Machine Vision and Applications (2005) 16: 128–138
Digital Object Identifier (DOI) 10.1007/s00138-004-0144-7

Machine Vision and Applications

The CSU Face Identification Evaluation System

Its purpose, features, and structure

J. Ross Beveridge, David Bolme, Bruce A. Draper, Marcio Teixeira

Computer Science Department, Colorado State University

Published online: 20 August 2004 –

Abstract. The CSU Face Identification Evaluation System includes standardized image preprocessing software, four distinct face recognition algorithms, analysis tools to study algorithm performance, and Unix shell scripts to run standard experiments. All code is written in ANSII C. The four algorithms provided are principle components analysis (PCA), a.k.a eigenfaces, a combined principle components analysis and linear discriminant analysis algorithm (PCA+LDA), an intrapersonal/extrapersonal image difference classifier (IIDC), and an elastic bunch graph matching (EBGM) algorithm. The PCA+LDA, IIDC, and EBGM algorithms are based upon algorithms used in the FERET study contributed by the University of Maryland, MIT, and USC, respectively. One analysis tool generates cumulative match curves; the other generates a sample probability distribution for recognition rate at recognition rank 1, 2, etc., using Monte Carlo sampling to generate probe and gallery choices. The sample probability distributions at each rank allow standard error bars to be added to cumulative match curves. The tool also generates sample probability distributions for the paired difference of recognition rates for two algorithms. Whether one algorithm consistently outperforms another is easily tested using this distribution. The CSU Face Identification Evaluation System is available through our Web site and we hope it will be used by others to rigorously compare novel face identification algorithms to standard algorithms using a common implementation and known comparison techniques.

Keywords: Face recognition – Evaluation – Statistical tools

1 Introduction

The CSU Face Identification Evaluation System was created to evaluate how well face identification systems perform. It is not meant to be an off-the-shelf face identification system. In addition to algorithms for face identification, the system includes software to support statistical analysis techniques that aid in evaluating the performance of face identification systems. The current system is designed with identification rather than verification in mind. The identification problem is: given a novel face, provide the identity of the person in the novel image based upon a gallery of known people/images.

As in the original FERET tests [15], the CSU Face Identification Evaluation System presumes face recognition algorithms first compute similarity measures between images and then determine which stored images are most similar to novel images. When this is true, the complete behavior of a face identification system can be captured by the similarity matrix. The CSU Face Identification and Evaluation System creates similarity matrices using the four algorithms described below. Analysis tools are provided that operate on these matrices. These analysis tools currently support cumulative match curve generation and permutation of probe and gallery image testing. A researcher's new algorithm may be compared to standard algorithms using these tools by providing a comparable similarity matrix.

The four algorithms included in the CSU Face Identification Evaluation System are principle components analysis (PCA), a.k.a eigenfaces [18], a combined principle components analysis and linear discriminant analysis algorithm (PCA+LDA) [20], an intrapersonal/extrapersonal image difference classifier (IIDC) [12], and an elastic bunch graph matching (EBGM) algorithm [14]. The PCA+LDA, IIDC, and EBGM algorithms are based upon algorithms used in the FERET study contributed by the University of Maryland, MIT, and USC, respectively.

The first version of the CSU Face Identification Evaluation System was released in March 2001. Starting in December 2001, releases have been identified by version numbers. At the time of this writing, the most recent release was version 5.0, released on 1 May 2003. Version 4.0 was available from 31 October 2002 through 30 April 2003 and was downloaded nearly 2,000 times over that period. As of 1 March 2004, version 5.0 was downloaded over 3,000 times. There is a growing User's Guide [4] included with the CSU Face Identification Evaluation System, and portions of this document are drawn from that User's Guide, although the User's Guide is considerably longer and contains more operational details.

Correspondence to: J.R. Beveridge (e-mail: ross@cs.colostate.edu)

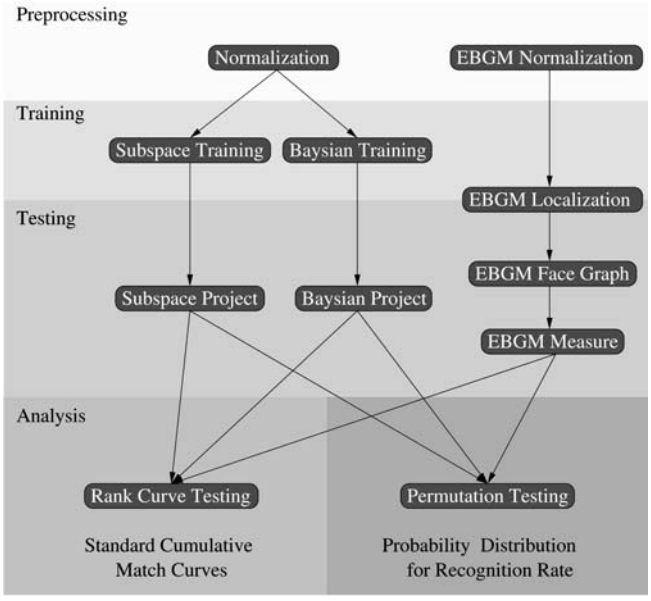


Fig. 1. Overview of execution flow for the CSU Face Identification Evaluation System

For simplicity's sake, the CSU Face Identification Evaluation System will henceforth be called the System. The System, along with additional information about the CSU evaluation work, is available at our Web site [2].

2 System overview

There are four phases of operation for the System: image data preprocessing, algorithm training, algorithm testing, and analysis of results (Fig.1). Preprocessing reduces unwanted image variation by aligning face images, equalizing pixel values, and normalizing contrast and brightness.

Following preprocessing, there is a training phase. Training for the PCA, PCA+LDA, and IIDC algorithms creates subspaces into which test images are subsequently projected and matched. The training phase for the EBGM algorithm is different. There is no global subspace created by the EBGM algorithm and therefore no need to generate one from training imagery. However, the algorithm does require landmark features to be hand selected from a set of training, or model, images.

Testing for all four algorithms creates square matrices that record the distance between all pairs of test images. Operationally, whether to store distances or similarities is a somewhat arbitrary choice, but consistency is essential. The System stores distances, as discussed in Sect. 2.2.

The fourth and final phase performs analyses on the distance matrices. This includes computing recognition rates (`csuAnalyzeRankCurve`), conducting virtual experiments (`csuAnalyzePermute`), or performing other statistical analysis on the data. See Sects. 5 and 6 for examples.

2.1 Imagery and image lists

The CSU Face Identification Evaluation System was developed using frontal facial images from the FERET data set. As described further in Sect. 3, the `csuPreprocessNormalize` utility resamples imagery to align the face to a standard size and place. It then normalizes pixel values. Preprocessing generates image files that contain pixel values stored in a binary floating point format.

Lists of images are critical to keep track of training sets and test sets. The FERET evaluation used simple ASCII image list files with one image name per line. The weakness of this format is that some algorithms need to know which images are of the same person, so the System uses ASCII files in which each line contains the names of all the images for a particular person. This convention generalizes to other data sets without hard coding subject naming conventions in the algorithms themselves. Therefore, image lists have the form:

```
SubjectA_Image1.sfi
SubjectB_Image1.sfi
SubjectC_Image1.sfi
SubjectD_Image1.sfi

SubjectA_Image2.sfi
SubjectB_Image2.sfi
SubjectC_Image2.sfi
SubjectD_Image2.sfi

SubjectA_Image3.sfi
SubjectB_Image3.sfi
SubjectC_Image3.sfi
SubjectD_Image3.sfi
```

The executable `csuReplicates` will produce a file of this form given a list of FERET images. We refer to this as a subject replicates table, and the file is denoted using a file extension *.srt. Some of the tools are also capable of taking a flat list of image names (*.list); however, it is recommended that only ".srt" lists be used. The image lists needed to replicate the original FERET tests as well as our own experiments are included with the System and are documented in the User's Guide [4].

2.2 Distance files

Each algorithm produces a distance matrix for the images in the testing list, and this matrix is divided into distance files. One file is produced for every image in the list. Each line in this file contains the name of another image and the distance to that image. The file has the same name as the probe image for which distance to gallery images are being recorded, and all files are placed in a common directory.

The System assumes smaller distances imply images are more similar and are hence a better match. Unfortunately, some algorithms produce similarity scores, where higher values imply a better match. In this case, the similarity values are negated to produce a "distancelike" value. Examples include correlation in the PCA algorithm and the Bayesian and maximum likelihood measures in the IIDC algorithm. More will be said about distance measures in Sect. 4.2.

2.3 System documentation

With each release of the CSU Face Identification Evaluation System, the User's Guide grows [4] and, we trust, becomes

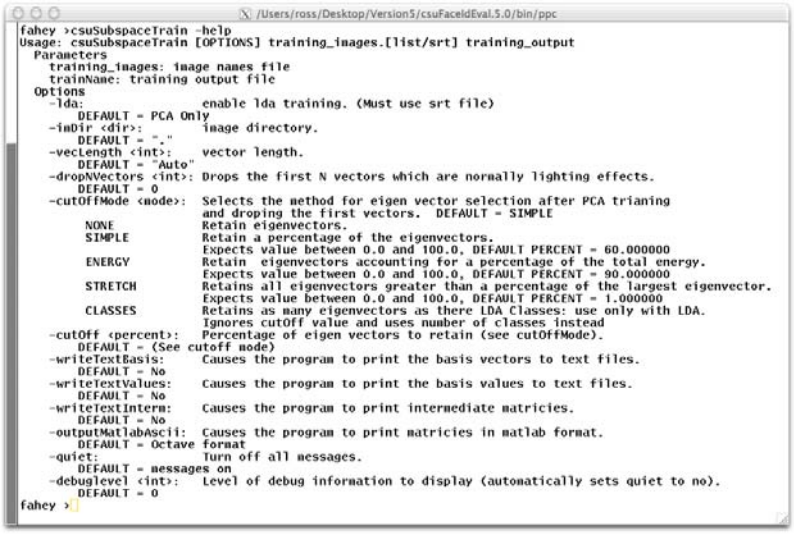


Fig. 2. Example of command line help for `csuSubspaceTrain` executable

more useful. In addition to information in the User's Guide, it is helpful to be familiar with the types of experiments that we've conducted using the system [5,1,10,9]. For each of our tools, the most current documentation and specific instructions as to options are available through the special command line argument "-help". Figure 2 shows an example.

3 Preprocessing

The executable `csuPreprocessNormalize` preprocesses images in a five-step process that tranforms a PGM image to a normalized image. An example of a normalized image and a summary of the five steps are shown in Fig. 3. Step 2 aligns the face images such that the centers of the eyes are always at the same location in the processed image. This geometric alignment is essential for all of the face identification algorithms included in the System. To accomplish this alignment, `csuPreprocessNormalize` needs an ASCII file containing the exact coordinates of the eyes in each of the PGM images being processed. This file is called `coords.3368` for the FERET data and is kept in the `image lists` directory of the System. A user wishing to process imagery form another data set must provide an equivalent file.

The `csuPreprocessNormalize` program accomplishes many of the same tasks performed by the `facetonorm` program distributed with the FERET data by NIST. However, the two programs are not functionally equivalent. For example, histogram equalization in `csuPreprocessNormalize` is performed only for pixels in the unmasked portion of the face. Some aspects of `facetonorm` are problematic, including a tendency to core dump for some combinations of arguments. We prefer to use `csuPreprocessNormalize`.

4 Four face recognition algorithms

The four face recognition algorithms included with the System were chosen because of their importance in the FERET evaluations [15]. Specifically, the PCA algorithm was included in FERET as a baseline algorithm, and it remains useful as such today. The other three algorithms were selected because they performed well in the FERET tests.

For each algorithm a brief overview is provided followed by a summary of the key execution steps, for example the traning and testing steps for the PCA algorithm. Choice of distance measure is important [13,5], and Sect. 4.2 reviews standard and not-so-standard distance measures available for the PCA and PCA+LDA algorithms. Particular attention is drawn to the whitened distances, including the whitened cosine measure. This measure is easily understood geometrically and performs well in practice.

4.1 Principle components analysis (csuSubspace/PCA)

The first algorithm uses principle components analysis (PCA) [18,13] to characterize the space of possible faces from a training sample. As is common now with many subspace methods, images are unrolled into feature vectors: each pixel in the original image is an element in the feature vector. Since all frontal face images are in some broad sense similar, these feature vectors are highly correlated. PCA is used to discover these correlations and in so doing find a lower-dimensional subspace in which faces may be represented.

The linear subspace found using PCA has several useful properties. One is dimensionality reduction, as just mentioned. Dimensionality reduction allows nearest-neighbor matching to be carried out in spaces of perhaps 100 dimensions rather than spaces with dimensionality of $10,000$ or more. This is possible because the first dimension represents the direction of maximal variation in the training data, the second dimension the second most significant direction relative to variation, and so on. The number of dimensions recovered by PCA is bounded above by the number of training images, and it is common to further truncate the space by removing those dimensions with the least variation in the training data.

Two additional useful properties of PCA are that it removes correlation between features and yields sample variance estimates for the principle axes. To be more precise, the linear subspace obtained using PCA has the property that the

1. Integer to float conversion – Converts 256 gray levels into floating-point equivalents.
2. Geometric normalization – Lines up human chosen eye coordinates.
3. Masking – Crops the image using an elliptical mask and image borders such that only the face from forehead to chin and cheek to cheek is visible.
4. Histogram equalization – Equalizes the histogram of the unmasked part of the image.
5. Pixel normalization – Scales the pixel values to have a mean of zero and a standard deviation of one.

Fig. 3. Normalized FERET image and summary of normalization procedure

sample covariance between features for the training images will be zero when those training images are projected into the subspace. Likewise, the sample variance for the training images will be the same as the sample variance estimate generated when PCA is initially performed. These two properties together make it easy in practice to compute the whitened distance measures defined in Sect. 4.2.2.

Training: PCA training is performed by `csuSubspaceTraining` running in default mode. The PCA basis is computed by the snapshot method using a Jacobi eigensolver from the Intel CV library. Some basis vectors can be eliminated from the subspace using the cutOff and dropNVectors command line options. These methods are described in detail in [19]. The training program outputs a training file that contains a description of the training parameters, the mean of the training image, the eigenvalues, and a basis for the subspace.

Distance metrics: Distance files are generated by `csuSubspaceProject`. It requires a list of images and a subspace training file. The code projects the feature vectors onto the basis and then computes the distance between pairs of images in the list. The output is a set of distance files containing the distance from each image to all other images in the list. The distance metrics include city block (L1), Euclidean (L2), correlation, covariance, whitened cosine (PCA only), and LDASoft (LDA only). These distances are defined below in Sect. 4.2.

4.2 Linear discriminant analysis (csuSubspace/PCA+LDA)

The second algorithm uses both principle component analysis and linear discriminant analysis. This PCA+LDA algorithm is based upon the work of Zhao and Chellapa [20]. LDA uses Fisher's linear discriminants to find a linear transformation that accentuates differences between classes while reducing differences within classes. The goal is a subspace in which classes are linearly separable. Each human person constitutes a distinct class, and therefore training must include multiple images per person.

As proposed by Zhao and Chellapa [20], PCA is used first to reduce dimensionality. The training images are projected into the PCA subspace and then LDA is used to find a further transformation that accentuates the differences between classes. The final result of training for the PCA+LDA algorithm is a single transformation matrix formed by multiplying the PCA and LDA basis vectors. The PCA+LDA algorithm then operates by projecting images into this combined PCA+LDA subspace and comparing images.

Training: PCA+LDA training is performed by the `csuSubspaceTraining` executable using the `-lda` option. PCA is first performed on the training data to determine an optimal basis for the image space. The training images are projected onto the PCA subspace to reduce their dimensionality before LDA is performed. Computationally LDA follows the method outlined in [8]. A detailed description of the implementation can be found in [3].

Distance metrics: Distance files for the PCA+LDA algorithm are generated using `csuSubspaceProject` and consquently essentially the same set of choices are available for both the PCA and the PCA+LDA algorithm.

Briefly, the standard and not-so-standard distance measures available in `csuSubspaceProject` are summarized below.

4.2.1 Common distance measures

The following four common distances measures are available in `csuSubspaceProject`.

CityBlock (L1).

$$D_{CityBlock}(u,v) = \sum_i |u_i - v_i| \,.$$

Euclidean (L2).

$$D_{Euclidean}(u,v) = \sqrt{\sum_i (u_i - v_i)^2}.$$

Correlation. This is a standard correlation measure between two vectors with one major caveat. Because all subsequent processing treats these measures as distance, correlation scores are mapped onto the range 0 to 2 such that a correlation score of 1 maps to 0 and a correlation of -1 maps to 2:

$$S_{Correlation}(u,v) = \frac{\sum_i (u_i - \bar{u})(v_i - \bar{v})}{(N-1)\sqrt{\frac{\sum_i (u_i-\bar{u})^2}{N-1}}\sqrt{\frac{\sum_i (v_i-\bar{v})^2}{N-1}}}$$

$$D_{Correlation}(u,v) = 1 - S_{Correlation}(u,v).$$

Covariance. This is the standard covariance definition and is the cosine of the angle between the two vectors. When the vectors are identical, it is 1.0, and when they are orthogonal, it is zero. Thus covariance and the associated distance are

$$S_{Covariance}(u,v) = \frac{\sum_i u_i v_i}{\sqrt{\sum_i u_i^2}\sqrt{\sum_i v_i^2}} \cdot$$

$$D_{Covariance}(u,v) = 1 - S_{Covariance}(u,v)$$

LDASoft. The soft distance measure proposed by Zhao [21] is a defined as

$$D_{LDASoft}(u,v) = \sum_i \lambda_i^{0.2}(u_i - v_i)^2.$$

This distance measure is particular to the PCA+LDA algorithm. It is essentially the L2 norm with each axis weighted by the associated generalized eigenvalue λ_i. To be more precise, λ_i is the generalized eigenvalue found when solving for the Fisher basis vectors. Raising λ_i to the 0.2 power is not an obvious choice, but considerable discussion of this choice appears in Zhao's dissertation [21]

4.2.2 Whitened distance measures and whitened cosine

The system also defines these standard measures in what we shall call a whitened space.[1] The first step in computing whitended distance measures is to understand the transformation between projected image space and whitened space.

Recall that the eigenvalues produced by PCA are the sample variance estimates for the dimensions defined by the associated eigenvectors, in other words the axes of the subspace. Whitened space is defined as a space where the sample variance along each dimension is one. Therefore, the transformation of a vector from projected image space to whitened space is performed by dividing each coefficient in the vector by its corresponding standard deviation. This transformation yields a dimensionless feature space with unit variance in each dimension.

To clarify the definitions below, let u and v be vectors defined in projected image space, in othere words the standard PCA space. Let m and n be the corresponding vectors in whitened space and λ_i be the i-th eigenvalue coming from PCA. Observe that $\sigma_i^2 = \lambda_i$, where σ_i is the standard deviation along the i-th dimension of the PCA subspace. The relationship between the vectors before and after whitening may be written as

$$m_i = \frac{u_i}{\sigma_i} \qquad n_i = \frac{v_i}{\sigma_i}.$$

Whitened L1. This measure is exactly the same as the city block measure, only the distances are scaled to whitened space. So for images u and v with corresponding projections m and n in whitened space, whitened L1 is

$$D_{WhtL1}(u,v) = \sum_i |m_i - n_i| .$$

[1] The naming convention is evolving, and what we call here whitened space is called Mahalanobis space in version 5.0 of the System [4].

Whitened L2. This measure is exactly like Euclidean distance only computed in whitened space, so for images u and v with corresponding projections m and n in whitened space, whitened L2 is:

$$D_{WhtL2}(u,v) = \sqrt{\sum_i (m_i - n_i)^2}.$$

Whitened cosine. Whitened cosine is the cosine of the angle between the images after they have been projected into the recognition space and further normalized by the variance estimates. Thus for images u and v with corresponding projections m and n in whitened space, the whitened cosine is

$$S_{WhtCosine}(u,v) = cos(\theta_{mn}) = \frac{|m|\,|n|\,cos(\theta_{mn})}{|m|\,|n|} = \frac{m \cdot n}{|m|\,|n|}$$

$$D_{WhtCosine}(u,v) = -S_{WhtCosine}(u,v).$$

While the other whitened measures are included largely for generality, the whitened cosine has proven remarkably useful and robust in the context of PCA and is our distance measure of choice for the PCA algorithm.

4.3 The intrapersonal/extrapersonal image difference classifier (csuBayesian/IIDC)

The third algorithm in the System is the intrapersonal/extrapersonal image difference classifier.[2] It is based on an algorithm developed by Moghaddam and Pentland [12]. There are two variants of this algorithm, a *maximum a posteriori* (MAP) and a *maximum likelihood* (ML) classifier. This algorithm examines the difference between two images as a basis for determining whether the two images are of the same person. Difference images that originate from two images of different people are said to be *extrapersonal*, whereas difference images that originate from two images of the same person are said to be *intrapersonal*.

The key assumption in Moghaddam and Pentland's work is that intrapersonal and extrapersonal difference images originate from distinct and localized Gaussian distributions within the space of possible difference images. The actual parameters for these distributions are not known a priori, so the algorithm begins by fitting a Gaussian distribution to the intrapersonal and extrapersonal difference images in the training data. This step, called density estimation, is performed using principle components analysis (PCA). During testing the classifier takes each image of unknown class membership and uses the estimated probability distributions for identification.

Training: The program `csuBayesianTrain` trains the Bayesian classifier. For the interpersonal and extrapersonal spaces, principle components are computed by the snapshot method using a Jacobi eigensolver from the Intel CV library. Selected basis vectors can be eliminated from the subspace using the cutOff and dropNVectors command line options. These methods are described in detail in [19].

[2] This classifier has been called the Bayesian image classifier in some of our earlier papers.

The training program outputs two training files, one for each subspace. The training files contain a description of the training parameters, the mean training image, the eigenvalues, and a set of basis vectors for the subspace.

Distance metrics: The `csuBayesianProject` code is used to generate distance files. It requires a list of images and two subspace training files that are written by `csuBayesianTrain` (one for the extrapersonal difference images and another for the intrapersonal difference images). The code projects the feature vectors onto each of the two sets of basis vectors and then computes the probability that each feature vector came from one or the other subspace. The output is a set of distance files containing the distance from each image to all other images. The similarities may be computed using the maximum a posteriori (MAP) or maximum likelihood (ML) methods. From a practical standpoint, the ML method uses information derived only from the intrapersonal images, while the MAP method uses information derived from both distributions.

The CSU implementation of the IIDC algorithm was written by Marcio Teixiera; details may be found in [17].

4.4 The elastic bunch graph matching algorithm (csuEBGM/EBGM)

The CSU EBGM is based on an algorithm from the University of Southern California [14]. The algorithm locates landmarks on an image such as the eyes, nose, and mouth. Gabor jets are extracted from each landmark and are used to form a face graph for each image. Face graphs serve the same role as the projected image vectors in the PCA or PCA+LDA algorithm – they represent the image in a low-dimensional space. After a face graph has been created for each test image, the algorithm measures the similarity of the face graphs.

The EBGM algorithm uses a special process to normalize its images. The standard normalization performs masking by setting to zero pixels outside an oval bordering the face. This masking introduced a very hard and spatially defined edge, and, for lack of a more precise term, it causes the wavelets to ring. It has been our experience that the EBGM algorithm performs very poorly when used with images masked in this fashion, and so we have modified the masking operation to soften the edge. The images normalized for the algorithm can be produced by running the script `EBGMPreprocessing.sh`. This script calls `csuPreprocessNormalize` with the special parameters needed to create EBGM images.

The EBGM algorithm is divided into three executables that perform landmark localization, face graph creation, and similarity measurement. These programs should be run using the script `EBGM_Standard.sh`.

Landmark localization: The landmark localization process automatically adjusts the placement of landmarks on novel images. Landmarks are found using the program `csuEBGMGraphFit`. The output of this program is a file for each novel image that contains the automatically located landmark locations. This program has two high-level parts: bunch graph creation and landmark localization.

Face graph creation: The program `csuEBGMFaceGraph` uses the automatically located landmark locations from the previous process and normalized imagery to create face graphs for every image in the database.

Similarity measurement: The program `csuEBGMMeasure` compares face graphs and generates a distance matrix.

The CSU implementation of the EBGM algorithm was written by David Bolme; details may be found in [7].

5 Analysis tools: overview

Two analysis tools are provided with the System. The first tool, called `csuAnalyzeRankCurve`, generates cumulative match curves of the type commonly reported in the FERET evaluation [15] and the later Vendor Tests [6,16]. The other tool, `csuAnalyzePermute`, permutes probe and gallery image choices to generate a sample probability distribution for recognition rates at different recognition ranks. It may be used to perform nonparametric analysis such as that used when comparing PCA to PCA+LDA [5]. A general overview of each tool is provided in this section. Additional details and specific examples are provided in Sect. 6.

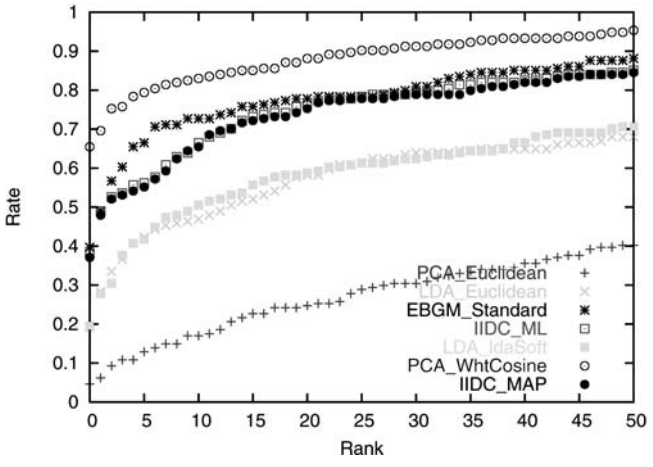
5.1 CSU rank curve

The System has been developed to evaluate how well face recognition algorithms perform identification, and as already suggested above, the identification problem is to match up a novel image of a face to a stored face image of the same person. The identification task should not be confused with the verification task, which is to determine if a person is who they claim based upon the quality of match between a novel image of the person and a stored image of the person. For identification, it is assumed that a corresponding image of the person to be recognized is in the gallery of images being searched. Evaluation centers on how many stored images must be examined before the correct match is found, and this is reported using cumulative match characteristic (CMC) curves such as those shown in Fig. 4.

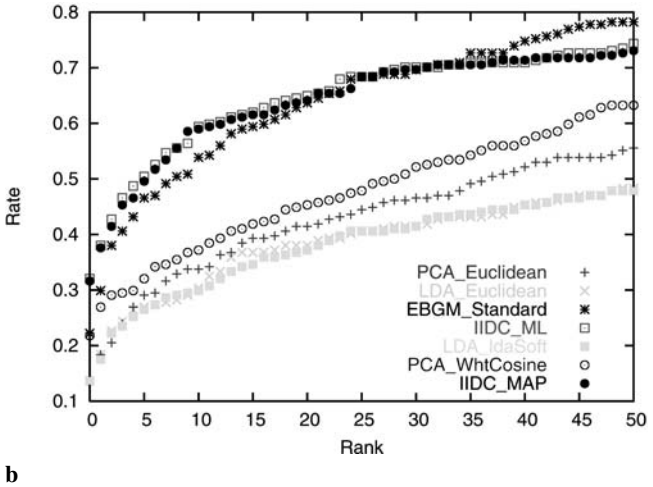
To be more precise, following the FERET protocol [15], a set of algorithms are compared using a probe and gallery set. For each probe image, there is a corresponding gallery image of the same person. In order to generate a CMC curve, the gallery is sorted by decreasing similarity for each probe image, and the probe image is said to be correctly recognized at rank k if the gallery image of the same person is among the first k images in the sorted gallery. The horizontal axis of a CMC curve is the rank variable k and runs from rank 1 up through the number of images in the gallery. The vertical axis is the number, or fraction, of probe images correctly recognized at that rank. The rank curve tool `csuAnalyzeRankCurve` generates CMC curves.

5.2 CSU permute

A weakness of the CMC analysis described above is that one cannot draw any conclusions about when an observed difference between algorithms is statistically meaningful. The



a



b

Fig. 4a,b. Cumulative match curves. **a** FERET FC probe set. **b** FERET Dup 2 probe set

permutation of gallery and probe images methodology implemented by the `csuAnalyzePermute` is one way of developing an explicit model of uncertainty associated with the recognition rate estimates used in CMC curves. It thus allows one's analysis to go deeper and answer a very basic question: What might have happened if I had tested on a different random selection of probe and gallery images?

The `csuAnalyzePermute` tool may be thought of as performing many virtual experiments using a common set of distance files and producing sample probability distributions for recognition rates under the assumption that probe and gallery images are exchangeable for each individual person. It does this by randomly sampling probe and gallery sets from a larger set of test images. Distance files must already be computed, and there must be at least three images of each person in the study. Our FERET studies use image lists with four replicates per person. The program `csuAnalyzePermute` randomly assembles probe and gallery sets and computes recognition rates. It typically does this 10,000 times.

A structured sampling pattern is used when randomly selecting probe and gallery images. Imagine a table of image names, one row per person with replicate images arrayed in the columns as shown above in Sect. 2.1. A fixed sampling pattern is defined such that the number of times a probe or gallery image is selected from a column is the same for each column, or at least as nearly equal as possible. So, for example, in the first row the first column image may be chosen as the probe image and the third column image as the gallery image. Randomness is introduced by permuting the order of the rows, i.e., people. The name `csuAnalyzePermute` derives from this permutation step. To read more about the methodology lying behind `csuAnalyzePermute` and how it may be used to characterize the performance of an algorithm, see [5].

The `csuAnalyzePermute` tool generates sample probability distributions for each of a set of algorithms being compared. There is a distinct probability distribution for each rank, and consequently it is possible to define standard error bars on a CMC curve, as illustrated in Fig. 5.[3] Sample probability distributions are also generated for differences in recognition rates between algorithms, and this provides a more discriminating means of determining when the difference between two algorithms is statistically significant.

6 Analysis tools: details and examples

This section provides another level of detail of interest to some and certainly of interest to anyone actually wishing to use the tools.

6.1 CSU rank curve details and example

The rank curve tool `csuAnalyzeRankCurve` takes a probe and gallery image list. It also takes a set of algorithms to compare or, more precisely, a list of one or more directories containing distance matrices represented as sets of files as defined above in Sect. 2.2. Two tab-delimited ASCII files are produced by `csuAnalyzeRankCurve`. The first records the rank of the closest gallery image of the same person for each probe image, and the second records the recognition counts and rates for each algorithm. These files may be loaded into standard spreadsheet programs.

To better understand what these files contain and how they may be used, let us provide a detailed example associated with our standard FERET script `runAllTests_feret.sh`. The two ASCII files generated for the dup2 probe set are `DUP2_images.txt` and `DUP2_Curve.txt`. Table 1 shows the first ten rows of each file. For the file `DUP2_images.txt`, the first row contains the column headers. The first column contains the names of the probe images in the analysis, and the remaining columns summarize recognition rank data for each algorithm and distance measure combination being studied. For the sake of space, in the table, names are shortened and the full names are given in the legend.

The values in columns 2, 3, etc. of file `DUP2_images.txt` are interpreted as follows. A zero means that for the corresponding probe image and distance measure, the first match of a gallery image of the same person is found at position zero in the gallery sorted by increasing distance relative to the probe image. This is a somewhat long way of saying that zero means the best match is of the same person. Likewise, a one means there is one gallery image of another person

[3] An alternative means of computing error bars has been developed by Micheals and Boult [11].

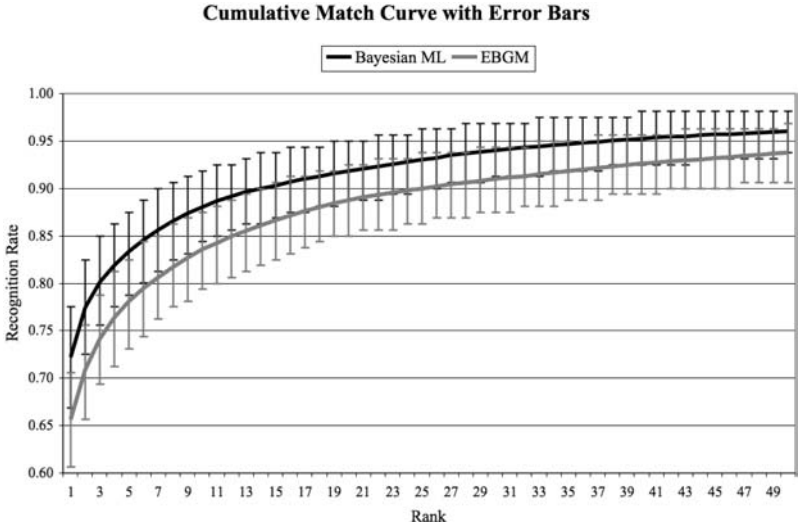


Fig. 5. This figure compares our EBGM and Bayesian algorithms. The rank *curves* include 95% error bars that were estimated by `csuAnalyzePermute`

that is closer than the gallery image of the same person, and so on. This value is commonly called the recognition rank. Now, perhaps unfortunately, it has become common to speak of recognition at rank one when speaking of the best match, and this is a classic one-based versus zero-based counting issue. So, while the file carefully gives a zero, you will find us and others counting a zero as a successful recognition at rank one.

The recognition rank files are worth studying in their own right. Observe probe image `00019ba010_960521.sfi` [4] in file `DUP2_images.txt`. The difference between a PCA algorithm using Euclidean distance and one using whitened cosine is the difference between a recognition rank of 59 versus 0. In other words, for this particular probe image, PCA does perfectly using whitened cosine and relatively poorly using Euclidean distance. Also note that large recognition ranks do appear. For `DUP2_images.txt`, the gallery contains 1,196 images, and recognition ranks in the seven and eight hundreds are observed.

The recognition rank file also provides some insight into how the cumulative match curves are generated. To compute the recognition count, i.e., how many images are correctly recognized by going a particular depth into the list of sorted gallery images, scan down a column and count how often the recognition rank is less than or equal to the desired depth.

The second file produced by `csuAnalyzeRankCurve` contains recognition counts and associated recognition rates at different ranks. Rank runs from zero to the number of probe images in the probe set. Scanning a given row one sees the different recognition rates for each algorithm at a given rank. In the file `DUP2_Curve.txt` in Table 1, the rank one recognition rate[5] for the IIDC algorithm using the ML distance measure is 0.32 – higher than any other algorithm distance measure combination. The dup2 FERET probe set is a difficult test.

The `DUP2_Curve.txt` file is easily used to generate standard cumulative match curves. The version 5.0 distribution of the System includes a short Python program that uses GNUplot to generate cumulative match curves. Examples for two of the four standard FERET probe sets are shown in Fig. 4. These CMC curves are similar but not identical to those shown in original FERET evaluation [15]. Some differences are expected since the System is using new algorithm implementations, new image preprocessing code, and, perhaps most importantly, different training image sets.

Keeping these caveats in mind, the System includes scripts that will preprocess the FERET imagery, train the algorithms, run the algorithms to generate distance matrices, and, finally, build CMC curves for the standard set of FERET gallery images and each of the four standard FERET probe image sets. This script is a good baseline, or point of departure, for people wanting to understand what was done in the FERET evaluation and wanting to adapt it to their own purposes. In particular, all a researcher need do to compare their own algorithm to a standard algorithm is generate a comparable distance matrix in an appropriately named directory, e.g., "AMN" for amazing new algorithm, and include the new distance matrix/directory in the list given to `csuAnalyzeRankCurve`.

6.2 CSU permute details and example

The `csuAnalyzePermute` tool takes as an argument a list of distance matrices or, more precisely, a list of directories containing distance files. It will compute three files for each distance measure along with a summary file. As before, it is probably easiest to explain these files by example. So, for example, the file `PermBayesian_ML_HistCounts.txt`[6] corresponds to the IIDC algorithm using the maximum likelihood (ML) distance. Figure 6 shows a portion of this file loaded into Microsoft Excel.

The top row of Fig. 6 gives the column headers, and the first column, rc, is an exact recognition count running from 0 to

[4] These image names are the original NIST-generated FERET image names, with the `sfi` suffix indicating it is a single floating-point image.

[5] This is an example of the zero-based versus one-based counting issue already mentioned.

[6] This example was generated prior to the change of naming from "Bayesian" to "IIDC".

Table 1. First ten rows in files `DUP2_images.txt` and `DUP2_Curve.txt`

File `DUP2_images.txt`

ProbeName	Alg.1	Alg.2	Alg.3	Alg.4.	Alg.5	Alg.6	Alg.7
00019ba010_960521.sfi	9	12	81	460	481	59	0
00019bj010_960521.sfi	404	463	280	828	800	423	228
00029ba010_960521.sfi	0	0	21	21	11	8	0
00029bj010_960521.sfi	1	0	41	222	211	35	35
00029fa010_960530.sfi	0	0	668	2	3	77	403
00029fa010_960620.sfi	3	2	778	198	136	105	650
00029fa010_960627.sfi	0	0	412	189	224	38	192
00029fb010_960530.sfi	0	0	550	1	2	99	193
00029fb010_960620.sfi	0	0	743	27	24	114	927

File `DUP2_Curve.txt`

Rank	Alg.1		Alg.2		Alg.3		Alg.4		Alg.5		Alg.6		Alg.7	
0	74	0.31	75	0.32	52	0.22	32	0.13	32	0.13	33	0.14	51	0.21
1	88	0.37	89	0.38	70	0.29	43	0.18	41	0.17	43	0.18	63	0.26
2	97	0.41	100	0.42	89	0.38	53	0.22	52	0.22	48	0.20	68	0.29
3	106	0.45	109	0.46	95	0.40	57	0.24	55	0.23	56	0.23	69	0.29
4	109	0.46	114	0.48	101	0.43	62	0.26	59	0.25	63	0.26	70	0.29
5	116	0.49	118	0.50	109	0.46	63	0.26	62	0.26	68	0.29	75	0.32
6	121	0.51	123	0.52	110	0.47	64	0.27	64	0.27	69	0.29	80	0.34
7	125	0.53	128	0.54	115	0.49	65	0.27	67	0.28	74	0.31	81	0.34
8	130	0.55	130	0.55	118	0.50	66	0.28	68	0.29	77	0.32	83	0.35
9	137	0.58	132	0.56	119	0.50	68	0.29	69	0.29	79	0.33	86	0.36

Algorithm legend for columns

Alg.1	`distances/feret/Bayesian_MAP`
Alg.2	`distances/feret/Bayesian_ML`
Alg.3	`distances/feret/EBGM_Standard`
Alg.4	`distances/feret/LDA_Euclidean`
Alg.5	`distances/feret/LDA_ldaSoft`
Alg.6	`distances/feret/PCA_Euclidean`
Alg.7	`distances/feret/PCA_WhtCosine.`

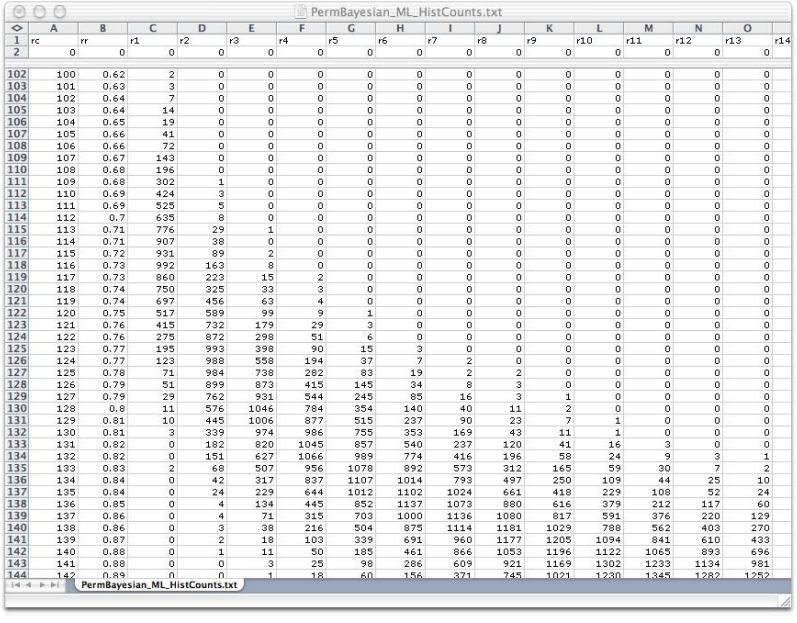


Fig. 6. Portions of the sample recognition rate distribution file `PermBayesian_ML_HistCounts.txt` containing the raw histogram counts of how often an algorithm recognized exactly k people at rank 1, 2, etc.

the number of probe images. The second column is the recognition rate: recognition count divided by the number of probe images. The third column, labeled $r1$, indicates how many times out of $10,000$ randomly generated probe and gallery sets the algorithm correctly recognized exactly rc people. So, looking at the upper left portion of the top table, we see that the IIDC algorithm using the ML distance exactly recognized 100 people 2 times in $10,000$ trials. Scanning down the $r1$ column, the histogram peak is 992 at recognition count 116, indicating that 116 people were recognized correctly more often than any other single number of people.

The second file generated by `csuAnalyzePermute` is a minor variant of the first, where the raw recognition counts are divided through by the number of random trials: $10,000$. However, the interpretation of this file is now very elegant: each column starting with $r1$ is a sample probability distribution for recognition count or, equivalently, recognition rate.

Given the sample probability distributions, it is a relatively simple matter to compute the average recognition rate at each rank, as well as the mode of the distribution at each rank. Perhaps more importantly, it is also possible to compute a two-sided confidence interval. This is done by simply identifying the central portion of each distribution accounting for 95% of the outcomes. To be even more precise, to find the lower bound on the confidence interval, our algorithm starts at rc equal to 0 and sums the values in the histogram until reaching 250, the cutoff 250 being 0.025% of 10,000. The analogous operation is performed starting from the top to compute the upper bound.

The third file produced by `csuAnalyzePermute` does what we have just described and summarizes the sample distribution by providing the average, mode, and upper and lower confidence bounds on recognition count for rank 1, 2, etc. It is essentially a cumulative match curve with 95% confidence error bars. The first ten lines of the file `PermBayesian_ML_CMCurve.txt` are shown in Table 2. The first row contains column headers, and the columns are: recognition rank, lower bound on the 95% confidence interval, mode of probability distribution, upper bound on confidence interval, and mean recognition count. From these files, it is relatively simple to generate cumulative match curves with associated error bars. However, note that this file records recognition counts, and all values must be divided by the number of probe images to arrive at recognition rates.

Figure 5 shows these curves for the IIDC algorithm using the ML distance compared to the standard EBGM algorithm. This comparison is over a set of 640 FERET images of 160 people where the random permutations of probe and gallery images were selected $10,000$ times. The error bars overlap considerably over the entire range of the curve, and certainly at rank 1. Were this the only information available to us, it would be likely we would be forced to conclude that the difference between the two algorithms is not significant.

However, as we discuss in more detail in [5], observing whether error bars overlap is a very weak and not terribly appropriate test. Better is to compute the actual difference in recognition rates for each of the $10,000$ trials, and then look at the distribution of this new random variable. Interpreting the resulting distribution is relatively easy, since a predominance of positive values indicates one algorithm is better, a predominance of negative values indicates the other algorithm is better, and distributions centered around zero indicates there is no difference between the algorithms. The program `csuAnalyzePermute` carries out this test for all pairs of algorithm/distance matrices for which it is given. The results are tabulated and placed on a Web page.

Table 2. First ten lines of the file `PermBayesian_ML_CMCurve.txt`. This file provides a cumulative match curve with error bars

Rank	Lower	Mode	Upper	Mean
1	107	116	124	115.5
2	116	123	132	123.9
3	121	128	136	128.2
4	124	132	138	131.1
5	126	134	140	133.4
6	128	136	142	135.3
7	130	137	144	137.0
8	132	138	145	138.5
9	133	139	146	139.8

For the pairwise test between the Bayesian and EBGM algorithm, the paired test is more discriminating than checking for overlap in error bars in Fig. 5. The mode of the distribution for the IIDC recognition rate minus the EBGM recognition rate is 11, indicating that 11 more images were correctly recognized more often than any other number. More importantly, the EBGM algorithm had the higher recognition rate in only 213 out of the $10,000$ trials, and in the other $9,787$ trials IIDC did better. This outcome can be translated into a one-sided test of statistical significance. To be more precise, given the null hypothesis that there is no difference between the algorithms, given the observation that $9,787$ out of $10,000$ times IIDC did better, we can reject the null hypothesis: the probability of the observed outcome given the null hypothesis is less than 0.05.

7 Conclusion

The CSU Face Identification Evaluation System includes everything necessary to compare new face recognition algorithms to baseline algorithms using the protocol of the FERET evaluation [15]. It also includes new statistical tools for estimating recognition rate probability distributions under the assumption that probe and gallery images are exchangeable. It has been available publicly since 2001 and been downloaded over $5,000$ times since November 2002. We hope the System will provide a solid baseline for comparing future face recognition algorithms.

Our work studying and refining the four face recognition algorithms included in the System continues; for the IIDC and EBGM algorithms in particular, there is still more that needs to be understood about algorithm configuration. Performance on the four standard FERET probe sets for the IIDC and EBGM algorithms is not as high as that achieved by the original creators of the algorithms and captured in the FERET tests. This strongly suggests that training and tuning play a significant role in how these algorithms perform in practice.

One of the larger lessons emerging from our efforts is that replication of results obtained by others is not as easy as many presume. There is an illusion among some in computer science that because we work with algorithms that have concrete real-

izations as software, replication of major experiments should be a trivial matter not worthy of serious effort or study. This is an illusion, and it neglects the many many factors left out of even the best-written archival accounts of excellent research.

The CSU Face Identification Evaluation System has been developed in part to address this issue of experimental replication. It is our hope that in the near future, when two authors of different papers compare new algorithms to a standard, they will use our implementations. If they do, then those of us reading these papers can be much more confident that the standard will the same across the two papers. This may seem a modest goal, but it will greatly aid all of us when interpreting results.

The CSU Face Identification Evaluation System has also been developed to advance the practice of algorithm comparison. At present, most researchers simply report that algorithm A, typically theirs, got a recognition rate of x, while the recognition rate for algorithm B is y. The obvious question left unaddressed: Is the difference between x and y significant? This actually is a very subtle question with many different answers conditioned upon how the question is formalized. However, that said, one obvious source of variation is changes in the probe and gallery images, and the permutation of probe and gallery image test provided in the System is one excellent way of measuring this common source of variation and allowing one to assess when differences in recognition rate are significant relative to these variations.

Acknowledgements. This work is supported by the Defense Advanced Research Projects Agency under contract DABT63-00-1-0007.

References

1. Bartlett MS, Draper B, Baek K, Beveridge R (July 2003) Recognizing faces with pca and ica. Comput Vision Image Understand 91:115–137
2. Beveridge JR (2003) Evaluation of face recognition algorithms Web site. `http:\\www.cs.colostate.edu\evalfacerec`
3. Beveridge JR (2001) The geometry of LDA and PCA classifiers illustrated with 3D examples. Technical Report CS-01-101, Computer Science, Colorado State University
4. Beveridge JR, Bolme D, Teixeira M, Draper BA (2003) The CSU Face Identification Evaluation System User's Guide: Version 5.0. Technical report, Colorado State University
5. Beveridge JR, She K, Draper B, Givens GH (December 2001) A nonparametric statistical comparison of principal component and linear discriminant subspaces for face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 535–542
6. Blackburn DM, Bone M, Phillips PJ (2003) Facial Recognition Vendor Test 2000: executive overview. Technical report, Face Recognition Vendor Test (www.frvt.org)
7. Bolme DS (June 2003) Elastic bunch graph matching. Master's thesis, Computer Science, Colorado State University
8. Duda RO, Hart PE, Stork DG (2001) Pattern classification, 2nd edn. Wiley, New York
9. Givens G, Beveridge JR, Draper BA, Phillips PJ, Grother P (2004) How features of the human face affect recognition: a statistical comparison of three face recognition algorithms. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2004 (to appear)
10. Givens GH, Beveridge JR, Draper BA, Bolme D (2003) Using a generalized linear mixed model to study the configuration space of a PCA+LDA human face recognition algorithm. Technical report, Computer Science, Colorado State University
11. Micheals RJ, Boult T (December 2001) Efficient evaluation of classification and recognition systems. In: Proceedings of the IEEE conference on computer vision and pattern recognition 2001, I:50–57
12. Moghaddam B, Nastar C, Pentland A (1996) A Bayesian similarity measure for direct image matching. In: Proceedings of ICPR, B:350–358
13. Moon H, Phillips J (1998) Analysis of pca-based face recognition algorithms. In: Boyer K, Phillips J (eds) Empirical evaluation techniques in computer vision. IEEE Press, New York
14. Okada K, Steffens J, Maurer T, Hong H, Elagin E, Neven H, von der Malsburg C (1998) The Bochum/USC face recognition system and how it fared in the FERET Phase III test. In: Wechsler H, Phillips PJ, Bruce V, Soulié FF, Huang TS (eds) Face recognition: from theory to applications. Springer, Berlin Heidelberg New York, pp 186–205
15. Phillips PJ, Moon HJ, Rizvi SA, Rauss PJ (October 2000) The FERET evaluation methodology for face-recognition algorithms. Trans Pattern Anal Mach Intell 22(10):1090–1104
16. Phillips PJ, Grother P, Micheals RJ, Blackburn DM, Tabassi E, Bone JM (2002) FRVT 2002: overview and summary. Technical report, Face Recognition Vendor Test 2002 (www.frvt.org)
17. Teixeira ML (July 2003) The Bayesian intrapersonal/extrapersonal classifier. Master's thesis, Computer Science, Colorado State University
18. Turk MA, Pentland AP (June 1991) Face recognition using eigenfaces. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 586 – 591
19. Yambor WS (2000) Analysis of PCA-based and Fisher discriminant-based image recognition algorithms. Master's thesis, Colorado State University
20. Zhao W, Chellappa R, Krishnaswamy A (1998) Discriminant analysis of principal components for face recognition. In: Wechsler, Philips, Bruce, Fogelman-Soulie, Huang (eds) Face recognition: from theory to applications, Springer, Berlin Heidelberg New York, pp 73–85
21. Zhao W, Chellappa R, Phillips PJ (1999) Subspace linear discriminant analysis for face recognition. University of Maryland, Computer Science Department