cut out faces (frontal and profile)

The other database that we will use is the BANCA database [21]. It contains several frontal images of people taken in three different scenarios depending on the image quality: controlled, degraded and adverse. Four sessions per scenario were recorded, spanning three months, each of them containing 10 images of 52 different subjects. This structure is repeated for four different countries: England, France, Italy and Spain, and we will use the Spanish part. As we have done with the XM2VTS database, the faces were manually cut; obtaining an average size of 94x144 pixels (with the Spanish part). In Fig. 10 we can see examples from this database.

Fig. 10. Sample images from the BANCA database. From left to right: controlled, degraded and adverse scenario

In addition to this, we will finally test our system with “real” data, as images obtained from the CHIL system, like the ones shown in Fig. 7 and from other similar sequences.

Furthermore, as we can identify this stage with the pre-processing step in a pattern classification system, we have to perform other tasks as the normalization of the images. For example, the faces have to be converted to grey-scale, as the colour information is redundant for the next stages, especially for the holistic methods. For that reason we should perform an RGB-YUV conversion to keep the luminance of the faces, and save the images in an appropriate format (SUN raster .ras with 8 bits per pixel in our case).

Besides, faces must be scaled to a determined size and normalized in mean and norm. This is done for the characteristics of the feature extractor as we will see later. In any case, it is clear that when the faces are big we have more available information than if the images are

smaller, but their processing will be harder as we will have to deal with a higher dimensional space. So we should consider this trade-off when determining the size of the faces but we are limited by the original sizes. For that reason we will use three different typical sizes, 10x15, 20x30 and 40x60 for the frontal faces, and 10x20, 20x40 and 30x60 for the profile views (note that we keep an aspect ratio equal to 1.5 for the frontals and 2 for the profiles, as the aspect ratios obtained from the average size of the faces manually cut of the XM2VTS database are 1.48 for the frontals and 2.05 for the profiles).

Another characteristic to consider is that as the left and right profile faces are commonly symmetric, we will not take into account the two profiles independently and we will mirror the right profiles to consider them as left profiles (we have chosen the left profiles because there are more available in the XM2VTS database, but in theory it is irrelevant to choose one side or the other).

Finally we will use masks to reduce as much part of the background as possible. For the frontal faces we will use oval masks (see Fig. 11), as we keep most of the discriminant part of the faces (a generic face has a shape similar to an oval), and the effect of the background and the hair is reduced. On the other hand, there is no easy definition for a generic mask for the profile images, so those images will not be masked. This masking could be done detecting some fiducial points as the eyes and the nose, and scaling the face maintaining their relative positions, so a mask could be implemented, but as we will see next these techniques would exceed the complexity of the feature extractor.

Fig. 11. Example of a cut out face and its masked version

5.3 Feature selection

We have seen that feature selection is a very decisive part in a pattern recognition system. Our goal is to find the set of numerical features that best represent our faces. As we have commented previously, there are two main approaches to obtain these features: (i) the holistic approach (or appearance-based) where all the information of the object, in our case the face, is used as a whole; and (ii) the feature-based approach, where some local features are used, for example the size of the nose, the distance between the eyes, etc. According to our application, the holistic approach has been chosen. This is done because the feature-based methods need high resolution images to detect the fiducial points of the face with a high precision, which does not seem appropriate for our faces extracted from the video sequences. Besides, this increases the complexity and the computational cost, but on the other hand these techniques are more invariant to illumination and pose changes. For these reasons, we have preferred the appearance-based methods although their performance is not as good a priori, but its complexity is clearly lower and seem more adequate according to the characteristics of our application.

Among the different holistic techniques we have seen the three main algorithms: PCA, LDA and the Bayesian intrapersonal/extrapersonal. As we could see, the Bayesian approach is a bit different from PCA and LDA, as it does not use the face space as the others do because it uses the difference of the images. Besides it has an "associated" classifier, according to the common assumption of Gaussian distributions and the use of the ML rule. Moreover, its performance is reduced if the faces are not well aligned and most of the background is removed [24], and the alignment needed for the coherent subtraction of the faces is a parameter that we cannot control in our system. For all these reasons, this technique is discarded.

We have seen that PCA is the simplest of these techniques, but it does not mean that its performance is worse than the more complex solutions, as the results depend on the characteristics of the application. One of the reasons to prefer PCA over LDA is given in [23], and is that when the training data set is small, PCA can outperform LDA. This will be a common situation in our case, where the number of vectors to create the initial models is reduced compared to the dimensionality of the data (see curse of dimensionality in section 2.4). Furthermore, with the election of the model that we will see next, which is updated periodically, LDA is a too complex solution because it should be trained continuously, changing the LDA projection matrices (whereas in PCA we keep the corresponding projection matrix). For these reasons PCA has been chosen as the technique used for feature extraction, although in any case the basic LDA algorithm has been implemented to compare it with PCA.

5.4 Model definition

Nevertheless, we still have to deal with the fact that the processes that we are representing are non stationary. PCA, that as we have seen is also the base for LDA and the Bayesian classifier, was originally created to model multidimensional random variables, and as we are extending it to model non stationary random processes we have to include in some way the temporal variation. There are different ways to tackle this problem, for example, computing the eigenvectors iteratively when the new samples are available one by one, or using an individual PCA approach where an eigenspace is created for every individual which is updated with the new available samples using parameters to give more importance to recent samples than to older ones [25].

On the other hand, our approach tackles this issue from a different point of view. First we are interested in a universal PCA approach; that is, a generic face space for all the individuals. Then a model for every individual is defined as a set of feature vectors of the same person in the universal face space. Finally these models are updated; that is, the existing vectors generated on the initial training stage and the testing samples are susceptible of being added or deleted from the models following some criteria, as if we were doing a continuous training.

For that reason, firstly we have to consider which the most appropriate vectors to add to the models are, and we can think about the images which are not represented by any class (that is, those images "far" from the existing classes, but the tracking assures us that it should belong to a determined class) and those which are only represented by one class (i.e. the

images that are clearly defined by one model, as if they were represented by two or more classes the model could grow too much and overrun other classes). Then, we have to think about methods to decide if the sample has to be added or not, and we could discuss about different approaches like a geometric or a probabilistic solution. Besides we can think about similar procedures for vector deletion. All these general ideas are part of the concept of model update, but they have to be specified according to the definition of our algorithm, that we will see next in section 5.7. For that reason, the concrete definition of the techniques used in model update will be explained in section 5.7.1.

In addition to this, we also have to notice two important aspects related to the models. The first is the possible use of a dynamic number of classes. This means that if a new unknown individual is presented to the system and the system verifies that its images do not belong to any existing class, it generates a new model for that person.

The second aspect to note is the use of a view-based approach [15] to handle the problem of having faces with different pose. One way to try to introduce the pose variation in our system is the use of a view-universal eigenspace which encodes the identities and views information, using images in different poses to generate the space. On the other hand, with the view-based formulation we have an extension of the known technique to a set of eigenspaces, one for each view. This can be seen like a set of parallel observers trying to see which of them provides a better representation for the face. The last solution seems more appropriate to our problem, because we have a trade-off between the complexity of the space and the number of spaces, and the results show that it is better to have a set of eigenspaces that represents accurately the faces than an eigenspace that represents all the views but will provide very complex models. Besides, this technique is also appropriate as the selection of the view can be done prior to the recognition stage. For instance, it is common for face detectors to find features as the eyes or the nose, so they can provide an estimation of the pose between frontal or profile, for example, according to the presence of the two eyes or only one. Anyway, it is possible to estimate the view using the residual description error (or distance from face space (DFFS) [9]); that is, the squared distance between the mean-adjusted and scaled to PCA input image and its projection into the face space,

$$\epsilon^2 = \|\tilde{\boldsymbol{x}} - \boldsymbol{y}\|^2$$

where

- ϵ^2 is the distance from face space
- $\tilde{\boldsymbol{x}}$ is the original image scaled to the PCA dimensions and normalized subtracting the mean image obtained while generating the PCA projection matrix
- $\boldsymbol{y}$ is the PCA projection of the image $\boldsymbol{x}$, i.e. $\boldsymbol{y} = \phi_{PCA}^T \tilde{\boldsymbol{x}}$

For example, we can estimate the distance of the image to the two subspaces (frontal and profile in our case), so the subspace with the minimum distance is used for the recognition. Besides, if the distance is high (higher than some predefined threshold) for the two subspaces we could estimate that we are not dealing with a face image.

5.5 Classifying

We have introduced in section 4.2 that, due to the fact that the density functions of the groups of samples are unknown, using nonparametric techniques to estimate the densities and building the classifiers seemed suitable for our system. Among the different procedures we could use, we have focused on two of them: the k-NN and the Parzen classifiers. Besides, anticipating the potential problems that we can find with the Parzen classifier in a small sample size situation, we have presented a modified version to try to overcome this issue. In any case, we will evaluate the performance of the system with the three classifiers: k-NN, standard Parzen and modified Parzen.

5.6 Training and testing

According to our application, we can consider a certain number of images to create the initial models; that is, a set of representative feature vectors for every person which will be used as the reference patterns for the posterior recognitions. So, we should take into account how to divide our data into the training (initial models) and the testing mode [5], like the hold-out, cross-validation or leave-one-out methods. In any case, we do not follow strictly this classical view since in a real application the whole data will not be available in the beginning. For that reason, it should be reasonable to use an initial number of frames of the video sequence for training and use the following frames for testing. However, it is also possible to have some initial models previously created in other sessions and used for posterior recognitions. Besides we update the models adding and removing elements, so we have to "train" the classifiers periodically. For example, in the Parzen classifier we will have to estimate the covariance matrix every time we add or remove an element of the model. In addition to this, as we have commented previously, we can create models when the system is working, so we can recognize people not included in the initial models. For that reason we should create temporal models where the faces of the same person are added until there is a number representative enough.

In our case, we can include in the training stage two actions: the generation of the PCA projection matrices and the generation of the initial models. The PCA matrices will be estimated using the XM2VTS database, whereas the testing will be done using the Spanish part of the BANCA database. This is done to reinforce the idea of creating a universal eigenspace, so we will generate the space with totally different faces than those used in the working mode, reassuring their independence. Moreover, we will use different images from BANCA for the training -when we generate the initial models-, and the testing part.

Furthermore, a batch of simulations should be made to obtain the optimal values for the set of parameters of our system, which can also be considered as part of the training stage. Some of these parameters are the size of the images, the number of eigenvectors kept in PCA projection matrix, the number of images per group, the optimal number of nearest neighbours in k-NN classifier, the type of combiner, etc. As we have seen, these parameters are diverse and we will consider them independently, although maybe there is some relation among them. This consideration is made to allow a simpler way to obtain the values, making independent simulations. Then in the testing part we will prove our algorithm evaluating the different available options.

Anyway, we can consider two different scenarios. The first one corresponds to the situation when the number of classes is fixed; that is, the training set is fixed and there are no updates, which matches a classical classification problem. In this situation, PCA and LDA can be used. The second scenario corresponds to the usual way to proceed of the system, when there can be a dynamic number of classes and a varying number of samples per model because of the updates. In this situation, only PCA has been chosen for the reasons previously explained in section 5.3.

For any of these two scenarios, the system can work in two modes, when we perform the recognition of one face or when it is done for a group of images, and now we are going to see their main characteristics.

5.7 Recognition of one face

First, we should consider the simplest situation: perform the recognition of one face in one of the face spaces that we have previously commented (the selection of the space must be made before the classification stage). Briefly, we have to project the face using the corresponding projection matrix (according to the view) and apply one of the classifiers to decide if the face matches one of the existing models, and if it does not match, decide if a new model should be created. We also have to consider the model update; that is, if the face should be added to the model or not.

Another of the important concepts that we have to consider is the difference between identify and verify an image. When a verification is performed, we know which is the a priori model that has to be used for the recognition, so we only have to compare the input image to that model and if the probability is higher than a certain level the *id* is confirmed. In contrast, in an identification we have to recognize the face among the different models, so we should compare the image with the existing models and the model with the higher probability, if it is higher than a certain threshold, is commonly selected. That is why verification is the usual work mode of our system, as we will have an estimated *id* preassigned by the tracking system. However, a basic configuration which forces the identifications is also implemented for testing purposes.

The main operations of the algorithm are described in Fig. 12. First, when a new face with a determined *id* arrives to the system we have to look if there is any model in our local database labelled like the image in order to verify if the face matches its corresponding model. If the verification fails or if there is no model labelled like the input face, we perform an identification in order to check whether there is any model that describes the input face. If a successful identification is done, the system alerts the tracking about an incorrect labelling to suggest a change of the preassigned *id* for the *id* corresponding to the identified model. On the other hand, if the identification fails; that is, there is no match, a new model should be created as, if we consider a perfect recognition system, it means that the input face is not described by any existing model. In any case we will not directly create a new model, because to perform a verification or an identification we will need a minimum number of samples in the model. For example, for the k-NN classifier we will need a minimum of k samples for the model, or for the Parzen classifier we will need a number of

samples at least equal to the dimension of the space to avoid the singularity of the correlation matrices. For that reason we will use two different structures, the group of models used for recognition and a group of temporal models, where these samples should be added. So after a failed identification we will look for a temporal model with the corresponding *id*. If it does not exist, a new temporal model is created, whereas if it exists, the sample is added to the model. Then we have to check if the size of the temporal model has the minimum size, and if this condition is accomplished the temporal model is moved to the group of models used for recognition. Finally we should note that the updates are done after a correct verification or after a failed verification and a failed identification; that is, when the sample is not represented by any model. In this last case the sample is directly added to the model with the preassigned *id*.

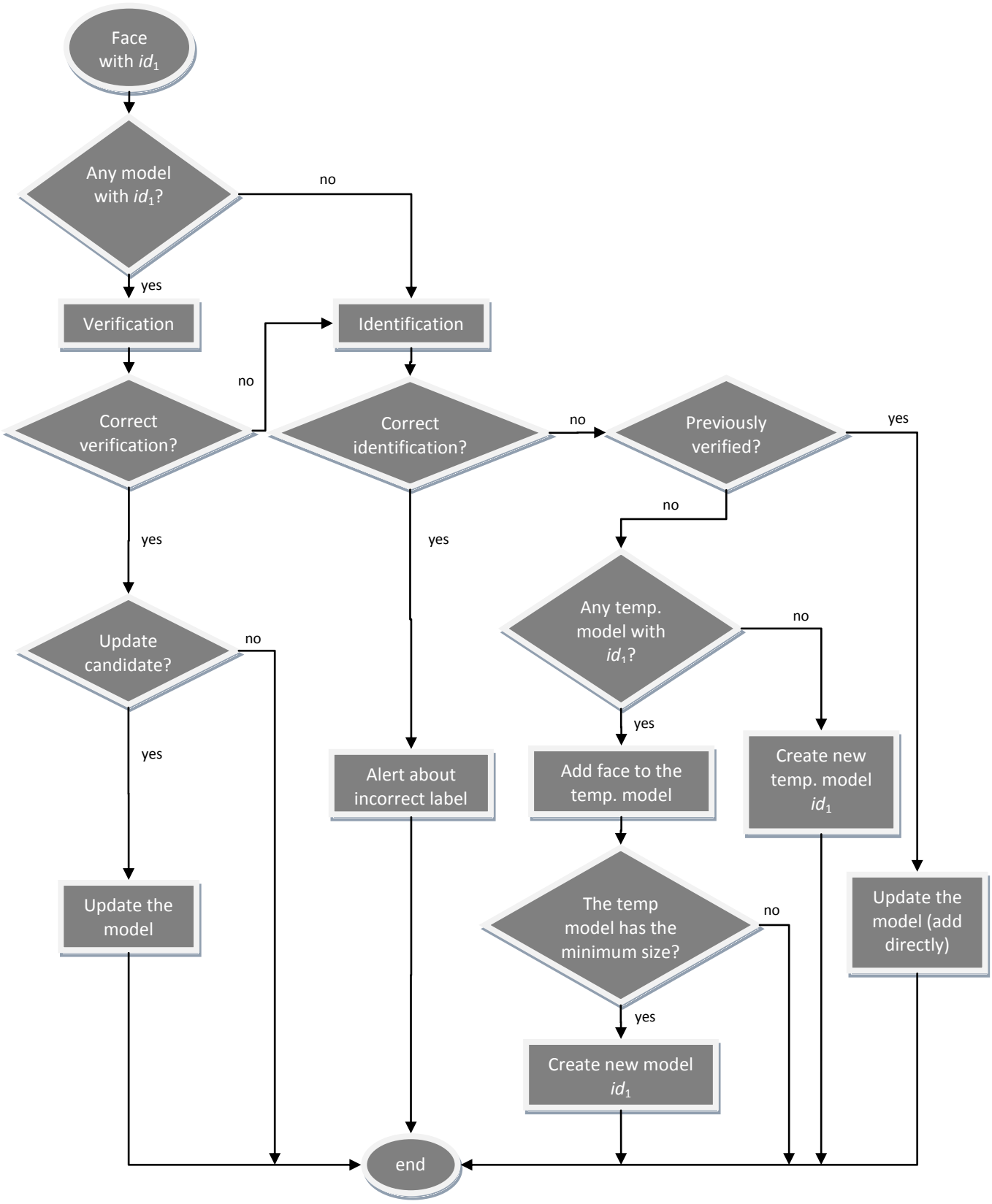


Fig. 12. Description of the algorithm for 1 face

5.7.1 Model update

We have introduced the need of the model update in our solution so now we are going to develop this concept. First we have to consider when we should perform the update, and as we can see in Fig. 12, it is proposed to be made only when a verification is done, as theoretically we are in a more reliable scenario than when an identification is needed.

Then we can think if the sample is appropriate to be added to the model; that is, if it is an update candidate. As we have commented previously, we can think about the images which are not represented by any class and those which are only represented by one class, but we have to define some practical criteria to refine this. With our classifiers, we obtain a list of the probabilities corresponding to the match of the sample with every existing class (or model), and we select the class with the highest probability. Now, we should decide when a sample is not represented by any class, for example by fixing a threshold for all the probabilities. If none of the probabilities exceeds this threshold, we can say that the sample is not represented and it is a candidate to be added to our system. This threshold should be fixed heuristically. In a similar way, if the sample is represented by some classes (i.e. one or more of the probabilities is higher than the threshold) we can define a ratio between the highest probability and the rest. So we divide every probability by the highest one, to obtain a measure of the separation among the classes to see how "much" every class represents the sample. If this ratio is lower than a threshold (different than the previous one, and also estimated heuristically) for the different classes, we could say that the sample is a candidate to be added to the model. If it is higher for one or more classes (excluding the class with the maximum probability which ratio will be 1), as the sample is represented significantly by more than one model, it is not a candidate. With this we are trying to minimize the possibility that one class overruns the others. We should note that the first threshold is the minimum probability that a sample belongs to a class, and as we will see later it is a parameter of the system. Besides, the election of the second threshold seems not a very determining factor, so it is commonly fixed in our simulations to 0.8 (according to this, if there is another class with an estimation for the posterior probability of at least the 80% of the highest value, we can say that the sample is represented at least by two classes).

After that, we have to think about methods to decide among the different candidates if they should be added or not, in order to avoid an unnecessary model growth. We could discuss about different approaches like a geometric or a probabilistic solution. For example, with the k-NN classifier it is clear that a geometric approach can be suitable, because we are considering the proximity of the new samples to the existing ones, so for example the samples which are too close to or surrounded by the existing ones can be avoided. This can be implemented by the cosine-aperture method [24]. On the other side, concerning the Parzen classifier (and also our k-NN solution which provides probabilities) we could consider a probabilistic approach adding those samples that changed the pdf and rejecting those samples closer to the present pdf. Finally, for the vector deletion we could take into account the same ideas that for vector addition or consider a solution based on the vector's life, deleting those vectors that are not used for recognition tasks after a certain time, or for the Parzen classifier use the reduced Parzen to keep a number of representative samples of the class reducing its total number.

In any case, it is important to note that these methods should be applied to the vectors which are only represented by one class because the vectors which are not represented by

any class will be directly added to the model. This is done after an identification (see Fig. 12), to assure that the face does not belong to any of the existing models (according to the ability of the system), so we will assume an optimistic scenario where we will trust the tracking about the identity of the face. It is important to update the models with these faces as we will be adding faces with a strong variation in reference to the existing ones, so the updated models should represent better the variability of the data.

According to its simplicity we will consider the cosine-aperture method for vector addition and deletion, which we are going to detail next. Besides, as we are considering the Parzen classifier we will evaluate the use of the reduced Parzen solution [30] for vector deletion if we use these type of classifier.

5.7.2 Cosine-aperture method

The cosine-aperture method is used to find a measure of how much a sample is placed inside the model. This approach is based on the idea of determining what the sample sees from its position in the hyperspace while looking at the model. If it sees a compact cluster of vectors, it means that the sample is placed on the exterior regions of the model. On the other hand, if it sees a set of vectors in a wide angle, we can infer that it is surrounded and it is placed in the middle of the model in some way. In Fig. 13 we can see two different models, and how the sample $\boldsymbol{x}_1$ is surrounded and should not be added to the model whereas $\boldsymbol{x}_2$ seems a good candidate (the decision regions are represented as circles around the vectors, in our case with higher dimensions we will have hyperspheres)

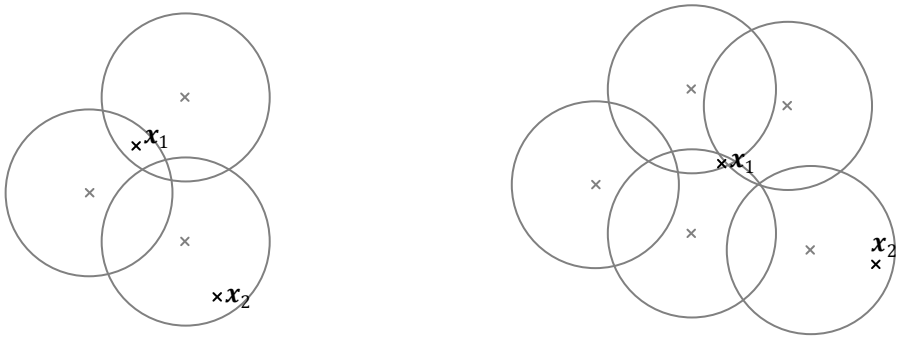


Fig. 13. Examples of candidate samples to be updated to the model

Firstly we should define a reference vector $\boldsymbol{\psi}$ which will be compared with the vectors of the model, and that is the vector between the test sample $\boldsymbol{x}$ and the centroid $\boldsymbol{\mu}$, defined as (the subscript k for the class is discarded for simplicity)

$$\boldsymbol{\psi} = \boldsymbol{\mu} - \boldsymbol{x}$$
$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}_i$$

where N is the number of vectors $\boldsymbol{y}_i$ of the model.

Then, the aperture of a model vector $\boldsymbol{y}_i$ can be defined as the angle between $\boldsymbol{\psi}$ and the vector $\boldsymbol{v}_i$ between $\boldsymbol{x}$ and $\boldsymbol{y}_i$ (see Fig. 14).

$$\boldsymbol{v}_i = \boldsymbol{y}_i - \boldsymbol{x}$$
$$\alpha = angle\ (\boldsymbol{\psi}\ , \boldsymbol{v}_i)$$

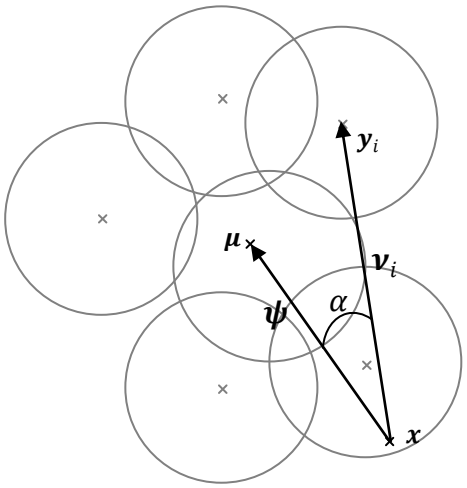


Fig. 14. Intermediate steps to estimate the aperture

In any case, we will consider the cosine of the aperture; that is,

$$\cos(\alpha) = \frac{\boldsymbol{\psi}^T \boldsymbol{v}_i}{\|\boldsymbol{\psi}\|\|\boldsymbol{v}_i\|} = \xi(\boldsymbol{\psi}\ , \boldsymbol{v}_i)$$

Finally the mean of all the aperture's cosines is calculated as

$$\bar{\xi} = \frac{1}{N}\sum_{i=1}^{N} \xi(\boldsymbol{\psi}\ , \boldsymbol{v}_i)$$

This measure is a representation of the compactness of the model according to the position of the sample to analyze. So values near to 1 indicate that the mean aperture is small and the vectors are seen as a compact cluster, whereas values close to 0 indicate that the mean aperture is large and the vectors are located around the sample. Then we can set a threshold value for $\bar{\xi}$, for example $\bar{\xi}_{th} = \cos(30°) \cong 0.866$, so for values of $\bar{\xi}$ lower than $\bar{\xi}_{th}$ we can consider that the sample is relatively separated from the cluster of vectors of the model, so it is low redundant and should be added to the model. On the other hand, for values of $\bar{\xi}$ higher than $\bar{\xi}_{th}$ we can consider that the sample is placed inside the cluster, so it is redundant and should be discarded for the update, and if it is the case, even removed from the model.

In Fig. 15 we can see an example of both situations. On the left, we have a candidate sample which should be added, as it "sees" a narrow model from its situation, whereas on the right the sample should be discarded as it "sees" the model in a wide angle.

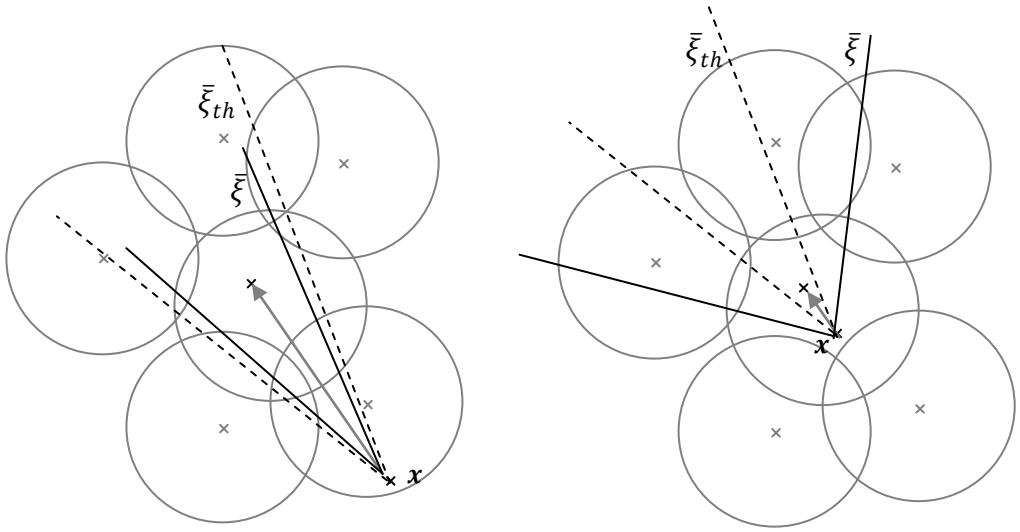


Fig. 15. Example of a candidate to be added to the model (left) or discarded (right) according to the cosine aperture criterion

5.8 Recognition of sets of faces

Now we can consider how to take advantage of the availability of different images of the same individuals to make a more reliable decision.

First of all we have to think about the information that we have and the possible ways to combine it. As the output of our classifier we usually obtain the label of the class or model that best matches our input image, so we make a "hard decision". The most known combiner that we could apply in this case is the majority vote; that is, the most repeated label will be selected. But depending on the classifier we can use more information, as the probabilities obtained with the Parzen classifier to make a "soft decision". In theory, having this extra basis can lead us to a better result. For that reason, different solutions based on the concept of decision profile (DP) [5] are proposed.

Let us suppose that we have $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_L)$, L feature vectors (every vector $\boldsymbol{x}_i$ has a dimension n) obtained from the projection of the ensemble of L faces and $\Omega = (\omega_1 \dots \omega_C)$ the set of labels of the C classes or models. Besides this, we can define $d_{i,j}(\boldsymbol{x}_i)$, which represents the support that the classifier gives to the hypothesis that the face i is represented by the class j (in the case of the Parzen classifier is the estimation of the posterior probability for the class). Then we can define the decision profile matrix (DP) as

$$DP(X) = \begin{pmatrix} d_{1,1}(\boldsymbol{x}_1) & \cdots & d_{1,j}(\boldsymbol{x}_1) & \cdots & d_{1,C}(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots & & \vdots \\ d_{i,1}(\boldsymbol{x}_i) & \cdots & d_{i,j}(\boldsymbol{x}_i) & \cdots & d_{i,C}(\boldsymbol{x}_i) \\ \vdots & & \vdots & \ddots & \vdots \\ d_{L,1}(\boldsymbol{x}_L) & \cdots & d_{L,j}(\boldsymbol{x}_L) & \cdots & d_{L,C}(\boldsymbol{x}_L) \end{pmatrix}$$

So every row is the list of the probabilities that the classifier gives to every image, and every column can be known as the support for every class.

At this point several methods to find the overall support for each class, and therefore assign the label with the highest support, can be found. These methods are commonly split in two groups: "class-conscious", when only one column of the DP matrix is used at a time, and "class-indifferent", when all the elements of the matrix are considered as new features to be processed by another classifier. Now we can present some examples of these techniques.

Firstly we can think of some simple combiners from the group of the "class-conscious", as the non-trainable combiners; that is, that do not need to be trained and the support for the class depends on one column: $\mu_j(X) = f\left(d_{1,j}(\boldsymbol{x}_1), \dots, d_{L,j}(\boldsymbol{x}_L)\right)$. The simplest combination functions that are interesting to our application are summarized in Table 2.

Combination method	Formulation
Simple mean or average	$\mu_j(X) = \frac{1}{L}\sum_{i=1}^{L} d_{i,j}(\boldsymbol{x}_i)$
Maximum	$\mu_j(X) = \max_i \left[d_{i,j}(\boldsymbol{x}_i)\right]$
Geometric mean	$\mu_j(X) = \left(\prod_{i=1}^{L} d_{i,j}(\boldsymbol{x}_i)\right)^{\frac{1}{L}}$
Generalized mean	$\mu_j(X) = \left(\frac{1}{L}\sum_{i=1}^{L} d_{i,j}(\boldsymbol{x}_i)^{\alpha}\right)^{\frac{1}{\alpha}}$

Table 2. Examples of simple class-conscious combiners

About the generalized mean combiner, the parameter α can be considered as the "level of optimism". The minimum combiner is obtained when $\alpha \to -\infty$, which corresponds to the most pessimistic election; that is, we know that all the members of the ensemble support the class at least as much as $\mu_j(X)$. On the other hand the maximum combiner is obtained when $\alpha \to +\infty$, which corresponds to the most optimistic choice; that is, we would accept an ensemble degree of support of $\mu_j(X)$ if at least one member supports the class with that degree. Besides we can obtain other particular cases, like the harmonic mean obtained when $\alpha = -1$, the geometric mean when $\alpha = 0$ and the arithmetic mean when $\alpha = 1$.

Another approach to this problem is the ordered weighted averaging (OWA). First, every column is ordered from higher to lower values. Then a vector $b = (b_1, b_2, \dots, b_L)^T$ with different weights (but summing 1) is defined, and the support for that class is estimated performing the dot product of the two vectors. According to these definition of b, we can obtain, for example, the maximum combiner with $b = (1,0,\dots,0)^T$, or the average with $b = (1/L\,, 1/L\,, \dots, 1/L)^T$.

About the "class-indifferent" methods, we can take into account the decision templates (DT) combiners. Firstly the DT, considered as the most typical DP for each class, is estimated as

$$DT_j = \frac{1}{N_j} \sum_{\substack{z_k \in \omega_j \\ z_k \in Z}} DP(z_k)$$

where N_j is the number of elements of the data set Z from ω_j. Then it is compared with the current DP using some similarity measure. Examples of these measures are the squared Euclidean distance $DT(E)$ or the symmetric difference $DT(S)$.

$$DT(E): \mu_j(X) = 1 - \frac{1}{L \times C} \sum_{i=1}^{L} \sum_{k=1}^{C} \left(DT_j(i,k) - d_{i,k}(X) \right)^2$$

$$DT(S): \mu_j(X) = 1 - \frac{1}{L \times C} \sum_{i=1}^{L} \sum_{k=1}^{C} max \begin{Bmatrix} min\{DT_j(i,k), 1 - d_{i,k}(X)\}, \\ min\{1 - DT_j(i,k), d_{i,k}(X)\} \end{Bmatrix}$$

Also based on this idea of DT, we can find other combiners as the Dempster-Shafer (refer to [5] for further information).

With these premises in mind, we design our system following the same structure that was shown in Fig. 12, but now the process is made not only for a single image but for a group of them. For that reason the system will store the different images in groups according to its *id* until the group is full, and then this group will be processed. Then, we will treat every face individually to obtain a decision profile matrix, and then any of the previously commented rules for the combination of results will be used. We have implemented the basic combination methods shown in Table 2 and also the combiners based on the decision template concept, using $DT(E)$ and $DT(S)$. Anyway we could find problems using the DT approach when the number of classes is variable, as in our case, so although the basic tools have been implemented its full development has been left for a future study.

6 Results

Now we are going to see some results obtained from different simulations. As said before, we have to perform a set of simulations to obtain the optimal values of the different parameters of the system. The best way to get those values would be to generate a matrix taking into account all the different combinations of the parameters, but for simplicity we will consider them independently. Besides doing simulations to obtain these optimal values, other types of simulations should be made to test the performance of the system, simulating the different situations where the system should be used.

Therefore, a batch of simulations should be made to evaluate, among others, the effect of:

- The size of the images.
- Number of eigenvectors kept in PCA projection matrix (dimension of the subspace) and if the first ones should be discarded.
- Minimum probability to consider that one sample belongs to a class.
- Minimum distance to consider that one sample belongs to a determined class.
- Number of images per group.
- Type of combiner to use.
- Parameter α of the generalized mean combiner (if used).
- Number of eigenvectors kept in LDA projection matrix.
- Number of faces in initial models.
- Type of classifier (k-NN, Parzen or Modified Parzen).
- Parameter k (number of nearest neighbours in k-NN).
- Type of estimation for the parameter h (width of the kernel in Parzen and Modified Parzen classifier): a fixed value for every class, the "Rule of Thumb" or using a multiple of the standard variation of every class.
- The multiplier for the standard variation (for the estimation of h, if used).

As we want the simulations to be as independent as possible, firstly some typical values will be used and then substituted for the optimal values. However, other characteristics tested here cannot be considered as parameters to optimize, and they will show us the behaviour of the system under different conditions that we could not control in a real application (for example the size of the images, the availability of enough training faces, etc).

For that reason, the simulations will be made starting from the initial values shown below in Table 3. We have chosen these values considering that they should not interfere with the parameter to evaluate and their effect will not mask the results. We should note that we will be commonly dealing with frontal faces, unless we specify that they are profiles.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1639	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	150	Yes	1-NN	1	0.5

Table 3. Initial values taken for the simulations

In this example table we can see that we will perform a test considering 52 different individuals, generating their initial models using 8 images (2 from every session) from the BANCA Spanish controlled images. Besides we will test the system with the rest of the available images from the same database, 1639 in this case. The rest of the parameters are quite self-explanatory, indicating if there are labelling errors in the test image list, the size of the faces, the type of projection (PCA or LDA), the number of kept eigenvectors (that is, the size of the image space), the use of the mask (only for the frontal faces), the type of classifier, the size of the group (1 in this case, if it is greater we will indicate the type of fusion) and the minimum probability to consider that a sample belongs to a class.

As we have commented previously, there are two possible scenarios. The first one is found when the number of classes is fixed generating previously the initial models, where we should see more clearly the behaviour of the classifiers. With this controlled scenario we will analyze the effect of the different parameters and the different options of our system, using both the BANCA and the XM2VTS databases (basically XM2VTS to generate the PCA projection matrices and BANCA Spanish for the testing). In addition to this, we will be able to perform tests like comparing the performance of PCA versus LDA using the optimal parameters obtained before.

The second scenario corresponds to the usual working mode of the system, with updates and the possibility of creating models on the fly. Here we will test the performance of the whole system in different situations, for example with other sequences and not only from the known databases. For that reason we will divide this section into two parts corresponding to these two scenarios, the first one where we will evaluate the various parameters to see their effects, and the second one where we will test the whole system with the optimal values for the parameters found before and we will evaluate specifically the effect of the updates and the results achieved with “real” data.

Anyway we should introduce the evaluation tools that we will use to measure the performance of our solution. As the output of the system we obtain an estimated *id*, after a successful verification or identification, or an indecision, when there is no match after an identification. This indecision label can be obtained when the person is new to the system, so we should create a new model, when it is not represented by any model, so it should be added to the corresponding model, or when the system fails. In any case, this indecision label does not mean that the system does not have an associated behaviour, only that the result obtained after a verification or identification by the system is not conclusive. That is, according to the capabilities of the system we cannot assure the identity of the face to recognize, but trusting the information given by the tracking system we can perform actions like creating a new model or directly adding a new sample to the related model.

These results can fit one of these four groups: (i) true positive (TP), when a face is correctly labelled with its actual *id*; (ii) true negative (TN), when an indecision is marked and the face does not correspond to a known person, that is, with a model in the system; (iii) false

positive (FP), when the estimated *id* and the actual *id* are different; and (iv) false negative (FN), when an indecision is labelled but the face has an *id* which corresponds to an existing model.

Then, with these values we can estimate different parameters, as the true positive rate (TPR, also known as sensitivity) where TPR = TP / (TP + FN), or the false positive rate (FPR) where FPR = FP / (FP + TN). Finally, with the true positive and false positive rates, we can draw the ROC (Receiver Operation Characteristics) curve. This is a graphical characterization of a classifier system, in which the TPR is plot versus the FPR when a discrimination threshold is varied. With this graphical help, we can easily compare the behaviour of different systems and also analyze the optimal point of the compromise between false positives and false negatives. That point is the EER (Equal Error Rate), the point where 1 – TPR = FPR; that is, the point of the curve that cuts with the line $1 - x$.

Anyway, sometimes the ROC curve does not represent clearly the behaviour of the system, for example when we analyze some parameters and the system only works in a limited part of the curve. In addition to this, if we want to use the ROC representation to optimize some parameters we should perform an extensive set of simulations to generate the different curves, and these parameters can be found with other tools with a lower time and computational cost. For these reasons the performance of the system can be also modelled by some typical parameters, which are the ones that we are going to use, as FAR (False Acceptance Rate, the number of mislabelled faces, i.e. the false positives, divided by the number of faces processed), FRR (False Rejection Rate, the number of not recognized faces unless there was model, i.e. the false negatives, divided by the number of faces processed) and TAR (True Acceptance Rate, TAR = 1 – FAR – FRR).

However, as we have commented previously we are not rejecting any input face in practice, because although an indecision is marked for a determined face, that face is added to its corresponding model or used to create a new model if its *id* is new to the system. In any case, if there are no labelling errors, the images directly added to an existing model should be considered as false negatives, as they are images not represented by any class although they should be by one, and the images used to create the initial temporal models should be considered as true negatives, because the input faces do not correspond to persons with an existing model in the system.

The different situations that we can find in our system are summarized in Table 4.

		Correctly labelled (match)	Wrongly labelled (no match with an existing model)
Verification		TP	FP
Identification		TP	FP
Indecision	Directly added to an existing model	FN	TN
	Directly added to create initial temp models	TN	FN

Table 4. Possible results of the system

In any case, we are going to see an extensive set of simulations, but we can give some clues of the main aspects to point out. We can divide these aspects into three groups.

First we are interested in knowing the differences using our three different classifiers, trying to find out their strong and weak points. Second, we want to analyze the influence of the characteristics of the models, according to their number and the type of images, and also the effect of the model update. Finally we want to know the improvement achieved when we use groups of images.

However, the tests are very extensive and although we have tried to make them as independent as possible, it is common that different parameters and characteristics were involved in the same simulation, so the results should be taken globally. Anyway, the main results will be summarized in the conclusions section.

6.1 Evaluation of the parameters

As commented before, we have to note that unless it is clearly specified we are dealing with frontal faces. The performance of the system with profile faces is evaluated in section 6.1.11. Besides in section 6.2, when we were considering "real data", we use both views: frontal and profile.

6.1.1 Evaluation of the minimum distance

This is an evaluation of the minimum distance to consider that a sample belongs to a class. This is not a real parameter in our system and its effect is only evaluated to see if the results are similar to our intuition; that is, for lower values of the distance most of the faces should be rejected and for higher values they should be accepted. For that reason a set of simulations is made following the parameters of Table 3 and without considering the updates; that is, simply rejecting the indecision results.

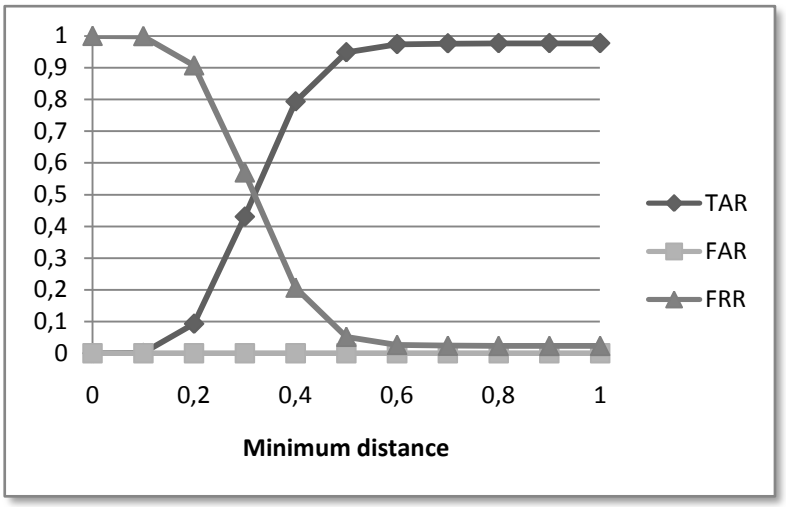


Fig. 16. Evaluation of the minimum distance

In Fig. 16 we can see that the results are close to the expected, and how the optimal value for the distance is approximately 0.5, when we obtain an approximately constant TAR ≈ 0.97. We should also note that the possible distances range from 0 to 1, as we are dealing with a normalized space, and how the FAR remains constant in 0; that is, there are no wrong matches.

6.1.2 Evaluation of the number of kept eigenvectors in k-NN

Now we can see some simulations made to evaluate the effect of the number of eigenvectors kept in the PCA projection matrix, which is equivalent to the dimension of the face space. For that reason, a first PCA matrix is generated, using 1180 faces from the XM2VTS database and their mirrored versions (a total of 2360 faces), scaled to a size of 10x15 pixels. Then, we create different matrices selecting a different number of first eigenvectors. We should note that we use the mirrored images as we need a minimum number of faces equal to the number of eigenvectors to generate, which is also equal to the

size of the images; that is, for the 10x15 images we need at least 150 images to generate the PCA matrix with a maximum of 150 eigenvectors. As we can see this number is not necessary in this case, but it will be for the bigger images of 40x60 to obtain a number close to the theoretical 2400 images needed. In Fig. 17 we can see that the optimal value of kept eigenvectors in this case is approximately 50, from a total of 150. Besides, we can see how our algorithm tends to reject people rather than to obtain a false acceptance.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1639	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
10 x 15	PCA	?	Yes	1-NN	1	0.5

Table 5. Parameters for the evaluation of the number of kept eigenvectors

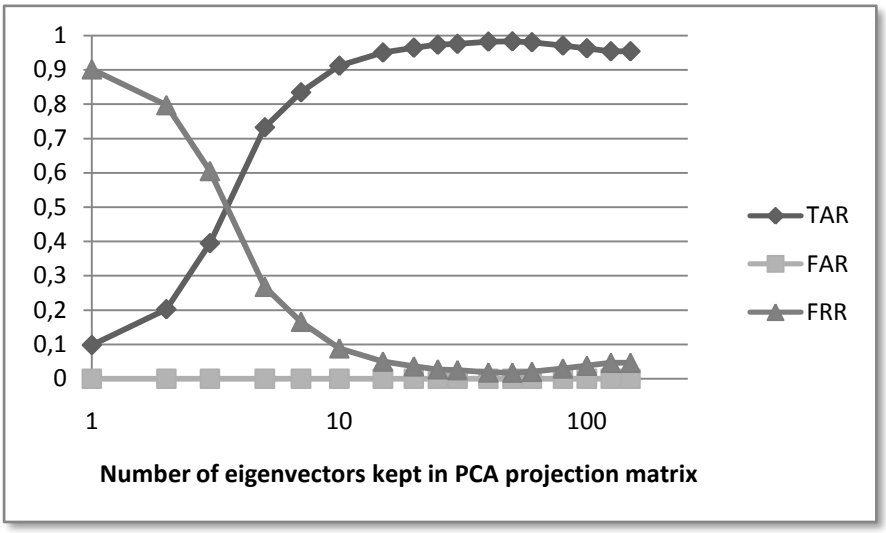


Fig. 17. Evaluation of the number of eigenvectors kept in PCA matrix

Now we can perform the same experiment but now changing the size of the images to one of the three possible sizes, 10x15 for the lowest recognisable faces, 20x30 for an intermediate approach and 40x60 for the biggest possible faces commonly found in our scenarios. Following the same procedure than before we can obtain the results in Fig. 18 and with more detail using a logarithmic scale in Fig. 19 (only the TAR is considered to make the graphics clearer).

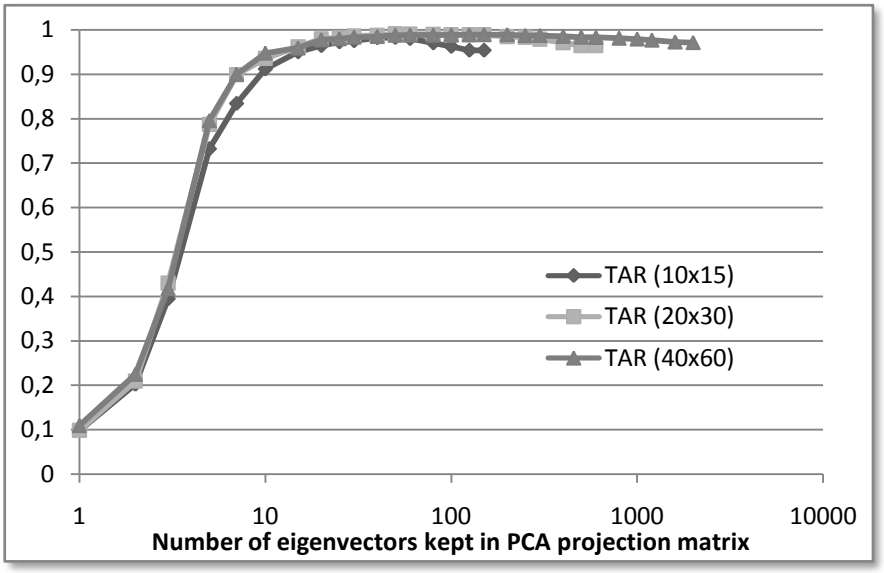


Fig. 18. Evaluation of the number of kept eigenvectors according to the size of the faces

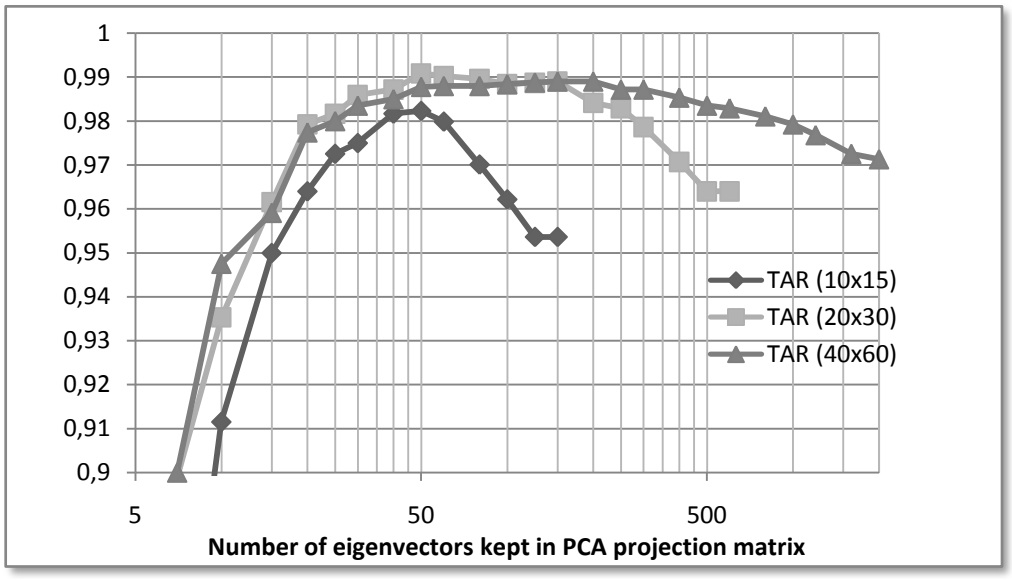


Fig. 19. Detail of the evaluation of the number of kept eigenvectors according to the size of the faces

As we can see, the results improve as the size of the images increases, but between 15 and 150 eigenvectors the TAR obtained with images of 20x30 is slightly better than the obtained with the faces of 40x60. Finally, it is interesting to calculate the percentage of the optimal number of kept eigenvectors according to the total of eigenvectors. If the optimal value is taken as the mean value of eigenvectors that gave a TAR higher than 0.98, we can obtain the next results summarized in Table 6:

	Size		
	10x15	20x30	40x60
Optimal # of eigenvectors	50	160	415
Total # of eigenvectors	150	600	2400
Percentage	33.3%	26.6%	17.3%

Table 6. Optimal number of kept eigenvectors for different sizes

We can observe that the percentage decreases as the size grows, so we can suppose that a minimum number of eigenvectors are needed to represent the data, and if the number

increases, as the complexity of the space grows exponentially, as we had seen in the curse of dimensionality, the extra dimensions do not help on the classification task. However, in the literature we can find typical values for the percentage of eigenvectors of the 40 or 50 %, so we should try to evaluate its effect in other situations.

For that reason, now we are going to make a similar experiment but using the degraded images of the BANCA database. The local models are generated with the controlled images as in the previous simulation, and now the minimum probability to consider that a sample belongs to a class is reduced, as we are in an adverse scenario and the obtained probabilities will be lower.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1928	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
?	PCA	?	Yes	1-NN	1	0.2

Table 7. Parameters for the evaluation of the number of kept eigenvectors in a degraded scenario

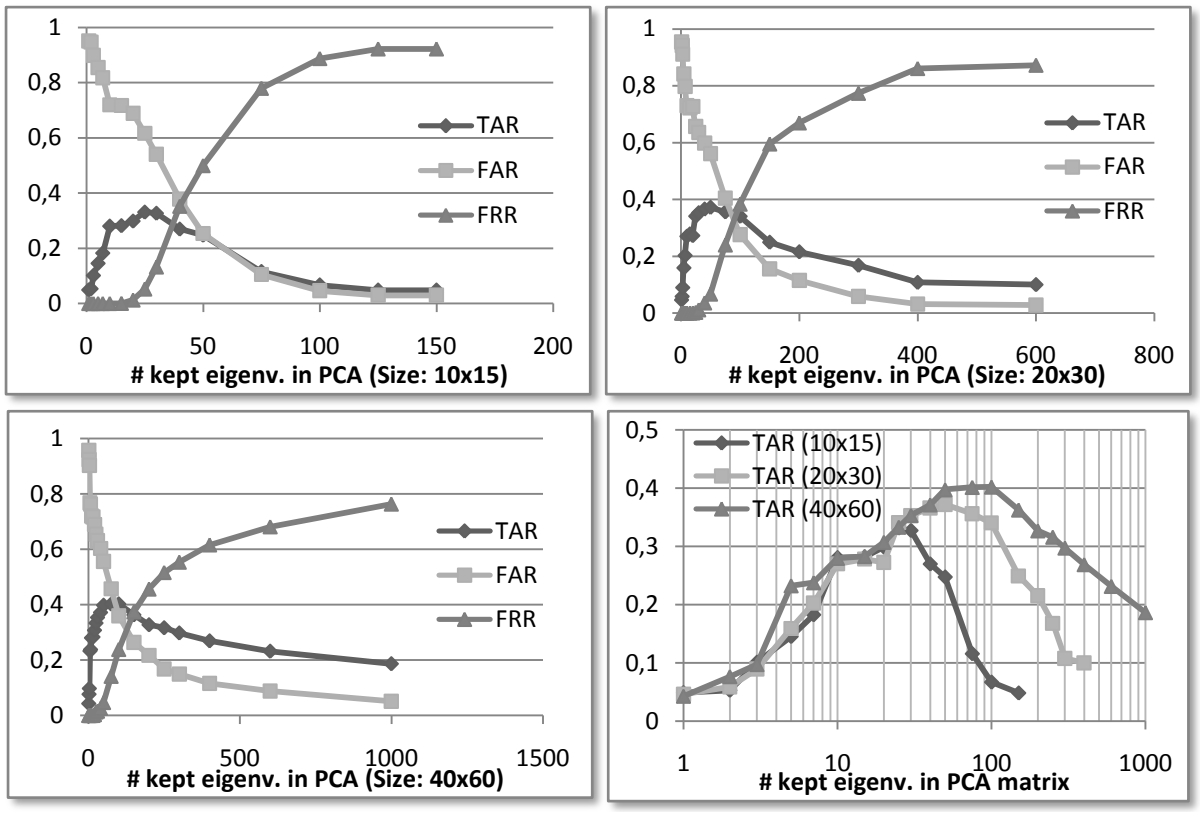


Fig. 20. Evaluation of the number of kept eigenvectors according to the size of the faces in a degraded scenario

In Fig. 20 we can see the different graphics according to the size of the images and a final graphic with the TAR of the three cases. The first remarkable aspect to consider is the reduction of the performance, achieving as highest TAR a value of 0.40 whereas, with the controlled images, values of 0.99 were reached. Anyway, the tendency of the TAR curves is

similar to the previous ones; that is, it rises from the lowest values of eigenvectors, reaches a maximum and then decreases. Following a similar criterion than before, we can estimate the optimal number of kept eigenvectors, which are summarized in Table 8.

	Size		
	10x15	20x30	40x60
Optimal # of eigenvectors	20	50	80
Total # of eigenvectors	150	600	2400
Percentage	13.3%	8.3%	3.3%

Table 8. Optimal number of kept eigenvectors for different sizes with degraded images

We can see that the optimal number of eigenvectors is lower than the obtained with the controlled images, but the results are clearly worse. We can guess again that the increase of the dimension of the space has a negative effect on the recognition rates, which is more patent as the degraded images to recognize have more clear differences than the controlled ones. In any case, this is a first estimation for the recognition rates of the degraded faces as these rates should be improved by using other tools, like using a higher number of neighbours or using some of the degraded images to create the initial models.

According to the results, we can propose the use of approximately the 20 or 25% of the eigenvectors when we deal with controlled images, whereas when the images used for the generation of the initial models and the testing are different (controlled and degraded respectively in this case) we should use about the 5 or 10%.

6.1.3 Evaluation of masking the faces

We had intuitively anticipated that masking the frontal faces with an oval mask should help on our recognition task, as we are rejecting most of the hair and the background and keeping almost all the face. In Table 9 we can see the parameters (the different parameters for the two experiments are put in different rows) used for the simulations and the results are shown in Table 10.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1639	BANCA SP controlled	No
52	8 (degraded)	1543	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	200	?	1-NN	1	0.5

Table 9. Parameters for the evaluation of masking the faces

	Controlled		Degraded	
	Mask	No mask	Mask	No mask
TAR	0.984137	0.959121	0.937135	0.880752
FAR	0.006711	0.001220	0.033701	0.014906
FRR	0.009152	0.039658	0.029164	0.104342

Table 10. Results with masked or unmasked faces

We can observe that the results have improved when we use the masked images, especially for the degraded faces. We should also note that the results obtained for the degraded images are clearly better than the ones obtained in the last evaluation in section 6.1.2 just using images from the same type; that is, a part of the degraded faces, to generate the initial models.

In any case, we are only taking into consideration well aligned and exactly cut faces, as the cutting was done manually, so the effect of the masking could be different if the faces are not exactly frontal or if the cutting is not very accurate. For that reason we should evaluate the effect of masking the frontal faces with non-optimal images, which is what we are going to do next when we test the system with "real" data in section 6.2.2.

6.1.4 Evaluation of discarding the first eigenvectors

Now we are going to analyze the effect of rejecting the first eigenvectors of the PCA projection matrix; that is, the first eigenfaces. As we have commented before, this technique is used to reduce the effect of the illumination changes, so we will test it for the controlled and the adverse images, to see how it affects to uniformly illuminated images and faces with illumination changes.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1639	BANCA SP controlled	No
52	8 (degraded)	1543	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
-	PCA	-	Yes	1-NN	1	0.5

Table 11. Parameters used for the evaluation of removing the first eigenvectors

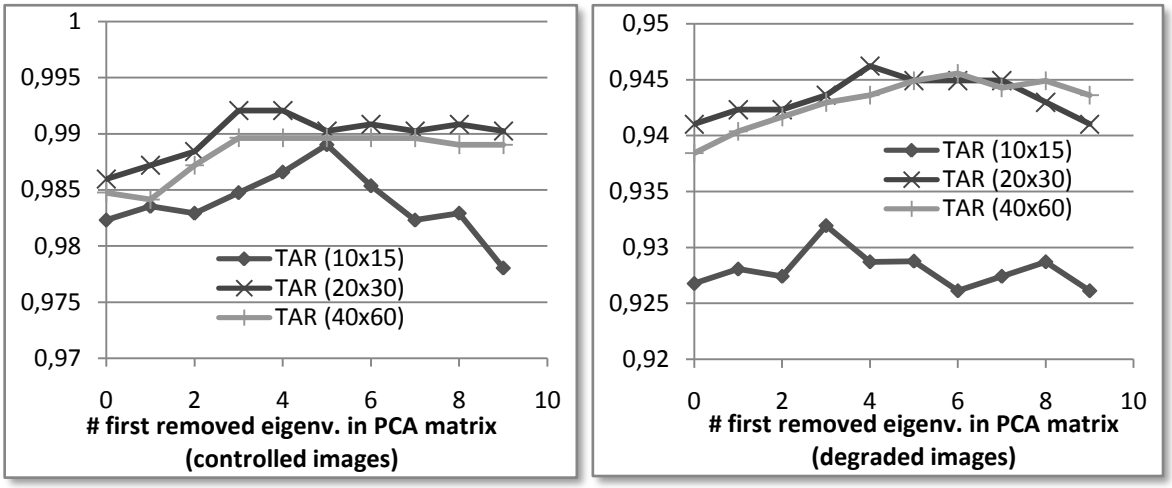


Fig. 21. Evaluation of the number of first eigenvectors removed in the PCA projection matrix

We can see in the graphs that the results improve slightly by removing some of the first eigenvectors in PCA. All the curves show a similar tendency in which the TAR increases initially and then progressive and slowly decreases (except some results with the 10x15 degraded images). With this technique we can achieve an enhancement of approximately the 0.6% by removing the first 4 or 5 eigenvectors. This result is close to what we have expected but its effect should probably be more manifest with faces with greater illumination changes, for example when the source of light is not as uniform or it presents changes from side to side that generate shadows.

6.1.5 Evaluation of the characteristics of the initial models

We will now analyze the performance of the algorithm when the number of vectors in the initial models is varied. Here we can consider two different aspects, the similarities between the images and the number of images. In our database we have 4 sessions for every type of images (that is, controlled, degraded or adverse), recorded in different days, and within each session we have 40 images of every person. Therefore, we have a big variation among the images from different sessions (as the images are taken in different days, there can be changes in the appearance), and a lower variation among the images of the same session (when there is only a slight variation in pose or facial expression). For that reason, we will analyze the effect of using images from different sessions or from the same session and the number of them.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	? (controlled)	1639	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	150	Yes	1-NN	1	0.5

Table 12. Parameters used for the evaluation of the size of initial models

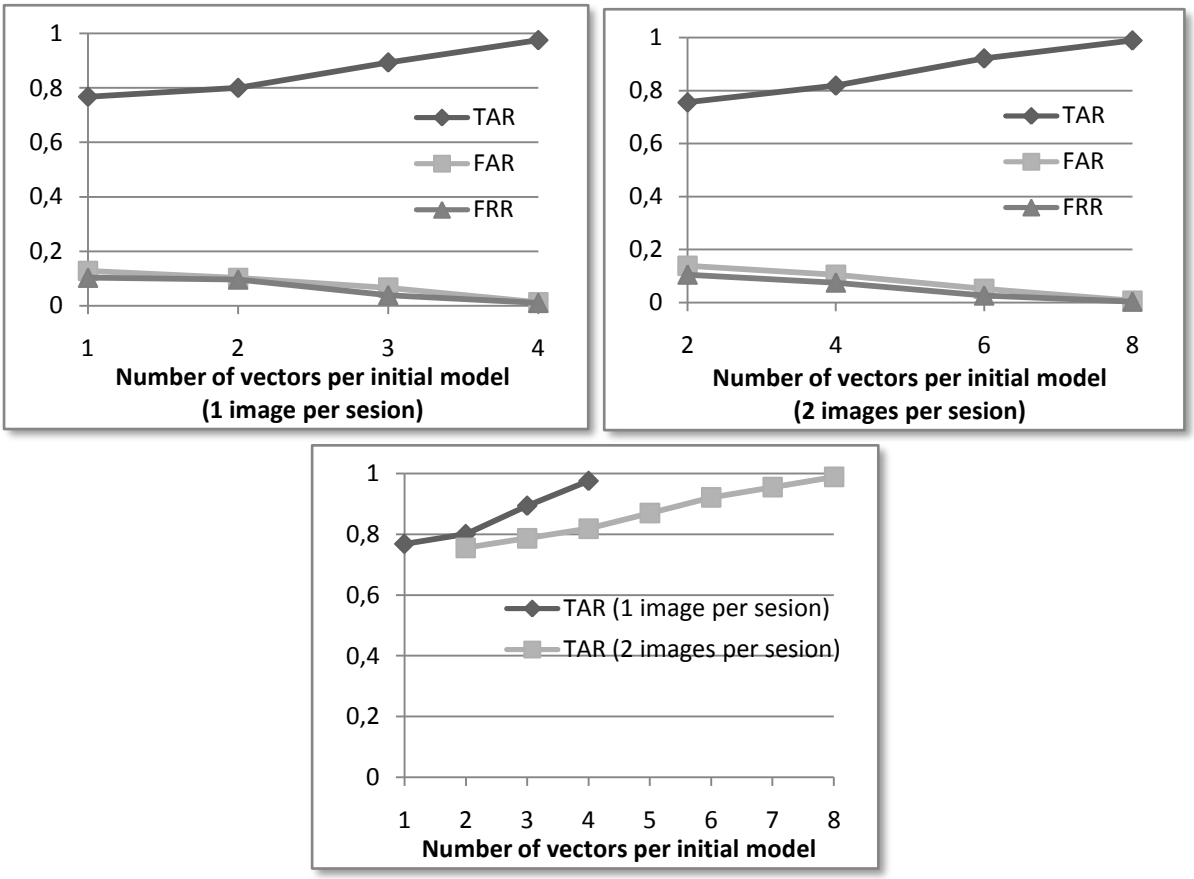


Fig. 22. Evaluation of the number of vectors per initial model. On the top left, one image per session is used. On the top right, two images per session are used. The comparison of the two scenarios is shown below

As we can see in the last chart, it is more useful to employ faces from different sessions than to use more faces from the same session. This is a result that we should expect, because it is important to use the set of faces for the initial models that best represent the global variability of the faces that we are trying to recognize. However, using images from the same session we improve the recognition rate; that is, the results using two images are slightly better than the obtained using one. This is due to the fact that we are modelling the variations of the faces within that session. However, it is clear that it will be harder to characterize the faces of session 4, for example, with initial models generated from the other sessions than if we use faces of session 4 in the initial models. Anyway, using one single image from session 1, we can achieve approximately a 77% of correct recognitions of images of the four sessions.

In addition to this we can remark the effect of using images from different types for the generation of the initial models and for the testing. In section 6.1.2 we have seen that if we use controlled images for the generation of the initial models and degraded faces for the testing we achieve a higher TAR of 0.40 whereas values of 0.99 can be found if the controlled images are used for both tasks. On the other hand, in section 6.1.3 we can see simulations using degraded faces for the training and the testing where we can find values of 0.9 for the TAR. With these results we should note that the faces used in the models have to be representative enough of the faces that the system will try to recognize later to obtain the best performance. Besides, with this behaviour we can get a justification for the need of the model update, as we are trying to capture the global variability of the faces and add it to the corresponding model.

6.1.6 Evaluation of the number of nearest neighbours in k-NN

Now we are going to test the performance of the algorithm when the number of nearest neighbours in the k-NN classifier changes. We will test it in first instance with the controlled images.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1639	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	150	Yes	?-NN	1	0.5

Table 13. Parameters used for the evaluation of the number of nearest neighbours

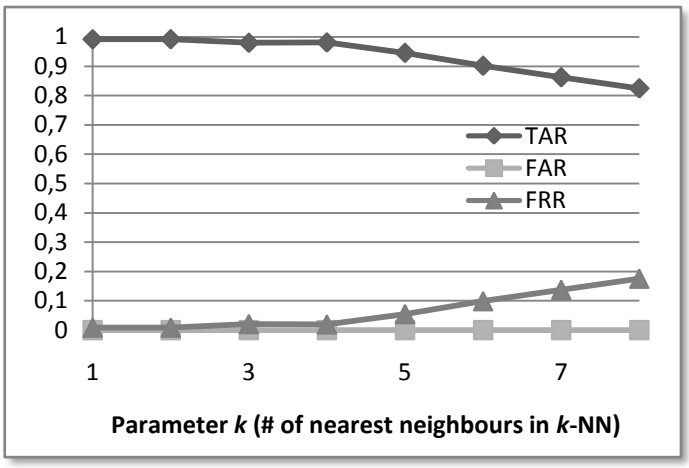


Fig. 23. Evaluation of the number of nearest neighbours in k-NN

In Fig. 23 we can see that as the number of neighbours is increased, the TAR decreases. This is due to the characteristics of the classifier, because when the number of neighbours grows it is more common that those neighbours correspond to different classes which will lead to lower estimations of the posterior probability, which in occasions can be lower than the minimum probability and in consequence, the corresponding faces will be rejected. We should also note that the even values will lead to possible ties, although in our algorithm are solved using the estimation of the probability based on distances, so we can see that the better results are obtained with one and three neighbours. In any case, this behaviour can be partially avoided by lowering the minimum probability as we are going to see next using faces from the adverse scenario.

However, firstly we will analyze why we should reduce this minimum probability. For that reason, we will perform a set of simulations using controlled images for the generation of the initial models and degraded for the testing, with the parameters shown in Table 14, and the results in Fig. 24.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1928	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	-	Yes	?-NN	1	0.5

Table 14. Parameters used for the evaluation of the number of nearest neighbours with degraded images

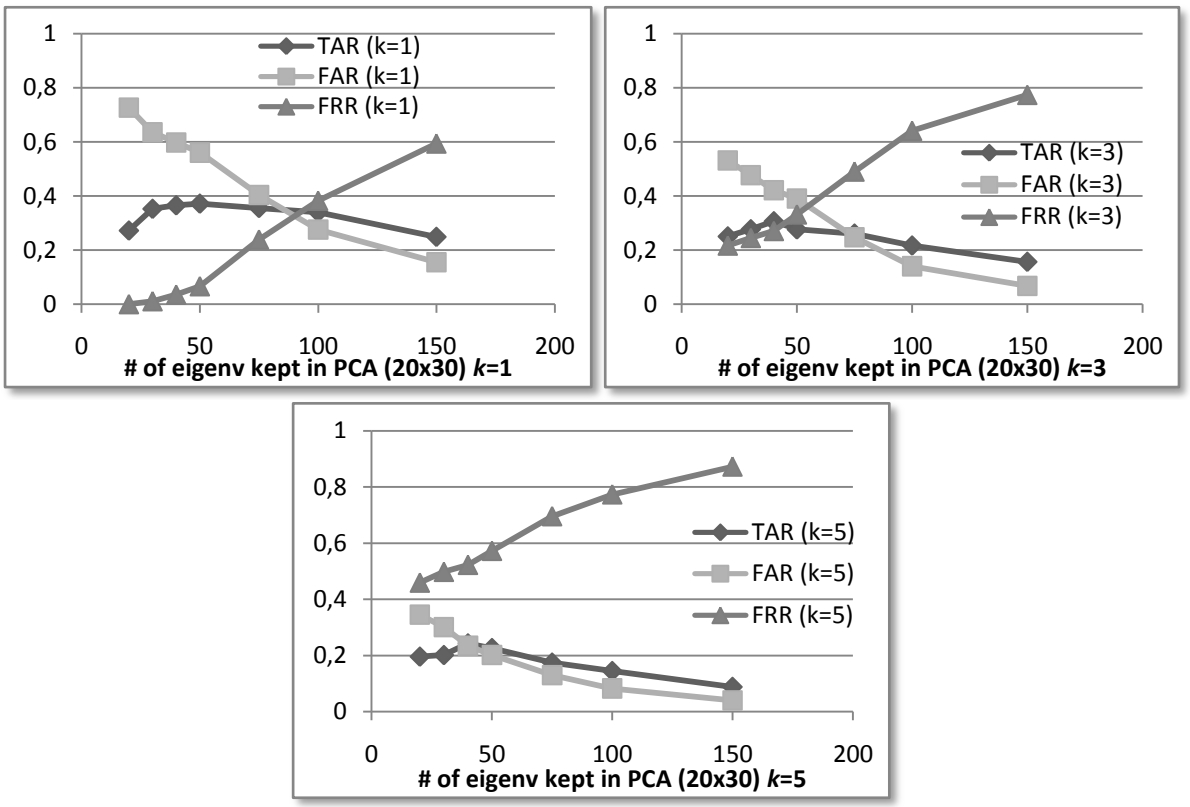


Fig. 24. Evaluation of k and the dimension of PCA in 20x30 degraded images

From these graphs we can see how when we increase the number of neighbours used in the classifier the rejection rate drastically increases too. We suspect that this behaviour is due to the way that the classifier (the soft k-NN) estimates the probabilities using the inverse of the distances of the nearest neighbours. This is also confirmed looking at the values of these probabilities, which lower their values when the number of neighbours is increased, from initial values about near 0.9 to values of 0.3 or even lower. For that reason it is reasonable to evaluate the effect of using a reduced minimum probability, so we will use some simulations made previously to determine the optimal number of eigenvectors kept in the PCA projection matrix in which we also vary the parameter k. The evaluations are made with the three typical image sizes. In Table 15 we can find some of the parameters used for these simulations and in Fig. 25 we can see the behaviour of the system for the intermediate images (we could generate similar graphs for the other sizes). As we can see it is hard to have a clear view of the performance of the system, so in Fig. 26 we will focus on the evaluation of the TAR for the different sizes.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1928	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
-	PCA	-	Yes	?-NN	1	0.2

Table 15. Parameters used for the evaluation of the number of nearest neighbours with degraded images and a low minimum probability

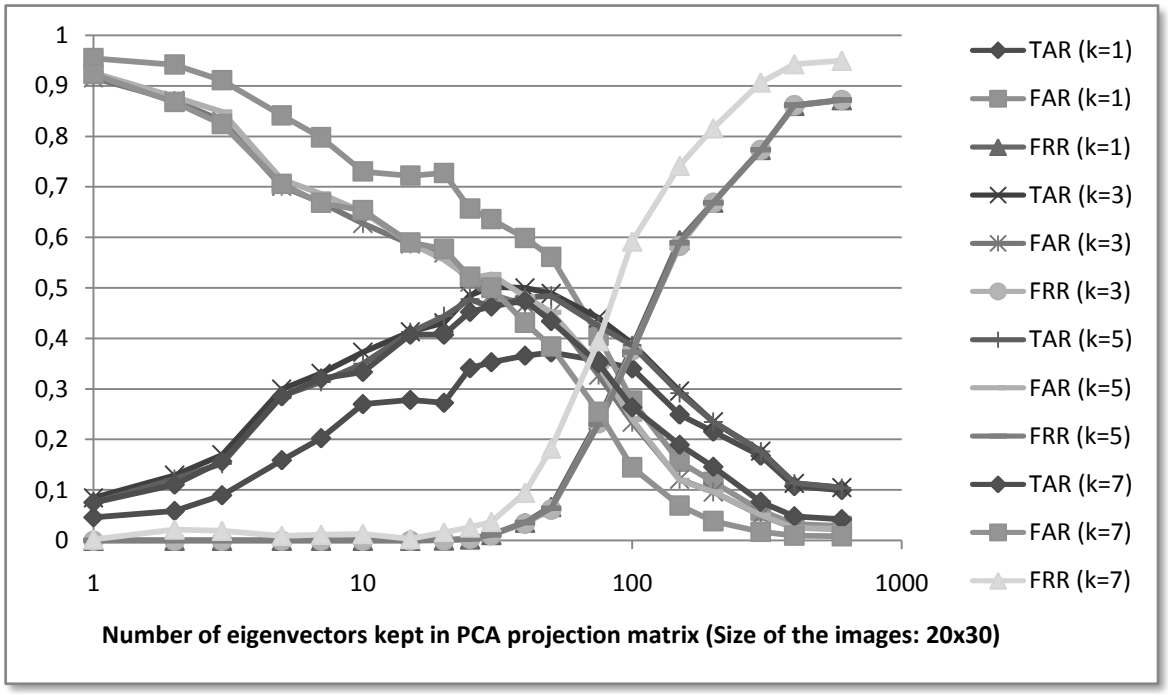
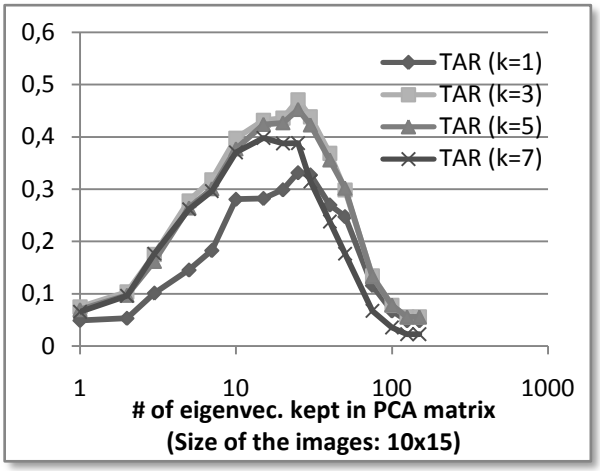
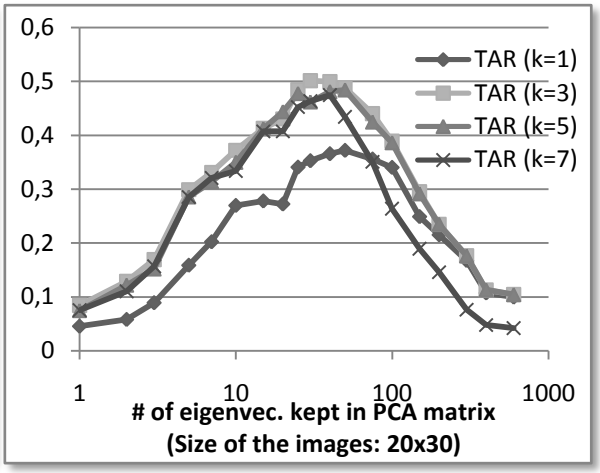


Fig. 25. Evaluation of k and the dimension of PCA in 20x30 degraded images and a low minimum probability





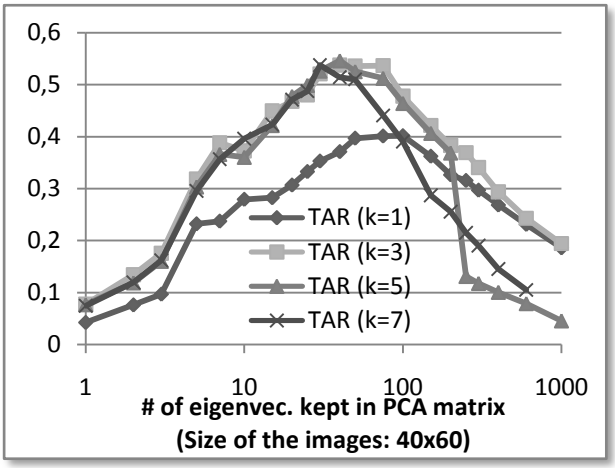


Fig. 26. Detail of the evaluation of k and the dimension of PCA for different sizes

It is observed that the best results are obtained using $k = 3$ and $k = 5$ (with a similar behaviour for every size), whereas for $k = 1$ the results are clearly poorer. In addition to this, we can see how the results obtained for $k = 7$ are slightly worse than the best ones.

In any case, we have previously seen that the tendency was the opposite when we were using the controlled images; that is, that the best results were achieved for the lower values of k, and as the parameter was increased the results were slightly getting worse. As we have commented previously, the intuition says that this behaviour is caused by the way that our algorithm estimates the posterior probabilities in our soft k-NN classifier, so to minimize the effect we will perform a new simulation but now lowering even more the minimum probability that determines if a sample is considered as belonging to a class (in these case only for the intermediate images, as the tendency will be similar for the other sizes). With these premises, we can see in Fig. 27 how the best results are achieved with the higher number of neighbours (in this case seven neighbours because we have models initially with 8 vectors), and that these results are better than the ones obtained previously (for the intermediate images, a maximum TAR of 0.65 versus a TAR of 0.5 obtained for $k = 3$ with a minimum probability of 0.2).

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1928	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	-	Yes	?-NN	1	0.001

Table 16. Parameters used for the evaluation of the number of nearest neighbours with degraded images with a very low minimum probability

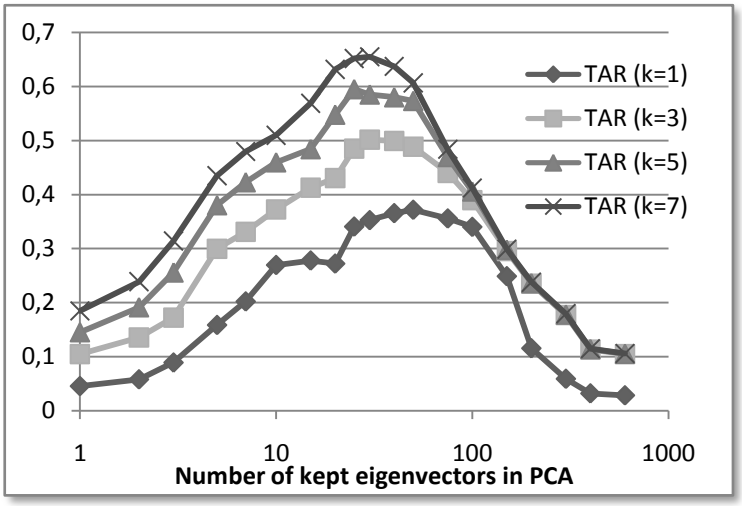


Fig. 27. Evaluation of k and the dimension of PCA in 20x30 degraded images using a very low minimum probability

Anyway, by lowering the minimum probability we are forcing our system to not reject any image and identify any face with an existing class. So, depending on the characteristics of the application field of the system we could apply this approach or not.

Furthermore, these results highlight two important aspects of our solution. The first one is that although we have been talking about the posterior probabilities as the output of the classifiers, in practice we are considering them as confidence levels when we are making the decision about if a face belongs to a determined class or not, so the interpretation of the minimum probability threshold can be taken more widely and not be restricted to a probability of 0.5 as we can initially think. The second concept is that some of the parameters and the results, in consequence, are dependant of the characteristics of environment, for example, if the testing images are similar to the ones in the initial models the best results are achieved with 1-NN, whereas if they are degraded it is better to use a 3-NN or a 5-NN, or play with the values of the minimum probability. This suggests us that an extensive and automatic training considering all the possible parameters should be taken into consideration for a real application.

6.1.7 Evaluation of the Parzen classifier

Concerning the Parzen classifier we can see that there are two main parameters that are determinant when we use this classifier. The first one is the size of the face space; that is, the number of kept eigenvectors in the PCA projection matrix. As we have commented before, since we are estimating the covariance matrices corresponding to every class from a limited number of training samples, we will need a number of vectors to generate the initial model at least equal to the dimension of the space so that the matrix will not be singular and it will be possible to invert it. According to this condition, it is clear that we will not be able to move on the number of eigenvectors used in k-NN, so we should use lower values. According to the other simulations and the number of images available in our databases, we commonly use a maximum number of eight vectors. Besides, as we use these vectors to estimate the covariance matrix of their respective classes, the vectors should be as independent as possible as we need to invert the matrix, so its dimension should be

contained to not increase excessively the complexity.

The second concept is the estimation of the optimal value for the parameter h, the window width of the Gaussian. As we have commented before, the election of h can drastically affect the performance of the Parzen classifier. For that reason different solutions have been implemented. The first idea was fixing a value for h for all the classes and try different values. Another solution is the estimation as $h = 1{,}06\,\sigma\,n^{-0{,}2}$ (where σ is the standard deviation of the class, and *n* is the number of vectors of the class). Finally, a solution where the multiple of the standard deviation ($k\sigma$) of every class is selected, and the multiplier k is set as a parameter.

Taking this into account, we should perform a set of simulations considering that these two parameters are related. First we will perform some simulations to see if the optimal value of h is very variable or if it remains approximately constant for our data. For that reason we will perform the simulations using the parameters of Table 17 (we will use only 10 classes to reduce the simulation time).

About the different methods to estimate the window width, the second one was the first to be discarded as we always get null probabilities. About the other two, the results were clearly better for the last technique where h is a multiple of the standard deviation and not a fixed value, although there is another parameter to optimize in our system. This can be due to the fact that with the last option we are selecting values of h of the same magnitude order of the standard deviation of every class. In contrast, when we select a fixed value it is harder to find the optimal value as we can easily pass from the optimal value and this value can be very different among the classes, whereas with the other option we keep it controlled with the parameter k but variable.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
10	8 (controlled)	314	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	3	Yes	Parzen	1	0.5
10 x 15	PCA	5	Yes	Parzen	1	0.5

Table 17. Parameters used for the evaluation of the parameter k in the estimation of h

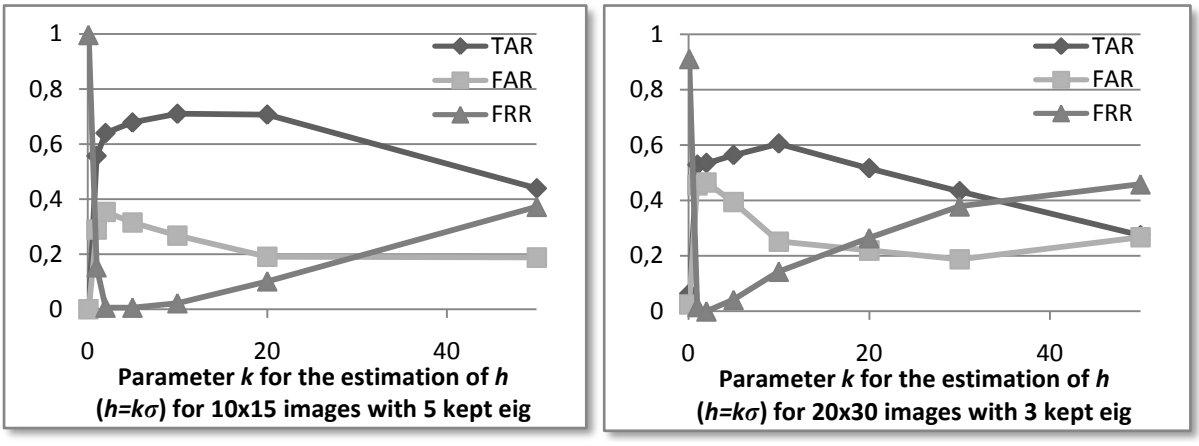


Fig. 28. Evaluation of the parameter k in the estimation of h

We can see in Fig. 28 that the optimal value in these two different scenarios is approximately obtained with a value of $k = 10$, so this value will be taken as the initial value for our posterior simulations. Now using the parameters shown in Table 18 we will analyze the effect of the dimension of the space for the Parzen classifier (we will also use the parameter $k = 10$).

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
10	8 (controlled)	314	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
-	PCA	-	Yes	Parzen	1	0.5

Table 18. Parameters used for the evaluation of the number of kept eigenvectors in Parzen

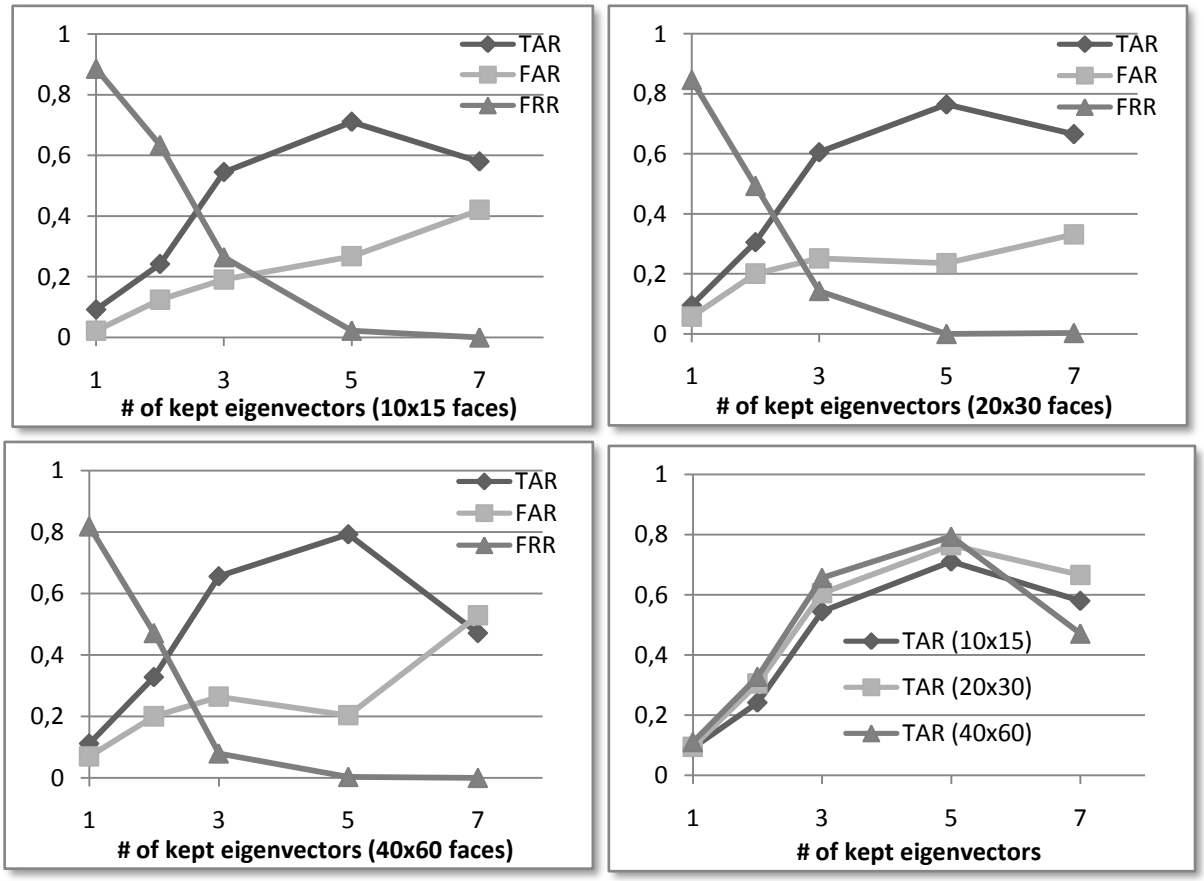


Fig. 29. Evaluation of the number of kept eigenvectors for the Parzen classifier

In Fig. 29 we can see three simulations using different image sizes, and we can see on the last graph that the best results are achieved for a dimension of the face space of 5 (the maximum was 8 as this was the number of initial vectors). We can also observe that the best TAR is obtained with the bigger faces, although the results are only slightly better than the obtained with the intermediate faces. Furthermore, we should note that the performance is clearly worst than the obtained with the k-NN classifier, with maximums of 0.8 versus 0.99.

6.1.8 Evaluation of the modified Parzen classifier

One of theoretical advantages of using the modified Parzen classifier is that it is less sensitive to the high dimension of the space and the small sample size problem when we estimate the sample covariance matrices. For that reason, we will perform the same simulations than with the Parzen classifier, varying the number of kept eigenvectors; that is, the dimension of the face space, for the different sizes. The parameters will be the same that the ones used before in Table 18, but now using the modified Parzen classifier.

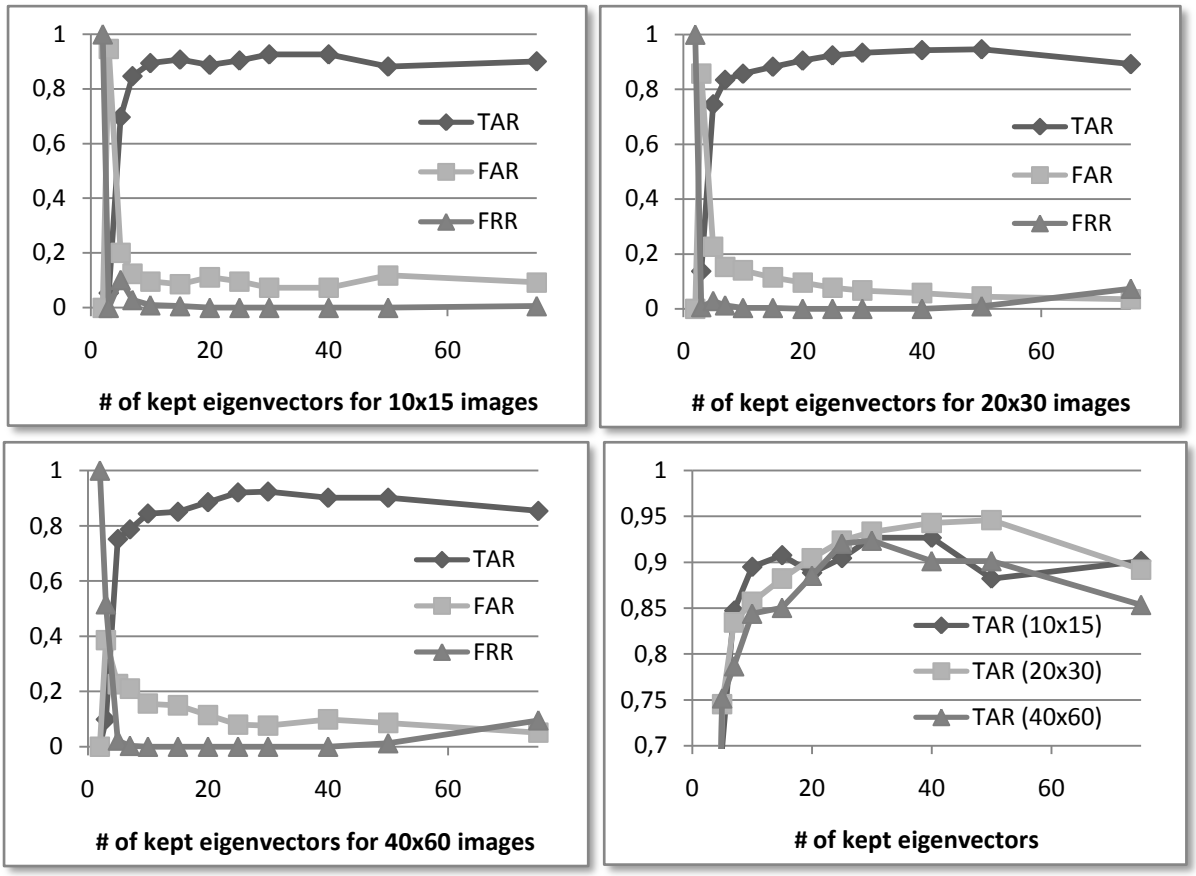


Fig. 30. Evaluation of the number of kept eigenvectors for the modified Parzen classifier

The first aspect to notice in Fig. 30 is that, due to the estimation of the covariance matrices using the Toeplitz technique, the dimension of the space is not constrained by the number of vectors used to estimate the matrices, so in practice we will be able to use higher dimensions than in a standard Parzen classifier. In addition to this, we can see that the results are clearly better than the obtained with the Parzen classifier, achieving a higher TAR of nearly 0.95 using 50 eigenvectors with the intermediate images. Concerning the sizes, the obtained rates follow a similar tendency, achieving the maximum between 25 and 50 eigenvectors, and then lowering their value.

Anyway, we should perform some simulations using all the images, so we will use the parameters that have given the best results previously, which are summarized in Table 19. The simulation will be made varying the parameter k used in the estimation of the window

width ($h = k\sigma$), around the initial value found before of $k = 10$. Besides, an equivalent simulation using the standard Parzen classifier will be performed to try to compare the results obtained with these two techniques.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1639	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	50	Yes	Mod. Par.	1	0.5
20 x 30	PCA	5	Yes	Parzen	1	0.5

Table 19. Parameters used for the evaluation of the modified Parzen classifier for all the controlled images

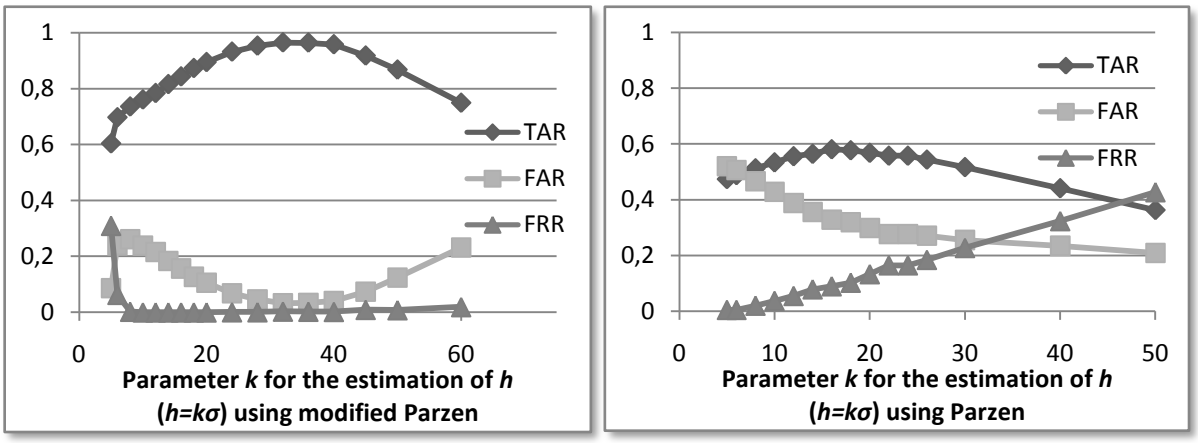


Fig. 31. Comparison of the Parzen (on the right) and the modified Parzen (on the left) classifier using all the controlled images

In Fig. 31 we can see that the performance of the modified Parzen classifier is clearly better than its equivalent using Parzen. As we have commented before, the possibility of using a higher dimension of the space seems determinant for the improvement, but in any case, using a 5 dimensional face space with the modified Parzen we can achieve (see Fig. 30) a TAR of approximately 0.75, clearly higher than the maximum obtained with the standard Parzen classifier. Anyway, one possible way to increase the TAR using the Parzen classifier is reducing the minimum probability to consider that one sample belongs to a class. If we look at the graph on the right, we can see that when the window width grows the FRR increases (we can find probabilities about 0.3 and 0.4), so reducing the probability we force the system to identify every face with an existing class, achieving a possible higher TAR of 0.75. Another aspect to consider is that the optimal value for the parameter k ($h = k\sigma$) using the modified Parzen is approximately 32, a bit far from the initial value of $k = 10$ considered before, which emphasizes the need of the parameter optimization considering more than one parameter at a time. Besides, we can see that this higher rate (a TAR about 0.95) has improved clearly from the obtained with Parzen and is closer to, but still lower than the obtained with k-NN (nearly 0.99).

Finally, we should consider the performance of the modified Parzen classifier when we are testing the system with the degraded images, and compare it with the results obtained

previously with k-NN. Three different scenarios are considered depending on the images used to create the initial models, part of the degraded images, of the controlled ones or from the controlled faces but using a very low probability to consider that a sample belongs to a class (tested before with 0.001). The main parameters for the simulations are summarized in Table 20 and the results are represented in Table 21 and Table 22, where also some of the main parameters which had to be optimized are shown.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1928	BANCA SP degraded	No
52	6 or 8 (degraded)	1543	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	50	Yes	Mod. Par.	1	-
20 x 30	PCA	-	Yes	$\boldsymbol{k}$-NN	1	-

Table 20. Parameters used for the comparison of the modified Parzen and the $\boldsymbol{k}$-NN classifiers using degraded images

	Initial models (mod Parzen)		
	Controlled	Degraded	Controlled (low min prob)
TAR	0.374595	0.887881	0.478937
FAR	0.625405	0.108231	0.521063
FRR	0	0.003889	0
Optimal k ($h = k\sigma$)	60	28	60
Min. prob.	0.2	0.5	0.001

Table 21. Evaluation of the modified Parzen classifier using degraded images

	Initial models (k-NN)		
	Controlled	Degraded	Controlled (low min prob)
TAR	0.501556	0.937135	0.654564
FAR	0.487552	0.033701	0.335062
FRR	0.010892	0.029164	0.010373
Optimal k (# of NN)	3	1	7
Min. prob.	0.2	0.5	0.001

Table 22. Evaluation of the $\boldsymbol{k}$-NN classifier using degraded images

Comparing both tables we can see how the results obtained with the k-NN classifier are globally better than the obtained with the modified Parzen, especially when we use the controlled images to generate the initial models, where we can pass from a TAR of 0.37 to 0.5, which can be a critical aspect for a real scenario; that is, when we try to recognize people in different conditions than the ones used when the person is declared to the system (when generating its model).

6.1.9 Evaluation of using sets of faces

Now we are going to see the results obtained using sets of images with the different fusion methods. We will test the algorithm using the degraded faces, because if we use the controlled images we can already obtain good results using single images, so we will analyze an unfavourable scenario where the effect of using sets of faces can be clearer. We have to test two main aspects, the number of images per group and the type of combiner. For that reason, we have performed a set of simulations following the parameters shown in Table 23, where the number of images is taken as the parameter to vary. A different graph is created for every combiner; that is, classical mean, maximum, geometric mean or generalized mean, in this case with a parameter $\alpha = 0.5$, which corresponds to an intermediate option between the classical and the geometric mean.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1928	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	50	Yes	3-NN	?	0.001

Table 23. Parameters used for the evaluation of the number of images per group

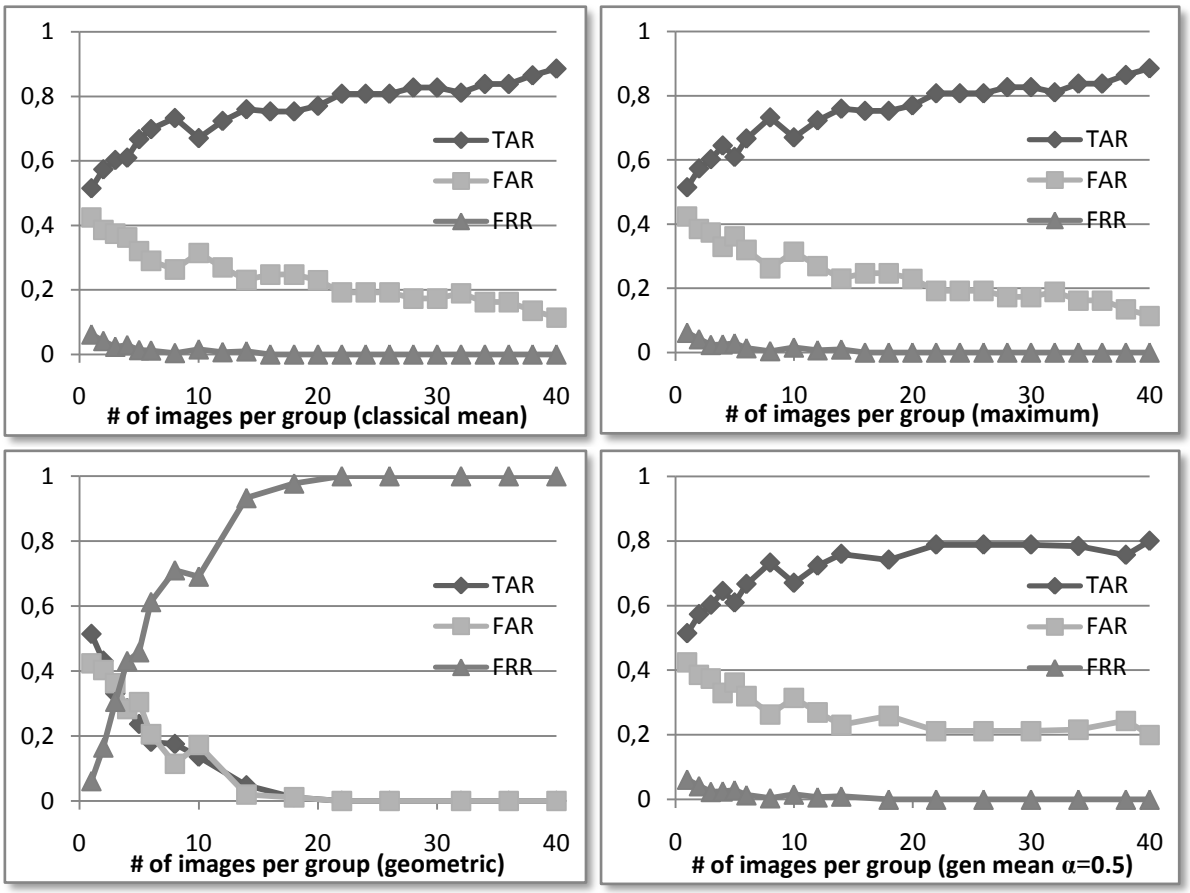


Fig. 32. Evaluation of the number of images per group and the type of combiner

In Fig. 32 we can find the different results obtained using our four combiners. The first aspect to notice is that using either the classical mean or the maximum, the results are clearly improved from a TAR about 0.5 to almost 0.9 when we use groups of 40 images (the biggest possible groups as that was the number of available faces from every person). Besides, we can see that the obtained results improve progressively with the number of images; that is, that according to the number of available samples it seems appropriate to use as many images as possible. In any case, depending on the scenario we will not have long bursts of faces from the same person, for example if there are many people on the scene, so in order not to introduce a long delay (as we have to wait to have a full group to perform the recognition) the number of images in a group should be reduced. A possible value can be eight faces, that as we can see from the graphs shows a peak in the tendency and will not introduce a long delay (for example, if the system provides faces at a rate of 4 faces per second, we will have to wait 2 seconds for the recognition using groups of 8 faces).

Another characteristic that we can see from the third graph is that using the geometric mean the TAR quickly decreases and the FRR increases. This is due to the fact that some samples are not identified with any model, so its probability is zero, and if some of these samples are part of the group, the combiner provides a null probability and in consequence the number of indecisions grows.

Finally we can see that the results obtained with the generalized mean using a parameter $\alpha = 0.5$ are slightly worse than the obtained for the classical mean or the maximum for the higher number of images per group (until 12 images the results were very similar). In any case, we can vary these parameter, considered as a level of confidence, but the results suggest that the performance will be at best similar to using the most common classical mean, so a priori it seems that the type of combiner is not a very decisive factor, but clearly rejecting the geometric combiner.

6.1.10 Evaluation of LDA

At this point we have seen the behaviour of the system in many situations varying the numerous parameters and considering the different options, but always using a fixed number of classes and generating a priori the initial models. As we have commented before, with these premises we would be able to use either PCA or LDA as feature extractor, so we are going to see the performance obtained with these two techniques using the optimal values for the parameters when possible.

Nevertheless, we have to consider some limitations about LDA (note that we refer to the combination of the PCA and LDA techniques as explained in 4.1.2, but we commonly talk about LDA) as commented in [23] to avoid the singularities of the within-class and the between-class scatter matrices. There are two main recommendations which affect to the number of images used and the dimension of the spaces. The first one is that the dimension of the first PCA matrix (used to initially reduce the dimension of the space) has to be lower than the total number of images used to create the initial models minus the number of initial models. The second one is that the dimension of the LDA projection matrix has to be lower than the number of initial models minus one, and at the same time lower than the dimension of the PCA matrix found before.

Firstly we will perform some simulations using the intermediate faces. As we have typically made simulations using 8 images for every one of the 52 initial models, the dimension of the intermediate PCA space should be lower than 364, and in our case is fixed to 150 (close to the optimal value obtained previously). Using the parameters shown in Table 24 we will evaluate the effect of the number of kept eigenvectors in the LDA projection matrix; that is, the dimension of the final face space, and we will compare it with the results obtained using the classical PCA approach as we have doing thus far.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1639	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	LDA	150 (PCA) ? (LDA)	Yes	1-NN	1	0.5
20 x 30	PCA	?	Yes	1-NN	1	0.5

Table 24. Parameters used for the evaluation of LDA

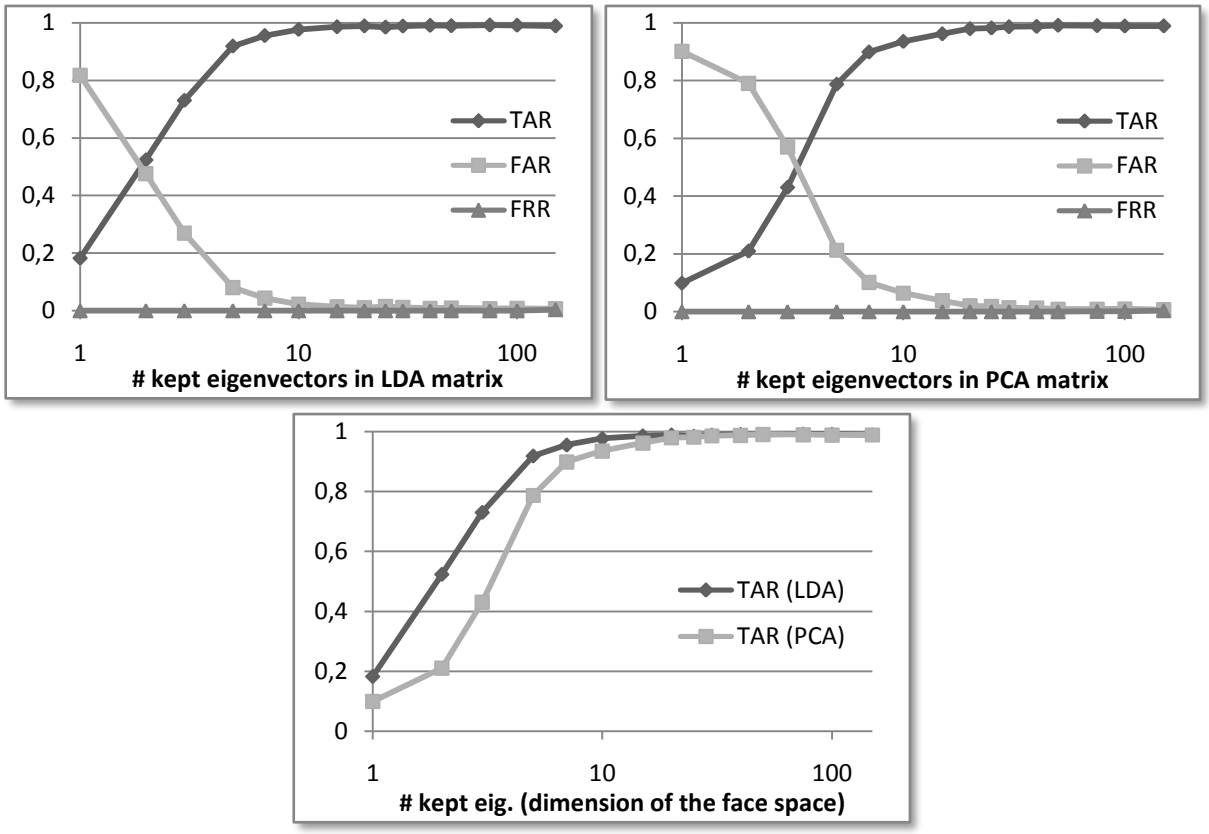


Fig. 33. Evaluation of the number of eigenvectors kept in LDA (top left), PCA (top right) and the comparison of both (down)

We can see in Fig. 33 how the TAR is significantly better for LDA over PCA for the lowest dimensions and how with about 20 eigenvectors or more the rates are similar and very close to 1. This experiment shows how the behaviour of both techniques is similar for the usual

number of eigenvectors, but for lower values LDA outperforms PCA, at least using the controlled images and the 1-NN classifier. Nevertheless, we should test it using other types of images to see if the tendencies are similar. For that reason, now we are going to test it with the degraded images, using the parameters shown in Table 25.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	8 (controlled)	1928	BANCA SP degraded	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	LDA	150 (PCA) ? (LDA)	Yes	3-NN	1	0.2
20 x 30	PCA	?	Yes	3-NN	1	0.2

Table 25. Parameters used for the evaluation of LDA with degraded images

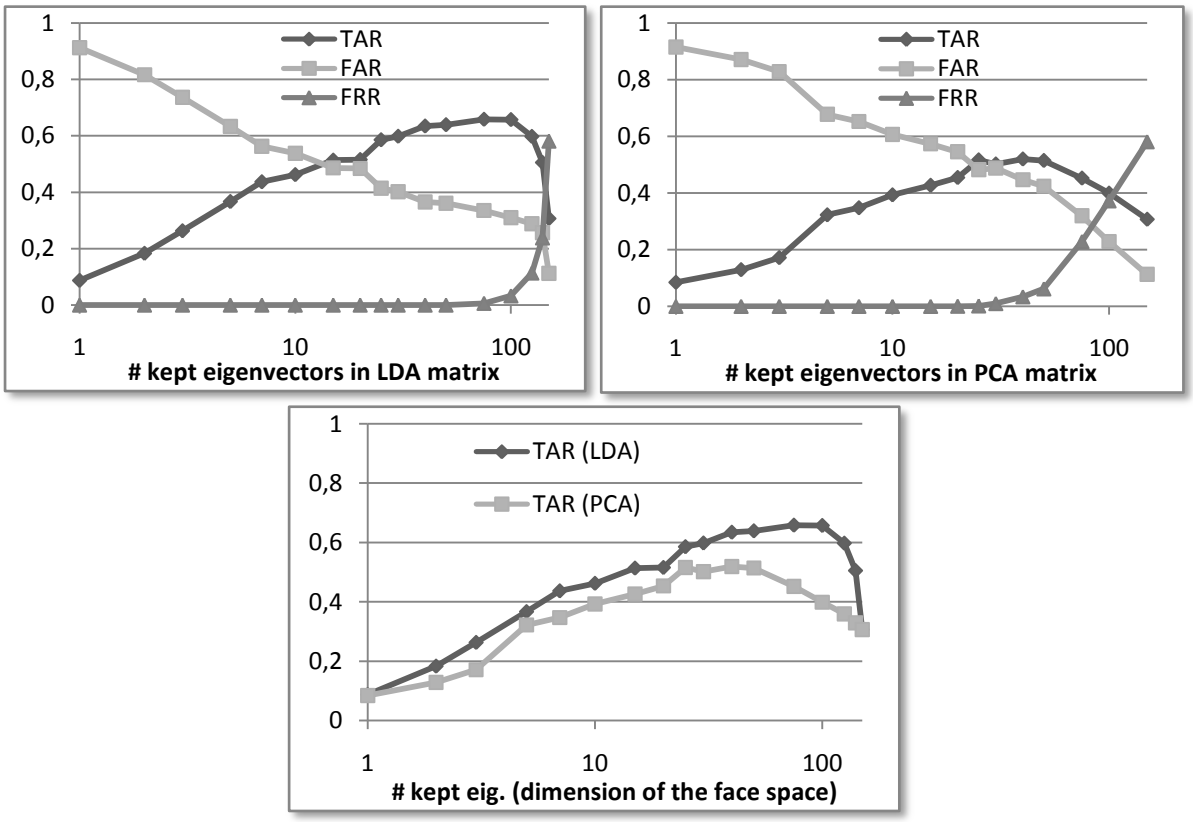


Fig. 34. Evaluation of the number of eigenvectors kept in LDA (top left), PCA (top right) and the comparison of both (down) using degraded images

Now again, if we look at the graphs in Fig. 34 we can see how LDA outperforms PCA, achieving a higher TAR for all the equivalent dimensions of the face space until a maximum of 0.66 with 75 eigenvectors versus a maximum of 0.52 with 40 eigenvectors using PCA. We should also note how the value of the TAR quickly degrades when we are approaching to 150, as that was the dimension of the intermediate PCA space used in LDA, and how the values for the three rates are exactly equal at this point for PCA and LDA.

6.1.11 Evaluation of the profile faces

Up to now we have tested the performance of our system and the effect of the different parameters using frontal faces. At this time, we are going to evaluate it with profile faces. With that purpose we will use the XM2VTS database, as in the BANCA database we do not have profile views. In any case, we are not going to make an extended evaluation as with the frontal faces, and we will focus only on the main aspects.

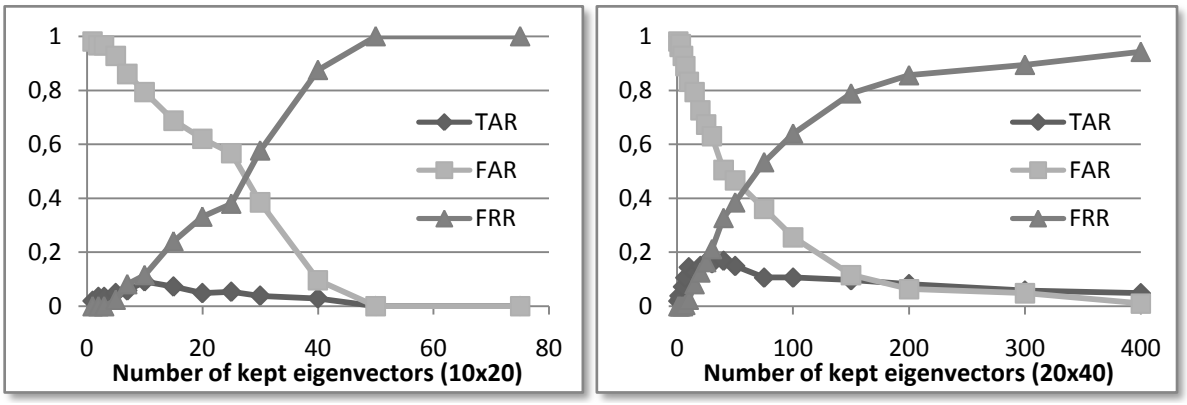
The first aspect to consider is that we should divide our database (1957 images) into three parts, one to generate the PCA projection matrix, the second to generate the initial models and a final one for the testing if we want to keep the different tasks as independent as possible. Initially we will use 1378 (corresponding to the individuals from the number 75 until the 371) to generate the PCA matrix and the rest (we will use images from 52 different people, as in the simulation made with the BANCA database) for the other tasks. Anyway, as we only have 8 images from every person, the number of images dedicated to the testing and the generation of the initial models will be smaller than with the frontals (where we have 40 images of every person).

The first simulation is made to evaluate the effect of the number of kept eigenvectors according to the size, following the parameters shown in Table 26.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	4	208	XM2VTS profiles	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
-	PCA	-	No	1-NN	1	0.5

Table 26. Parameters used for the evaluation of the number of kept eigenvectors with the profile faces



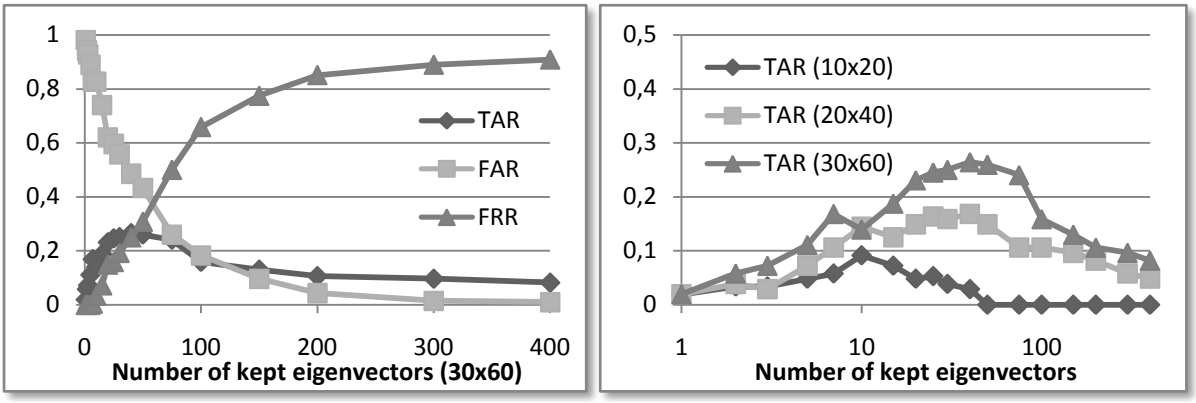


Fig. 35. Evaluation of the number of kept eigenvectors for the profile faces

As we can see from the graphs in Fig. 35 the first remarkable aspect is the low performance with the profile faces whereas using similar parameters with the frontal faces we can achieve a TAR about 0.9 and now we only obtain maximums of 0.26 (using 40 eigenvectors with the bigger images). In addition to this, we can see how the FRR quickly increases, and in contrast to the frontals, the probabilities obtained are zero so we cannot reduce the probability threshold as we had made before. For that reason, one of the possible ways to improve the performance is increasing the minimum distance to consider that one sample belongs to a class to force the algorithm to try to match the face with an existing model and try to reduce the possible indecisions. This is not a parameter that we have varied before as we fix it to 0.5, but it seems that the distances between the vectors are greater than in the frontal face case, so we are going to test it again using the parameters shown in Table 27 (only the bigger faces are considered this time).

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	4	208	XM2VTS profiles	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
30 x 60	PCA	-	No	?-NN	1	0.2

Table 27. Parameters used for the evaluation of the number of kept eigenvectors with the profile faces using a high minimum distance

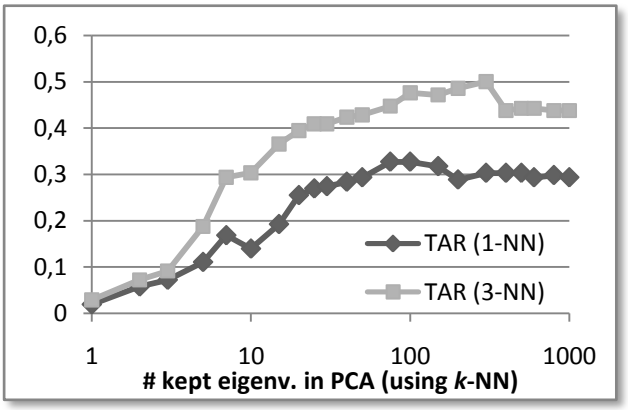


Fig. 36. Evaluation of the number of kept eigenvectors for the profile faces using a high minimum distance

We can see in Fig. 36 that increasing the minimum distance the performance of the system improves, especially when using three neighbours instead of one (the maximum number that we can test was three as there only were four images per model) achieving a higher TAR of 0.5 with 300 eigenvectors. This is a higher value of eigenvectors than the obtained before, where the highest TAR (but it was only 0.26) was achieved with 40 eigenvectors. We should also note that the minimum probability is reduced because some of the probabilities obtained with the 3-NN classifier were about 0.3 (due to the way that our soft solution estimates the probabilities).

Although we can guess that the results obtained will be worse than the obtained with k-NN, now we are going to evaluate the profile faces with the modified Parzen classifier, using the parameters shown in Table 28 (and a minimum distance of 0.5). As done before, we firstly perform a set of simulations to estimate an initial value for the multiplier of the standard deviation $(h = k\sigma)$, which in this case is 35. This is taken as a first estimation for the parameter, which we will use along the simulation made to evaluate the number of kept eigenvectors but it should be optimized in every specific case.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	4	208	XM2VTS profiles	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
30 x 60	PCA	-	No	Mod Par	1	0.1

Table 28. Parameters used for the evaluation of the profile faces using the modified Parzen classifier

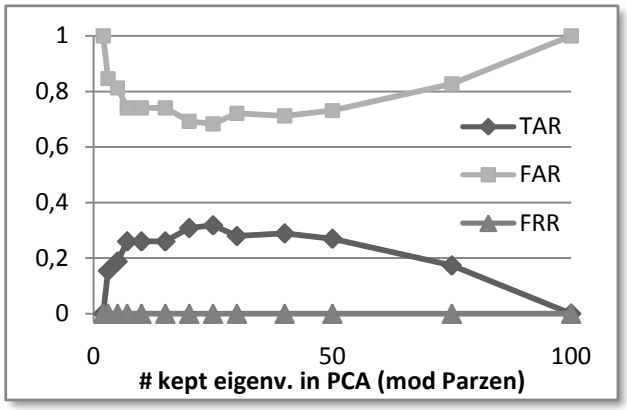


Fig. 37. Evaluation of the profile faces using the modified Parzen classifier

We can see in Fig. 37 that the highest TAR (0.32) is achieved for 25 eigenvectors, a higher value with a lower dimension of the space than using the 1-NN classifier. However, these rates can be improved reducing the minimum probability as we can see in the next graph (note that we have taken the parameters shown in Table 28, the optimal value for the eigenvectors, 25, and the optimal value for the window width, in this case with the multiplier $k = 35$).

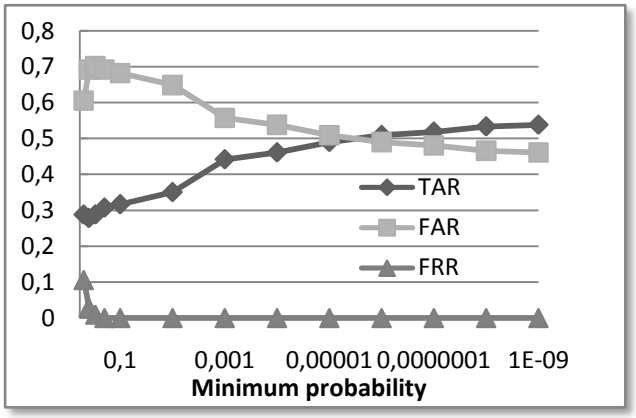


Fig. 38. Evaluation of varying the minimum probability using a modified Parzen classifier with the profile faces

We can see that the TAR increases to a maximum of 0.53. By lowering the probability so much we are trying to accept the verifications results (which can give very low probabilities) and not perform identifications (where, according to the results, some classes seem to overrun the others as most of the samples are identified with a reduced number of models, which is also what we can suspect when we had to increase the minimum distance with the k-NN classifier, as the samples are distant from its respective models).

To sum up, we can see that the recognition of the profile views is more difficult than with the frontal faces, and to achieve maximum rates about 0.5 we have to “force” our tools in some way.

6.2 Test of the whole system

At this point we are going to test the whole system; that is, using the updates and taking into account all the information about the parameters and the different available options that we have found in the previous tests. The reason to test the updates now is that with the databases that we have been using the number of images of the same person is reduced and also very similar among them, so the effect of adding or deleting images from the models (that as we have explained is used to model the variability of the faces) will not be very significant to the final result. For that reason, we expect that if we have bigger sets of faces we will be able to test the suitability of these techniques and also try to detect the weak points of the algorithm when we use "real" data. In any case, we will see the effect of the updates using the available databases and infer some conclusions about its behaviour.

6.2.1 Evaluation of the updates

Firstly we will evaluate the effect of the model updates using the controlled images. For that reason, we have performed a modification of the system where there are no updates, changing the function that adds the vectors to the corresponding model for a void function, with the intention of comparing the results with the usual working mode. Besides, another of the possible ways to see if the updates improve the performance of the system is running the program twice, the first one with the models generated with a determined set of faces, where the updates are performed, and then test again with these updated models. We will make some simulations in an analogue way to the tests performed in section 6.1.5 where the effect of the number of vectors of the initial models was analyzed, so we will start with the parameters shown below and we will vary the number of initial vectors starting with 1 image from session one, then 2 images, 1 from session one and another from session two, up to 4 images from the different sessions. Furthermore, in addition to the rates that we usually have considered for the evaluation of the algorithm, we will also use other parameters to see the effect of the updates like the mean number of vectors in the local models at the end of the simulation, the relative mean number of descriptors (RMND, the rate between the mean number of vectors and the mean number of testing images), the number of directly added images and the number of updates using the cosine aperture method.

# of classes	# initial faces per class (initial models)	# testing images	Database	Labelling errors
52	? (controlled)	1639	BANCA SP controlled	No

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20 x 30	PCA	150	Yes	1-NN	1	0.5

Table 29. Parameters used for the evaluation of the updates with controlled images

# of images per model	TAR	FAR	FRR	RMND	Mean # of vectors (end sim)	# of directly added	# of updates (cos ap)
1 (no update)	0.582602	0.142160	0.275778	3.17	1	-	-
1 (normal mode)	0.849908	0.117755	0.032337	18.24	5.75	53	194
1 (after 1st update)	0.882245	0.093960	0.023795	20.62	6.50	39	0
2 (no update)	0.746797	0.107383	0.145821	6.34	2	-	-
2 (normal mode)	0.879195	0.092739	0.028066	13.11	4.13	46	65
2 (after 1st update)	0.889567	0.085418	0.025015	15.62	4.92	41	0
3 (no update)	0.890177	0.062843	0.046980	9.51	3	-	-
3 (normal mode)	0.922514	0.059182	0.018304	13.30	4.19	30	32
3 (after 1st update)	0.924344	0.058572	0.017084	15.01	4.73	28	0
4 (no update)	0.975595	0.012813	0.011592	12.68	4	-	-
4 (normal mode)	0.976205	0.014643	0.009152	14.46	4.55	15	14
4 (after 1st update)	0.976815	0.014033	0.009152	15.37	4.84	15	0

Table 30. Evaluation of the updates with the controlled images

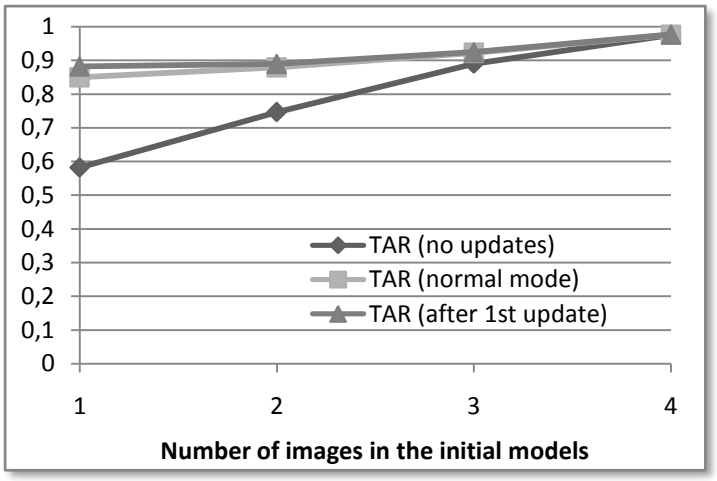


Fig. 39. Evaluation of the updates varying the size of the initial models

One of the first aspects to notice is how the performance clearly improves when we use the updates, especially when we have one or two vectors for the initial models, whereas with three or four images the values are very similar. We can suppose some of the main reasons for this behaviour, because as we are testing the system with images from the four sessions, when we only use one image from session 1 and do not perform any updates, the system cannot capture the global variability of the faces, and for that reason the TAR is lower than the obtained using the normal mode. For that reason, when we use one image of every session for the initial models the performance is quite similar using updates or not, as we are representing the variation of the faces in the models. In addition to this we can see that the performance between the normal model and after the first update improves very slightly in general, with a higher improvement of only a 2.5% when we start with 1 image per model.

Moreover, we can deduce some other issues about the behaviour of the updates. Firstly, that the number of updates reduces when we use more images in the initial models and how, excluding the case of using an image per model, the RMND is similar and the mean number of vectors at the end of the simulation is also similar, with values between four and five vectors. We can also see how the second time that we run the program, there are no updates following the cosine aperture criterion (as theoretically the system has already

added all the possible candidates). With this information we can infer that according to the updates criteria the maximum number of images to add to the models is limited, so the classes do not overrun among them, although the performance varies according to how these vectors represent all the faces. Another aspect to emphasize is that even though we add more vectors to the models, for example, at the end of the usual mode we have about six vectors per model when we start with a single image from session one, the performance is lower than the obtained starting with three images per model, when we finish the simulation with values between four or five vectors per model. These can be due to different reasons, for example we can suppose that the system adds vectors following the cosine aperture criterion but these are not the vectors which provide the best representation of all the faces, because they are added late, and some faces are rejected or misclassified until some representative samples are added to the model, or too soon as some vectors are added to the model instead of some possible future candidates which could provide better results but are rejected to not overrun other models.

To sum up, with these experiments we have been able to evaluate the effect of the model update and see that its effect is significant and very necessary when the initial models do not capture the global variability of the faces. Besides, according to the results we can see that the effect of the model reduction, removing the unnecessary vectors from the models, can also be very important to keep the size of the models controlled and also to try to select only the most representative vectors. For that reason the application of the model reduction should be done periodically during the usual working mode. However, as the sequences were not very long we had designed that part of the system separately, so the model reduction can only be performed offline, not when the system is working. For that reason the evaluation of its effect is left for future tests.

6.2.2 Testing with CHIL images

Now we are going to test our algorithm using images from the CHIL project. For that reason, we will use a set of data from a 2007 evaluation, where there are sequences from different institutions which participate on the CHIL project. The data are structured in different directories and split into training and validation sequences, where there are images corresponding to different cameras. The effect of the face detector is simulated using files where the coordinates of the bounding box (that is, the rectangle that delimits the face) and the coordinates of the eyes have been manually found.

So to obtain the faces that are going to be delivered to our recognition system, firstly we have to design a system that cuts out the faces with their corresponding coordinates. We start with the train or valid files (there are a set of them, which use different time lengths), which provide a list with the starting and the final timestamp, among other parameters, with a line for every individual which is going to be used in the training or the validation. Then we have to search in a sequence index the correspondence between the timestamp and the number of the frame, and read another file where for every camera, there are the coordinates corresponding to the determined face, ordered by the timestamp. Finally we have to read those coordinates and cut out the face from the corresponding image which is identified with frame number, save it to the corresponding directory in an appropriate format (SUN raster .ras and grey-scale in our case) and also determine the view. Thanks to

the eyes coordinates we can estimate the view, frontal if the two eyes are present and profile if only one of them is marked. Besides, as we will only use left profiles in our recognition system, the right profiles are directly mirrored. Finally we should perform these tasks for every timestamp between the initial and final values for the different cameras, and also for the different individuals of the train or valid files. The functions used for these tasks can be found on the L_CLEARID SoftImage library and the B_CLEARID binary, which have been modified for this purpose.

Following this procedure we have obtained different sets of faces, two training sets (A and B) and different validation sets according to the length of the sequences (sequences of 1, 5, 10 or 20 seconds) which we are going to use to test our system. In Fig. 40 we can see some examples of these faces.

Fig. 40. Examples of frontal (top) and profile (down) faces used for the evaluation with CHIL images

Considering the 2612 frontal faces of the sequence of 20 seconds we obtain an average size of 20.53 x 30.47, which corresponds to an aspect ratio of 1.48, and for the 3971 profile faces we obtain an average size of 18.09 x 30.90, which corresponds to an aspect ratio of 1.71. The first aspect to notice is that the frontal faces almost match exactly the faces that we have considered with an intermediate size (20x30), whereas for the profile faces the size and the aspect ratio are slightly different than the ones that we had used. These can be due to the fact that the profile faces are not exactly profiles, like a mug shot as we have in the XM2VTS database, and we can find faces from nearly frontal to almost backwards, as we can see in Fig. 41.

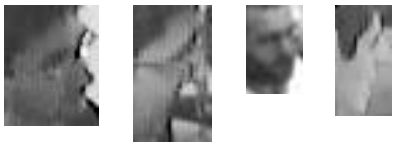


Fig. 41. Examples of faces considered as profile which not correspond to mug shots

However, before we start with the simulations we should recall what we have considered about the effect of masking the frontal faces in section 6.1.3. We had seen that using perfectly frontal faces, well aligned and cut, masking the faces can improve the recognition rates (for example a 5% in the experiment with the degraded images) but for images in non-optimal conditions we could suppose that masking the faces could even be detrimental, as we were removing potentially relevant information from our images. For that reason, we firstly have performed a simulation using the basic parameters shown below in Table 31, considering the masking or not. These parameters are taken as a compromise among the optimal values obtained previously with the controlled and the degraded images, as we are

trying to apply our tools to a general problem where the quality of the images is variable.

View	# of classes	# initial faces per class (initial models)	# testing images	Database
Frontal	28	73.8 (TRAIN_A)	680	VALID_5_1 (frontal)
Profile	28	113.6 (TRAIN_A)	1006	VALID_5_1 (profile)

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20x30 (fr)	PCA	150	?	3-NN	1	0.5
20x40 (pr)	PCA	100	No	3-NN	1	0.5

Table 31. Starting parameters for the testing with CHIL images

The results, summarized in Table 32, show how the effect of masking the frontal faces is still beneficial, as for example the TAR improves from 0.62 to 0.69. One of the possible reasons for this behaviour is that although with the oval mask we are potentially removing some significant part of the faces, what we are removing for sure is a big part of the background, that with its inherent variability could affect more to the performance of the system.

	Mask	No mask
TAR	0.695588	0.626471
FAR	0.083824	0.017647
FRR	0.220588	0.355882

Table 32. Comparison between the effect of masking the CHIL frontal faces or not

Now, using the same parameters shown before, we will test the system varying the number of images per group. Besides we will evaluate the performance using the combiners that have given the best results in the previous test, the mean and the maximum.

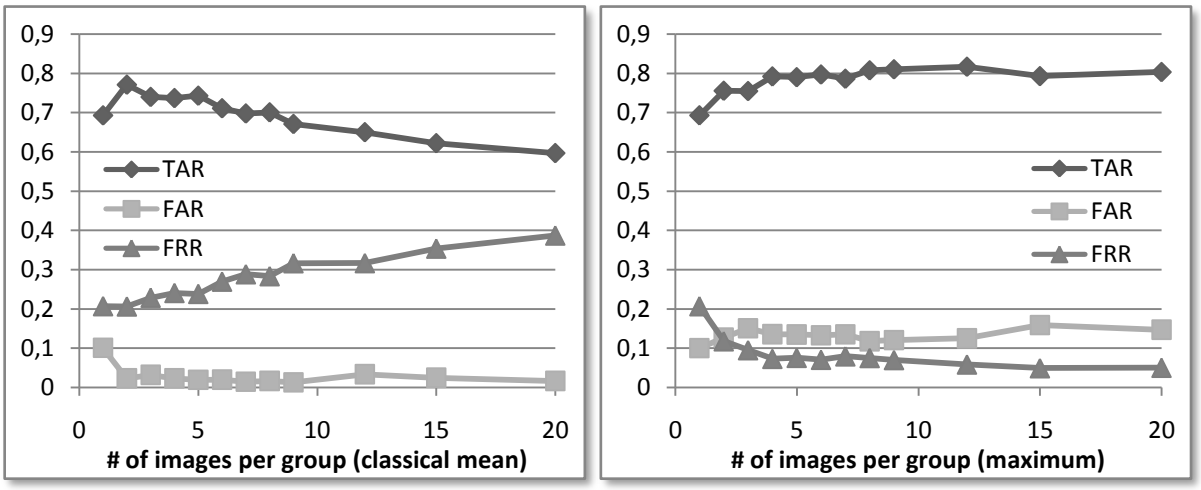


Fig. 42. Evaluation of the number of images per group with the CHIL images using a 3-NN classifier

In Fig. 42 we can see that the results obtained with the maximum combiner are better than the obtained with the mean. This is due to the fact that when there are images recognized

with a low probability, the resulting mean of the group is a low value. So, when the size of the groups increases, it is likelier that the images with low probabilities reduce the total mean value, so the recognition rates decrease. This effect can be avoided lowering the minimum probability to consider that the face corresponds to a class (we have fixed it to 0.5). In any case, we can pass from a TAR of 0.69 to 0.77 using two images. On the other hand, if we use the maximum combiner we consider a more optimistic scenario, so the faces recognized with lower probabilities do not contribute to the final decision, and we can see that the TAR can increase from 0.69 when we use one image to 0.82 when we use twelve.

Now we are going to perform an analogue experiment but using the modified Parzen classifier, using the parameters shown below. Again, we choose the parameters basing us on the previous simulations, and the optimal window width has been estimated first (in this case $h = 22\sigma$).

View	# of classes	# initial faces per class (initial models)	# testing images	Database
Frontal	28	73.8 (TRAIN_A)	680	VALID_5_1 (frontal)
Profile	28	113.6 (TRAIN_A)	1006	VALID_5_1 (profile)

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20x30 (fr)	PCA	50	Yes	Mod Par	?	0.5
20x40 (pr)	PCA	50	No	Mod Par	?	0.5

Table 33. Parameters used for the evaluation of the CHIL images with the modified Parzen classifier

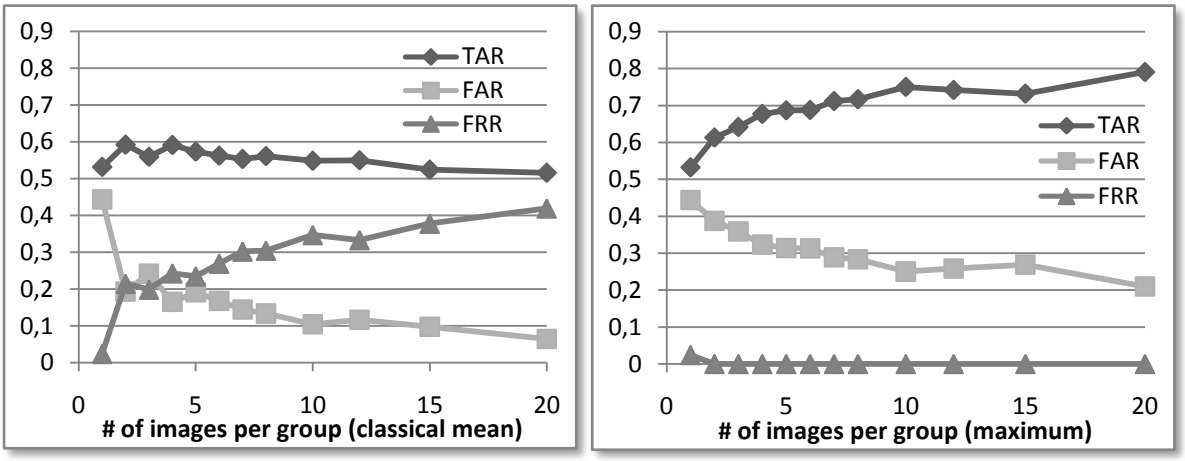


Fig. 43. Evaluation of the number of images per group with the CHIL images using the modified Parzen classifier

In Fig. 43 we can see how the tendency is similar to the observed using the 3-NN classifier, but the TAR is slightly worse (about a 10%) than in the analogue simulations made before.

Now we will see a final simulation, where we will evaluate the performance of the system with the whole sequence and we will evaluate other results, like the number of identifications and verifications, the number of updates, or the time taken for the evaluation. Besides we will compare the results obtained with the two training sequences, considering than the second one is longer, using the following parameters.

View	# of classes	# initial faces per class (initial models)	# testing images	Database
Frontal	28	73.8(A), 149.5(B)	2612	VALID_20 (frontal)
Profile	28	113.6(A), 223.6(B)	3791	VALID_20 (profile)

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20x30 (fr)	PCA	150	Yes	3-NN	?	0.5
20x40 (pr)	PCA	100	No	3-NN	?	0.5

Table 34. Parameters used for the evaluation of the whole CHIL sequence

Firstly we will evaluate the computational load taken for the different processes. The simulations are performed in a server with an Intel Xeon CPU at 3.00GHz and 10GB of RAM memory, accessed remotely. The duration of the processes is estimated using the Linux *time* command which provides three results, the elapsed real time between invocation and termination (where the time spent with other processes or disk delays can affect the estimation), the user CPU time (the time spent in user mode), and the system CPU time (the time that the system takes to attend the process; that is, in kernel mode). In any case, with these time estimations we look for an approximation to have an idea of the magnitude orders, so we will mainly consider the real time.

For example, to generate the initial models with the TRAIN_A frontal faces with 150 eigenvectors we need approximately 17 seconds, whereas the theoretical optimal time would be (adding the user and sys times) 7.4 seconds. With the TRAIN_B frontal faces, where there are approximately twice as many images, the real time is 50.4 seconds whereas the theoretical would be 23.7. For the simulation using the TRAIN_A faces for the generation of the initial models and the testing with the VALID_20 faces, we need 1 minute and 16 seconds to process the 6403 images. On the other hand, if we perform the same experiment using the modified Parzen classifier it takes 5 minutes and 47 seconds, although we are using 50 eigenvectors in every space instead of 150 for frontals and 100 for the profiles like when we used the 3-NN classifier.

The results of the simulations are summarized in Table 35, where four experiments have been performed, using the two training sequences and two different sizes, 1 or 10, for the groups of images (the combination of the results is made using the mean that as we have seen provides the best results together with the maximum, but it is not as optimistic which seems suitable for these tests with “real” data).

Training faces	TRAIN_A	TRAIN_A	TRAIN_B	TRAIN_B
Number of images per group	1	10	1	10
TAR	0.746837	0.787097	0.792129	0.845161
FAR	0.174293	0.200000	0.144464	0.150000
FRR	0.078869	0.012903	0.063408	0.004839
RMND	100.7	109.7	179.1	186.1
Mean # of frontal faces at the end of the simulation	92.3	99.5	165.1	172.1
Mean # of profile faces at the end of the simulation	137.9	151.3	244.5	253.6
# of directly added faces	1116	1240	925	930
# of updates (using the cosine aperture)	85	537	95	540
Verifications	4782	4880	5072	5240
Identifications	505	80	406	30
Simulation time	1’ 16’’	1’ 16’’	1’ 31’’	1’ 32’’

Table 35. Evaluation of the whole CHIL sequence

We can see from the results that the TAR clearly increases when we use groups of 10 images, whereas the FAR increases very slightly and the FRR is reduced. Another aspect to consider is that the number of updates is clearly higher when we use 10 faces per group, which is also reflected in the mean number of vectors at the end of the simulation. One of the possible reasons to explain this is that as the decision is taken for all the images of the group, the number of update candidates is higher than when they are processed individually, and more of them are finally added to the models. Another issue is that the number of identifications is reduced when we use the groups of 10 images, and we can guess that the explanation is similar than for the updates, taking the decision for a group provides higher probabilities than individually, so the groups are mostly verified. Finally we should note that the simulation time is higher when we use the TRAIN_B sequence, as initially the models have a higher number of vectors.

6.2.3 Testing with other sequences

Now we are going to use a sequence, which although it is not very long, shows a possible scenario for a real application. Here we start from a video sequence from the HESPERIA project (Homeland sEcurity: tecnologíaS Para la sEguridad integRal en espacIos públicos e infrAestructuras, project CENIT-2005), which shows a set of people entering to a demonstration room and has been recorded from a fixed camera in front of the door. Then the sequence has been divided into a set of frames where the different faces have been manually cut and tagged. In Fig. 44 we can see some examples of these faces.

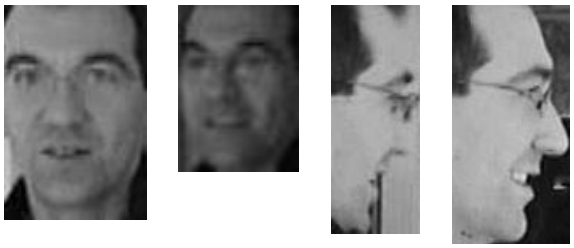


Fig. 44. Examples of frontal and profile faces used for the evaluation with other sequences (Hesperia domo)

The simulation is made using the parameters shown in Table 36, where we can see that there is a reduced number of images, for both the training and the testing. The mean is taken as the combiner, as we have seen that it provides good results and it is not as optimistic as the maximum combiner.

View	# of classes	# initial faces per class (initial models)	# testing images	Database
Frontal	14	3.2	179	Hesperia domo
Profile	11	3	49	Hesperia domo

Face Size	Type of projection	# kept eigenvec.	Mask in PCA	Type of classifier	Size group	Min prob.
20x30 (fr)	PCA	150	Yes	3-NN	?	0.5
20x40 (pr)	PCA	100	No	3-NN	?	0.5

Table 36. Parameters used for the evaluation with the Hesperia images

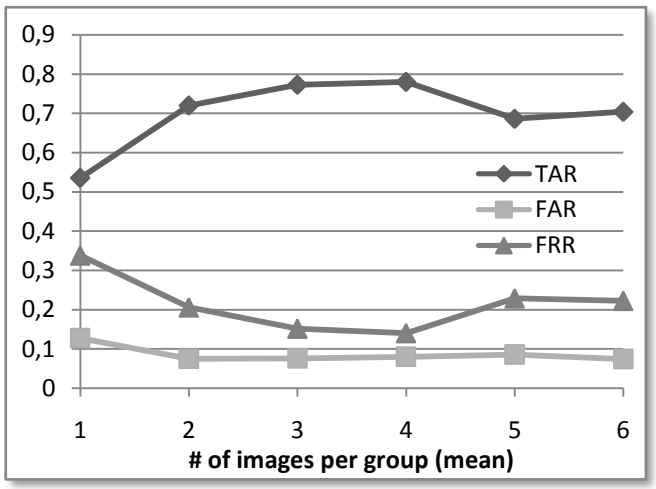


Fig. 45. Evaluation of the number of images per group with the Hesperia images

We can see in Fig. 45 the results obtained with this sequence, starting from a low TAR value of 0.53 to a maximum of 0.78 with groups of 4 faces (we have not considered bigger values for the number of images per group as there are not as many faces from the same individuals to fill the groups).

7 Conclusions and future work

Throughout this work we have been able to get an overview of the face recognition problem and its application to a controlled environment has been proposed. We have used a top-down approach for the evaluation of the different aspects of the problem, and this approach has been taken both for the study of the theoretical issues and for the design of the different parts of the system. For that reason we have started considering face recognition as a particular case of pattern recognition, and then we have seen the different parts of a generic pattern recognition problem and how they could be applied to our field.

About the different stages of a pattern recognition system we have mainly focused on three of them, feature selection, classification and model definition and update. In any case, in this section we are not going to detail the results obtained with the different tools as we have already done an exhaustive evaluation in the previous chapter, where we also have explained and justified the results whenever it was possible. Therefore we will give some general impressions about the behaviour of the system.

Concerning the feature selection we have evaluated the different parameters that influence the PCA representation and we have compared it to LDA. We have seen that although the results are quite good with PCA, with LDA are slightly better (at least in the performed test, although we should test it more extensively). For that reason, the suitability of LDA is left as a future aspect to study, considering its problems when dynamic models are used, as in our case (see [10] for a possible solution for this issue).

About the classifiers, we have seen that k-NN is simpler and provides better results than the Parzen classifier, clearly limited if the number of images to generate the initial models is reduced and with some limitations due to the choice of the optimal width. With the modified Parzen we have obtained results close (but worse) to the ones obtained with k-NN, but with a higher cost in time and resources. In any case, we should not discard this technique as it has some specific potentially usable characteristics. For example, with Parzen we can obtain estimations of the probability that one sample belongs to a determined class without considering the rest, like in a verification, whereas with the k-NN classifier we consider all the samples at the same time when we search for the nearest neighbours samples, which corresponds more to the concept of identification. In our case we have adapted the two techniques to verify or identify a face, but this difference in their conception could be used in other situations.

Finally, another important issue is the model update, which we have justified theoretically first and later we have evaluated its effect, where we have noticed the improvement achieved adding new faces to the models.

Furthermore, the effect of using sets of images has been tested and we have been able to see that the recognition rates improve in general, especially in optimistic scenarios where there are not long burst of unidentifiable faces which can burden the estimated probability of the group. Besides, we should notice that the combiner chosen is not a very deciding factor, so the mean is commonly used.

Anyway, there are some general aspects to improve, even for a posterior use of the

proposed algorithm or for the generation of other recognition systems based on this one. For example, we have become aware of the possible dependence of different parameters while we were testing some of them. Although we have started with the premise of the independence of the parameters, we have seen that some of them could be linked. Moreover, we have seen how some of the optimal parameters for the controlled and the degraded faces were different, which can lead us to think that there are no optimal parameters for all the situations, although we can find some compromise solutions. For these reasons, we could think of a possible improvement implementing some kind of training stage where the parameters should be optimized at the same time and according to the testing images.

Besides, another aspect to reconsider in our system is the estimation of the posterior probabilities. We have seen that the number of neighbours is directly related with the probabilities obtained with our soft k-NN classifier, and for the Parzen classifier the value of the window width h can also affect the estimation of the probabilities. Besides, for both the classical and the modified Parzen we have obtained the best results considering different window widths, taken as a multiple of the standard deviation for every class, but this cannot be taken as the optimal method. For example, varying slightly the values of the window width we can switch to estimations of the posterior probabilities from zero to nearly one, so the election of the optimal value should be done very accurately. Therefore, we should reconsider these two aspects, the estimation of the window width, which we can link to the need of an exhaustive training where the parameters are optimized at the same time, and the estimation of the posterior probabilities with the k-NN classifier using other approaches, even considering some type of hybrid method.

About the model update we have realized that adding vectors to the models the performance of the system is improved, but we have not considered vector deletion. Although we have considered it theoretically and the functions have been implemented, vector deletion was thought as an offline operation to be performed after the simulations, but we have seen in the discussion about the updates that it might be interesting to perform the vector deletion periodically within the sessions. Besides we can think of other criteria for the updates, not only the cosine aperture, for example using a temporal criterion or extending the entropy approach used in the reduced Parzen for the k-NN classifier. Thus, these aspects of vector removal are left as future modifications to study, especially if long sequences are available, which was not our case.

Finally we should consider some future lines of work, although some matters have already been introduced as improvements of the tools used in our system. Even though we have taken into account the basic techniques for feature extraction and classification, there are a lot of available options (we can find some of them in the literature surveys like [8]). For example, concerning feature extraction we have mainly focused on PCA but there are other types of techniques, like DCT (discrete cosine transform) or Gabor wavelets, whereas for the classifiers we could use tools like SVM (supporting vector machine) or neural networks (like the multilayer perceptron). Even with the applied techniques we could use other modifications, like using other distances and not only the Euclidean for the nearest neighbour classifier. Besides we can find other modifications for more specific problems, like the pose variation or the illumination changes. We have considered two subspaces, for frontal and profile faces, but another possible option was to use a 45° subspace (we have not considered it for the lack of images) or the use of morphing techniques to generate faces

in the desired view from the existing ones. Concerning the illumination we could propose systems where pre-processing the images we can estimate the direction of the light source and normalize the face in consequence (this can be suitable for faces with strong illumination changes, for example lighted from only one side). Anyway for our type of environments it is not a very important problem but in outdoor systems this might be crucial.

In addition to this, we could take advantage of the tools used for the fusion of the results with groups of images and use them for their original purpose, the fusion of different classifiers. We could adapt our DP (decision profile) and DT (decision template) matrices to perform a decision using more than one classifier for every image. We could use different classifiers or the same classifier with different parameters, adapted to the possible different characteristics of the faces. Finally some other classifier is chosen to make the final decision with the available data, for example selecting the best classifier or weighting the results to combine them. This was the original idea for the use of DT matrices [5], with combiners like $DT(E)$ and $DT(S)$. With this idea we were in some way trying to apply a concept similar to boosting; that is, combining weak classifiers to obtain a stronger classifier (although they will not be so weak). Besides we could extrapolate the concept to groups of images, using 3-dimensional matrices, even though the final decision would be more complex to take.

Finally, thinking in a possible implementation in a real scenario we should evaluate the performance of our solution with a real face detector and a real tracking system. Besides we should study and develop the potential interrelationships among these different parts of the system.

To sum up we can affirm that we have accomplished the objectives proposed in the introduction, starting from a general view of face recognition as a pattern recognition problem, following with the study of its different parts, the implementation of the selected tools using a modular architecture, performing an extensive set of simulations and finishing by testing the system with real data from a possible environment where this algorithm could be applied.

Bibliography

[1]. Face Recognition Vendor Test (FRVT) Webpage. [Online] http://www.frvt.org/.

[2]. *An introduction to biometric recognition.* **Jain, Anil K., Russ, Arun and Prabhakar, Salil.** 1, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, pp. 4-20.

[3]. CHIL project Webpage. [Online] http://chil.server.de/servlet/is/101.

[4]. **Duda, Richard O., Hart, Peter E. and Stork, David G.** *Pattern Classification, Second Edition.* N.Y. : Wiley-Interscience, 2000.

[5]. **Kuncheva, Ludmila I.** *Combining Pattern Classifiers: Methods and Algorithms.* N.J. : Wiley-Interscience, 2004.

[6]. *Statistical pattern recognition: a review.* **Jain, Anil K., Duin, Robert. P. W. and Mao, Jianchang.** 1, 2000, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, pp. 4-37.

[7]. **Bellman, R.E.** *Adaptive Control Processes.* Princeton, NJ : Princeton University Press, 1961.

[8]. *Face recognition: a literature survey.* **Zhao, W., et al.** 4, December 2003, ACM Computing Surveys , Vol. 35, pp. 399-458.

[9]. *Eigenfaces for recognition.* **Turk, M. and Pentland, A.** 1, 1991, Journal of Cognitive Neuroscience, Vol. 3, pp. 71-86.

[10]. *Discriminant analysis of principal components for face recognition.* **Zhao, W., Chellappa, R. and Krishnaswamy, A.** 1998, Proceedings, International Conference on Automatic Face and Gesture Recognition, pp. 336-341.

[11]. *Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection.* **Belhumeur, P.N., Hespanha, J.P. and Kriegman, D.J.** 7, 1997, IEEE Tran. Pattern Analysis and Machine Intelligence, Vol. 19, pp. 711-720.

[12]. *Beyond eigenfaces: probabilistic matching for face recognition.* **Moghaddam, B. and Wahid, W.** 1998, IEEE International Conference on Automatic Face and Gesture Recognition, pp. 30-35.

[13]. *Probabilistic visual learning for object representation.* **Moghaddam, B. and Pentland, A.** 7, 1997, IEEE Trans. Patt. Anal. Mach. Intell., Vol. 19, pp. 696-710.

[14]. *Face recognition by elastic bunch graph matching.* **Wiskott, L., et al.** 7, 1997, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, pp. 775-779.

[15]. *View-based and modular eigenspaces for face recognition.* **Pentland, A., Moghaddam, B. and Starner, T.** 1994. Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on. pp. 84-91.

[16]. *Probabilistic recognition of human faces from video.* **Zhou, S., Krueger, V. and Chellappa, R.** 1-2, July-August 2003, Computer Vision and Image Understanding, Vol. 91, pp. 214-245.

[17]. *A system identification approach for video-based face recognition.* **Aggarwal, G., Chowdhury, A.K.R. and Chellappa, R.** August 2004, Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, Vol. 4, pp. 175-178.

[18]. *From stills to video: face recognition using a probabilistic approach.* **Zhang, Y. and Martinez, A. M.** June 2004, Computer Vision and Pattern Recognition Workshop, 2004 Conference on, pp. 78-93.

[19]. *Video-based face recognition using adaptive hidden Markov models.* **Liu, Xiaoming and Chen, Tsuhan.** June 2003, Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, Vol. 1, pp. 340-345.

[20]. *XM2VTSDB: The extended M2VTS database.* **Messer, K., et al.** 1999, Proceedings,

international conference on audio- and video-based person authentication, pp. 72-77.
[21]. *The BANCA database and evaluation protocol.* **Bailly-Bailliére, E., et al.** 2003, Proceedings of the international conference on audio- and video-based biometric person authentication, pp. 625-638.
[22]. *The statistical utilization of multiple measurements.* **Fisher, R. A.** 1938, Annals of Eugenics, Vol. 8, pp. 376-386.
[23]. *PCA versus LDA.* **Martinez, A.M. and Kak, A.C.** 2, Feb 2001, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 23, pp. 228-233.
[24]. **Martínez Hernàndez, Claudi.** *Face recognition in the context of smart rooms. Application to the CHIL project.* UPC : PFC ETSETB, 2005.
[25]. **Liu, Xiaoming, Chen, Tsuhan and Thornton, Susan M.** *Eigenspace updating for non-stationary process and its application to face recognition.* Advanced Multimedia Processing Lab, Electrical and Computer Engineering Carnegie Mellon University. 2002.
[26]. **Fukunaga, Keinosuke.** *Introduction to statistical pattern recognition.* Second edition. San Diego, CA : Academic Press, 1990.
[27]. **Turlach, Berwin A.** *Bandwidth selection in kernel estimation: a review.* C.O.R.E. and Institut de Statistique Université Catholique de Louvain. 1993.
[28]. *A Brief Survey of Bandwidth Selection for Density Estimation.* **Jones, M. C., Marron, J. S. and Sheather, S. J.** 433, Mar. 1996, Journal of the American Statistical Association, Vol. 91, pp. 401-407.
[29]. *Small sample size effects in statistical pattern recognition: recommendations for practitioners.* **Raudys, S.J. and Jain, A.K.** 3, Mar 1991, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 13, pp. 252-264.
[30]. *The reduced Parzen classifier.* **Fukunaga, K., Hayes R. R.** 4, 1989, IEEE Trans. Pattern Anal. Mach. Intell., Vol. 11, pp. 423-425.
[31]. *Evaluation of a modified Parzen classifier in high dimensional spaces.* **Muto, Y., Nagase, H. and Hamamoto, Y.** 2000, 15th International Conference on Pattern Recognition. Proceedings, Vol. 2, pp. 67-70.
[32]. **Press, W. H., et al.** *Numerical recipes in C: the art of scientific computing.* Second edition. Cambridge : Cambridge University Press, 1992.

Appendix I: Implementation remarks

As we have commented previously, one of the objectives of the project was the development of the software as a modular architecture using the UPC's SoftImage platform. This is a software package which contains a set of libraries with different image processing tools, which are continuously updated with new contributions from people of the UPC Image Processing Group.

SoftImage was originally developed in C language, but at this time it is progressively being ported to C++. However, this project began before this initiative started to work, so it is completely developed in C and also uses the SoftImage libraries in C. In any case, C++ is based on and is largely compatible with C, so the code should be easily compiled and used in C++, although it will not take advantage of the object-oriented vision and its associated tools.

Besides, the development of the system was made in a Linux platform, specifically the Fedora distribution, and compiled with GCC. The code was debugged using the GNU Debugger (GDD), and tools like Valgrind were also used for finding memory errors and leaks.

With the intention of developing a modular code, the main algorithms of our solution are developed in a new library, which will use at its time some of the functions and data structures from the existing libraries. The code has been divided into different functions, trying to be as general as possible. With this, we want to get a solution where some of the parts of the algorithm can be changed, for example using a determined classifier or another, but keeping the general structure of the algorithm.

Basically the system is divided into two types of modules, the libraries, where most of the algorithms and functions are placed, and the binaries, which contain the main functions. Our library is called L_CLASSIFIERS, and it uses functions from L_FACE and other SoftImage libraries. About the binaries, we will use, among others, B_LM_GEN_ALL which generates the initial models, B_DB_PCA which generates the PCA projection matrix, and the three created to test our system, B_REC_LDA, B_REC_KNN and B_REC_PAR, corresponding to the recognition using the LDA, k-NN and Parzen techniques. These modules follow a similar structure and use three main inputs: the raw face images, an image specification file and a settings or configuration file. As the output of the system we will print the information corresponding to every face and its estimated *id* and different results and statistics, as we will see next. The program is called invoking to the settings file, for example, B_REC_KNN settings.conf. This settings file includes different paths, for example, for the PCA projection matrices, the face images path, the image specification file path, and different parameters like probability thresholds, number of images per group, the type of combiner, etc. The image specification file contains the image definitions as if they were delivered by the tracking system, so in every line we have the image name, the preassigned *id*, the view or pose estimation and the true *id* of the image (for example, 3001_m_g1_s01_3001_es_2_y.ras,individual_1,frontal,individual_1). This last *id* is not used by the system but it is needed to generate the statistics while we were testing our solution, for example to simulate labelling errors. In any case, for further information please refer to the code in the attached CD to this document and the SoftImage readme file.

Another important aspect refers to the implementation of some of the proposed solutions. For example, about the PCA algorithm, as we are dealing with high dimension vectors, the way that eigenvalues and eigenvectors are obtained can consume many resources and even present singularities. For that reason we will use a generic solution based on the code of [32], which uses matrix tridiagonalization to simplify the estimation of the eigenvalues and eigenvectors, and was previously available in a library of the SoftImage platform. Another example is the algorithm used for matrix inversion and the calculation of the determinant, which is based on the LU decomposition (L for lower triangular, and U for upper triangular) and backsubstitution routines, also based on [32].

In addition to this, we should notice that our system works with images in the Sun raster format (extension .ras) with 8 bits per pixel because we use grey-scale faces. This format was chosen as it is the native bitmap format of the UNIX platforms, so it is also widely used in Linux environments, as in our case.