

# The CSU Face Identification Evaluation System User's Guide: Version 4.0\*

David Bolme, Marcio Teixeira, J Ross Beveridge and Bruce Draper  
Computer Science Department Colorado State University

October 31, 2002

## Abstract

The CSU Face Identification Evaluation System provides standard face recognition algorithms and standard statistical methods for comparing face recognition algorithms. This document describes Version 4.0 the Colorado State University (CSU) Face Identification Evaluation System. The system includes standardized image pre-processing software, three distinct face recognition algorithms, analysis software to study algorithm performance, and Unix shell scripts to run standard experiments. All code is written in ANSI C. The preprocessing code replicates feature of preprocessing used in the FERET evaluations. The three algorithms provided are Principle Components Analysis (PCA), a.k.a Eigenfaces, a combined Principle Components Analysis and Linear Discriminant Analysis algorithm (PCA+LDA), and a Bayesian Intrapersonal/Extrapersonal Classifier (BIC). The PCA+LDA and BIC algorithms are based upon algorithms used in the FERET study contributed by the University of Maryland and MIT respectively. Two different analysis programs are included in the evaluation system. The first takes as input a set of probe images, a set of gallery images, and similarity matrix produced by one of the three algorithms. It generates a Cumulative Match Curve that plots recognition rate as a function of recognition rank. These plots are common in evaluations such as the FERET evaluation and the Face Recognition Vendor Tests. The second analysis tool generates a sample probability distribution for recognition rate at recognition rank 1, 2, etc. It takes as input multiple images per subject, and uses Monte Carlo sampling in the space of possible probe and gallery choices. This procedure will, among other things, add standard error bars to a Cumulative Match Curve. The CSU Face Identification Evaluation System is available through our website 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.

## 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. The related verification problem is: given a novel image of specific person, confirm whether the person is or is not who they claim to be.

As in the original FERET tests run in 1996 and 1997, Face Identification and Evaluation System presumes a face recognition algorithm will first compute a similarity measure between images, and second perform a nearest neighbor match between novel and stored images. When this is true, the complete behavior of a face identification system can be captured in terms of a similarity matrix. Face Identification and Evaluation System will create these

---

\*An abbreviated version of this Guide with somewhat more attention to purpose and less to practical details has been submitted to the International Conference on Computer Vision Systems 2003 under the title "The CSU Face Identification Evaluation System: Its Purpose, Features and Structure".

similarity matrices and provides analysis tools that operate on these matrices. These analysis tool currently support rank curve generation and permutation testing.

Earlier releases of the CSU Face Identification Evaluation System has been available through the web. The first release was made in March 2001. Starting in December of 2001, releases have been identified by version numbers. This document describes version 4.0. Since this system is still a work in progress, those using it are encouraged to seek the most recent version. The latest releases along with additional information about the CSU evaluation work is available at our website[2].

## 2 Installation

The CSU Face Identification Evaluation System code is primarily written in ANSI Standard C. As such, it should be easily used on almost any modern platform. Our distribution also includes scripts for compiling the system, running tests, analysis and plotting. These scripts utilize the GNU make utility and the sh Unix shell. Lastly, the plotting script uses Python and GNUplot. Thus, while the code itself does not presume a Unix platform, many of the features in this distribution do assume one is running some flavor of Unix. We have tested the distribution under Red Hat Linux 7.3, Solaris 5.8 and Mac OS X 10.1.

The CSU Face Identification Evaluation System does not install any components in the standard Unix directory tree. In other words, it will only modify the directory where it is unpacked. To create the executables all you have to do is unpack the tar file and run make in the top level directory. The will be created in the “bin” directory in a folder named after the machine architecture that it was compiled on. All source code for the system can be found under the directory “src”.

### 2.1 Testing the system

After installing the system, you can test it by using the “csuScrapShots” sample data which we provide for testing. To do so, type “scripts/runAllTests\_scraps.sh” from the top level directory. This script will run the various executables which make up the system and test various system components. If it executes completely, without generating an error message, you can be assured that the system is operating correctly. A warning, the “csuScrapShots” face data is testing code only. It is **not** to be used in any real evaluation. It was compiled by us from CSU yearbooks dating prior to 1927, thus clearing us of any copyright concerns. However, the data is poor quality and there is not enough to conduct a meaningful study. We encourage those using our system to obtain their own copy of the FERET image data[4].

### 2.2 Scripts

The “scripts” directory contains various scripts for normalizing the images and for running experiments. These scripts come in two different versions. The one’s suffixed by “feret” execute tests on a standard FERET image set (not provided) while the ones suffixed by “scraps” execute tests on the much smaller “csuScrapShots” image set which is provided with this distribution for testing purposes. Please see the “README.txt” file in the “scripts” directory for a more complete overview of the scripts provided in this release.

## 3 System Overview

CSU Face Identification Evaluation System can be split into four basic phases: image data preprocessing, algorithm training, algorithm testing and analysis of results: see Figure1. Preprocessing reduces unwanted image variation by aligning the face imagery, equalizing the pixel values, and normalizing the contrast and brightness. All three algorithms supported in this distribution may be meaningfully broken into a training phase and a testing phase. The training phase reads training data and creates a subspace into which test images will be projected

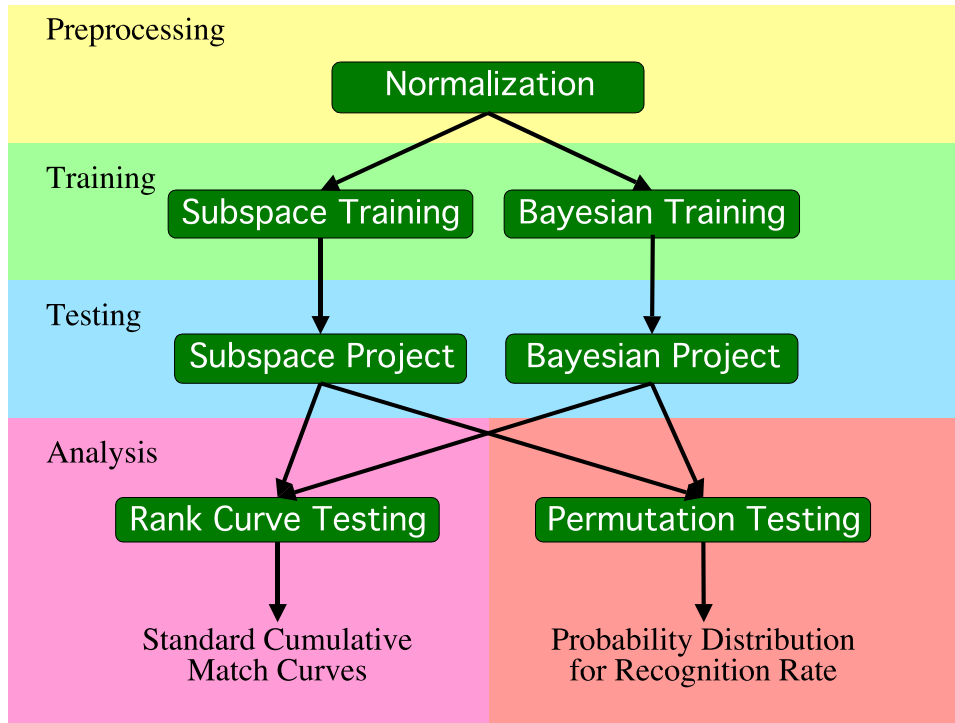


Figure 1: Overview of execution flow for the csuSubspace system, which includes a standard PCA identification algorithm and a PCA+LDA identification algorithm. Both of these are implemented csuSubspace, the LDA phase is an optional step in the training. There is also a Bayesian Intrapersonal/Extrapersonal Difference Image Classifier.

and matched. The testing phase reads the subspace information, projects images into this subspace, and generates a distance matrix. Typically the testing phase creates a distance matrix for the union of all images to be used either as probe images or gallery images in the analysis phase. The fourth phase performs analysis on the distance matrices. This include computing recognition rates (csuAnalyzeRankCurve), conducting virtual experiments (csuAnalyzePermute), or performing other statistical analysis on the data.

### 3.1 Imagery

The CSU Face Identification Evaluation System was developed using frontal facial images from the FERET data set. Images normalized using the csuPreprocessNormalize utility are written out to an image file that contains pixel values in a binary floating point format (sun byte order). Older versions of our system used a suffix “.nrm” to designate these files. The current system generates “.sfi” files. These are very similar, the distinction is as follows:

#### 3.1.1 SFI Images

SFI stands for Single Float Image. This format is the standard for the CSU face recognition project. It was created at CSU for the purpose of being a floating point version of pgm. It consist of multi channel raw floating point image data that is stored in Sun byte order. The images have a simple format similar to pgm with a short ASCII header (format\_id, width, height, channels) followed by (32 bit) binary floating point values.

### 3.1.2 NRM Images

NRM is the designation adopted at CSU for images generated by the old NIST normalization code. This is a legacy format and is kept around for compatibility purposes. It is very similar to SFI only with out the header. This format is only supported by part of the CSU distribution. It is recommended that old codes are modified to use the SFI replacement.

## 3.2 Image Lists

It is impossible to run experiments without first identifying sets of images to be used to train and test algorithms. Thus, we need files that store these image lists. In the original FERET evaluation, image list files were simple ASCII files with one image name per line. Some of our algorithms need to be able to know which images are of the same person, and this gave rise to a new file formatting convention. Specifically, each line contains one or more image of the same subject. Moreover, all images of a given subject must appear in one line. This convention will generalize to other data sets without hard coding subject naming conventions in the algorithms themselves. Therefore an image list should have the form:

```
SubjectA_Image1.sfi SubjectA_Image2.sfi SubjectA_Image3.sfi
SubjectB_Image1.sfi SubjectB_Image2.sfi SubjectB_Image3.sfi
SubjectC_Image1.sfi SubjectC_Image2.sfi SubjectC_Image3.sfi
SubjectD_Image1.sfi SubjectD_Image2.sfi SubjectD_Image3.sfi
```

csuReplicates will produce a file of this form if 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. See usage statements for more information. This distribution includes many common image lists, including the training images, gallery images, and four standard probe sets used in the FERET evaluations.

## 3.3 Distance Files

Each algorithm produces a distance matrix for all of the images in the testing list. This matrix is split up into distance files. One file is produced for every image in the list. Each line in these file contain the name of another image and the distance to that image. The file has the same name as the "probe image" and is placed in a distance directory.

All algorithms assume that smaller distances are a closer match. In many cases the base metric will yield a similarity score, where higher similarity yields higher scores. When this is the case the similarity values are negated to produce a “distance like” metric. Some examples of this are the Correlation and MahAngle distance metrics in csuSubpace, and the Bayesian and Maximum Likelihood Metrics in the csuBayesian code.

## 3.4 Documentation

The most current documentation on the executables can be obtained by running the executable with the special command line argument “-help”. This command interface is updated with the code and will produce a descriptive usage statements that covers all of the arguments and options that the executables expect.

## 4 Preprocessing

Preprocessing is conducted using the executable csuPreprocessNormalize. The executable performs five steps in converting a PGM FERET image to a normalized image. The normalization schedule is:



Figure 2: Example of a normalized FERET Image

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.

For an example image see Figure2

Our `csuPreprocessNormalize` is a newly written piece of code that accomplishes many of the same tasks performed by code originally written at NIST called “`facetonorm`”. The NIST preprocessing code was used in the FERET evaluation and a slightly modified version was released in our earlier code releases. There were many problematic features of the NIST code, including a tendency to core dump for some combinations of arguments. Our new version is much more robust. It is not identical in function to the NIST code. For example, histogram equalization in the new version is done only over the unmasked portions of the face. We would recommend using this newer version.

## 5 Algorithms

Version 4.0 of the Face Identification and Evaluation System comes with three face identification algorithms. These algorithms were chosen because they are well known and had high scores on the FERET Phase 3 test. The algorithms are intended to perform as a test platform for evaluation techniques and to serve as a common baseline for algorithm comparisons.

### 5.1 Principle Components Analysis (`csuSubspace/PCA`)

The first algorithm released by CSU was based on Principle Components Analysis (PCA)[6]. This system is based on a linear transformation in feature space. Feature vectors for the PCA algorithm are formed by concatenating the pixel values from the images. These raw feature vectors are very large (~20,000 values) and are highly correlated. PCA rotates feature vectors from this large, highly correlated subspace to a small subspace which has no sample covariance between features.

PCA has two useful properties when used in face recognition. The first is that it can be used to reduce the dimensionality of the feature vectors. This dimensionality reduction can be performed in either a lossy or lossless

manor. When applied in a lossy manor, basis vectors are truncated from the front or back of the transformation matrix. It is assumed that these vectors correspond to not useful information such as lighting variations (when dropped from the front) or noise (when dropped from the back). If none of the basis vectors are dropped it is called a lossless transformation and it should be possible to get perfect reconstruction for the training data based on the compressed feature vectors.

The second useful property is that PCA eliminates all of the statistical covariance in the transformed feature vectors. This means that the covariance matrix for the transformed (training) feature vectors will always be diagonal. This property is exploited for some distance measures such as L1, MahAngle, and Bayesian based classifiers.

**Training** PCA training is performed by the `csuSubspaceTraining` executable. The PCA is the default mode (it can also perform LDA training). The PCA basis is computed by the `snapshot` method using a Jacobi eigensolver from the Intel CV library. The basis vectors can be eliminated from the subspace using the `cutOff` and `dropNVectors` command line options. These methods are described in detail in[11]. The training program outputs a training file that contains a description of the training parameters, the mean of the training image, the eigenvalues or fisher values, and a basis for the subspace.

**Distance Metrics** The `csuSubspaceProject` code is used to generate distance files. It requires a list of images and a subspace training file. The code projects the feature vectors onto the basis. It 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, Mahalanobis Angle (PCA only), and LDA Soft (LDA only).

## 5.2 Linear Discriminant Analysis (csuSubspace/PCA+LDA)

The second algorithm is Linear Discriminant Analysis (PCA+LDA) based upon that written by Zhao and Chellapa[12]. The algorithm is based on Fischer's Linear Discriminants. LDA training attempts to produce a linear transformations that emphasize differences between classes while reducing differences within classes. The goal is to form a subspace that is linearly separable between classes.

When used in the Face Identification and Evaluation System each human subject forms a class. LDA training requires training data that has multiple images per subject. LDA training is performed by first using PCA to reduce the dimensionality of the feature vectors. After this LDA is performed on the training data to further reduce the dimensionality in such a way that class distinguishing features are preserved. A final transformation matrix is produced by multiplying the PCA and LDA basis vectors to produce a full raw to LDA space transformation matrix.

The final output of the LDA training is the same as PCA. The algorithm produces a set of LDA basis vectors. These basis vectors produce a transformation of the feature vectors. Like the PCA algorithm, distance metrics can be used on the LDA feature vectors.

**Training** Like PCA, LDA training is performed by the `csuSubspaceTraining` executable. This algorithm is enabled 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 by [3]. A detailed description of the implementation can be found in[5]. After training the basis vectors are truncated to the number of classes used in training.

**Distance Metrics** The `csuSubspaceProject` code is also used to generate distance files for LDA. Please see the PCA distance metrics section for more information.

### 5.3 Bayesian Intrapersonal/Extrapolational Classifier (csuBayesian/BIC)

The third algorithm in the CSU distribution is based on an algorithm developed by Moghaddam and Pentland[7]. There are two variants of this algorithm, a *maximum a posteriori* (MAP) and *maximum likelihood* (ML) classifier. This algorithm is interesting for several reasons, including the fact that it examines the difference image between two photos as a basis for determining whether the two photos are of the same subject. Difference images which originate from two photos of different subjects are said to be *extrapolational* whereas images which originate from two photos of the same subject are said to be *Intrapersonal*.

The key assumption in Moghaddam and Pentland's work is that the particular difference images belonging to the intrapersonal and extrapolational difference images originate from distinct and localized Gaussian distributions within the space of all possible difference images.

The actual parameters for these distributions are not known, so the algorithm begins by extracting from the training data, using statistical methods, the parameters that define the Gaussian distributions corresponding to the intrapersonal and extrapolational difference images. This training stage, called density estimation, is performed through Principle Components Analysis (PCA). This stage estimates the statistical properties of two subspaces: one for difference images that belong to the intrapersonal class and another for difference images that belong to the extrapolational class. During the testing phase, the classifier takes each image of unknown class membership and uses the estimates of the the probability distributions as a means of identification.

**Training** In the current CSU implementation, the extrapolational and intrapersonal difference images for training are generated using the "csuMakeDiffs" program and subsequently the parameters of the two subspaces are estimated by running PCA ("csuSubspaceTrain"). This is done independently for the intrapersonal and extrapolational difference images. Unlike in our earlier distribution, the current distribution does *not* include a separate and independent program for training the Bayesian classifier.

**Distance Metrics** The "csuBayesianProject" code is used to generate distance files. It requires a list of images and two subspace training files (one for the extrapolational 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 similarity from each image to all other images. The similarities may be computed using the maximum a posteriori (MAP) or the 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.

### 5.4 Elastic Bunch Graph Matching (csuGabor)

A fourth algorithm is under development which is a variant of the Bochum/USC Elastic Bunch Graph Matching algorithm[8]. The algorithm locates specific landmark points on the face of an individual. The final metric is based on the convolution of these points with Gabor kernels, as well as the similarity of the graph that connects these points. As of version 4.0, this algorithm is not yet ready for release.

## 6 Analysis

### 6.1 CSU Rank Curve

The rank curve code is used to generate the standard FERET cumulative match curves. It takes a probe and gallery list. It computes the rank of the closest gallery image of the same subject for each probe image. A rank curve is then generated by summing the number for each rank. The rank curve data is written to an ASCII file with column headers and one line per point on the rank curve.

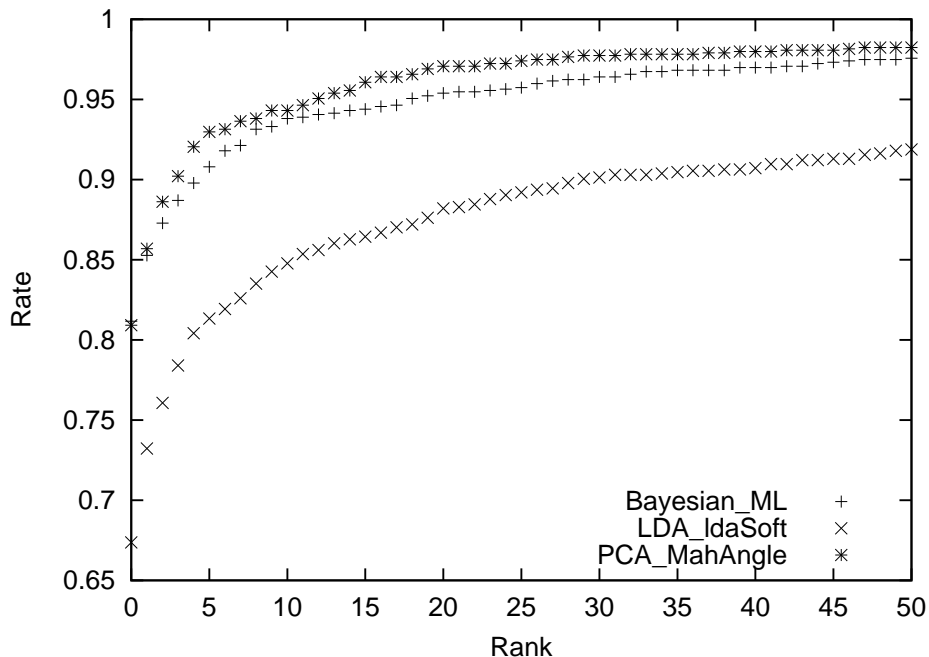


Figure 3: Cumulative Match Curve for FERET FA Probe Set.

The 4.0 distribution comes with scripts to generate cumulative match curve data for the four standard FERET probe sets. The curves themselves may be generated using a small Python program that in turn uses GNUplot to generate the figures. This program is in extras/plotRankCurves.py. Examples of these figures appear in Figure through .

The cumulative rank curves are similar to those shown in the original FERET evaluation[9]. However, they are not identical, nor would we expect them to be identical. There are many differences including new algorithm implementations, new image preprocessing code and perhaps most importantly different training image sets.

Keeping those caveats in mind, our script will pre-process the FERET imagery, train the algorithms, run the algorithms to generate distance matrices, and finally build cumulative match 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 wanted understand what was done in the FERET evaluation and wanting to adapt it to their own purposes.

## 6.2 CSU Permute

The CSU Permute code performs virtual experiments using the distance files. It does this by taking random permutations of the probe and gallery sets and then performs nearest neighbor classification. It then generates a sample probability distribution for recognition rate under the assumption that probe and gallery images are interchangeable for subjects. To read more about the methodology lying behind csuAnalyzePermute and how it may be used to characterize the performance of an algorithm see our paper from CVPR 2001[1]. An alternative means of computing error bars has been developed by Ross Micheals and Terry Boulton[10].

Figure 7 shows an example comparing the PCA and BIC algorithms on a set of 640 FERET images of 120 subjects. Observe that average performance of the PCA algorithm is higher than BIC, but that relative to standard error bars derived from the sample distributions, the difference does not appear significant relative to changes in the choice of probe and gallery images.

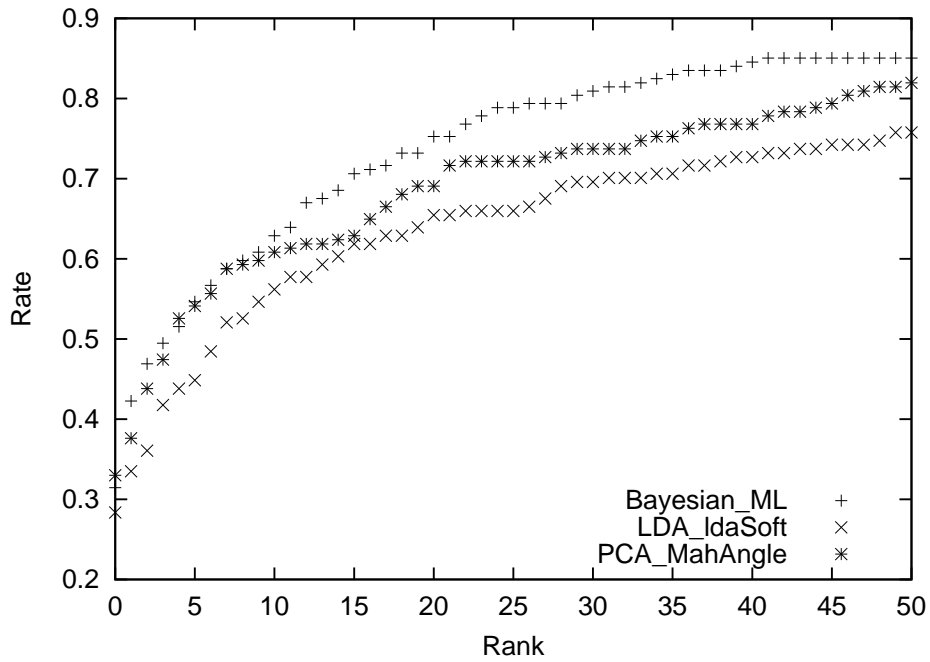


Figure 4: Cumulative Match Curve for FERET FC Probe Set.

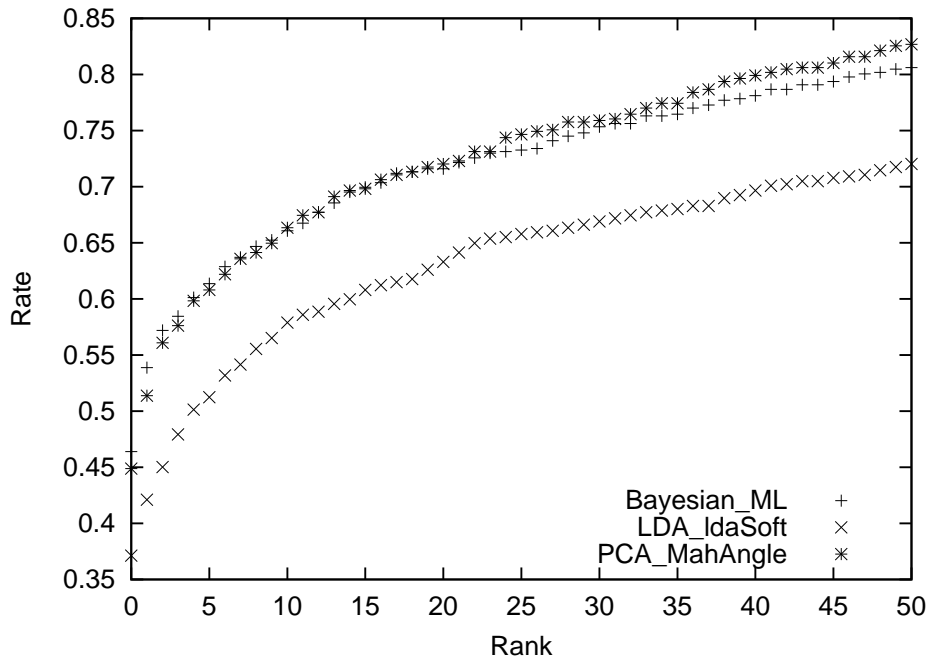


Figure 5: Cumulative Match Curve for FERET Dup 1 Probe Set.

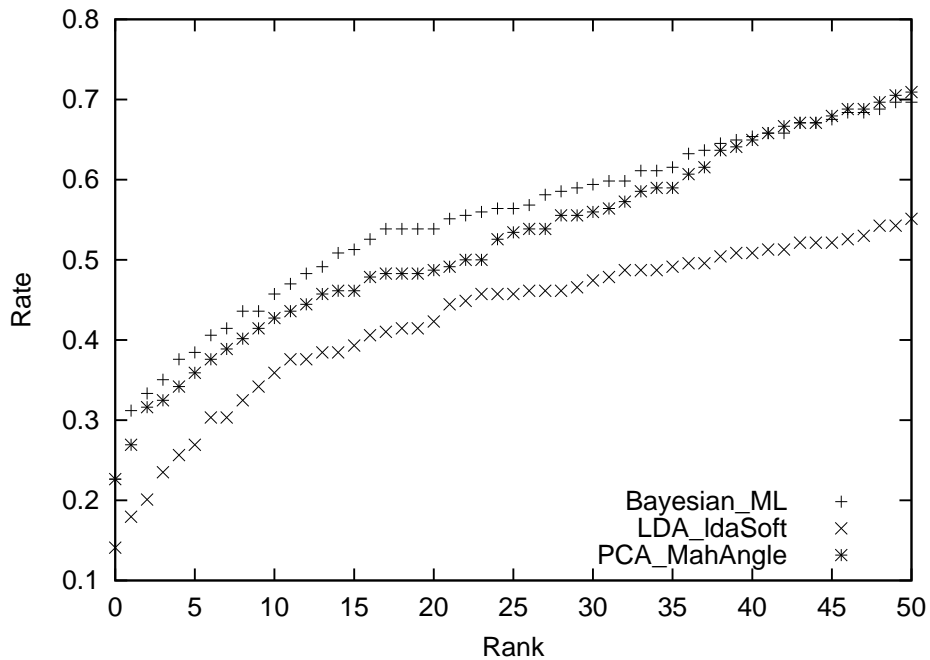


Figure 6: Cumulative Match Curve for FERET Dup 2 Probe Set.

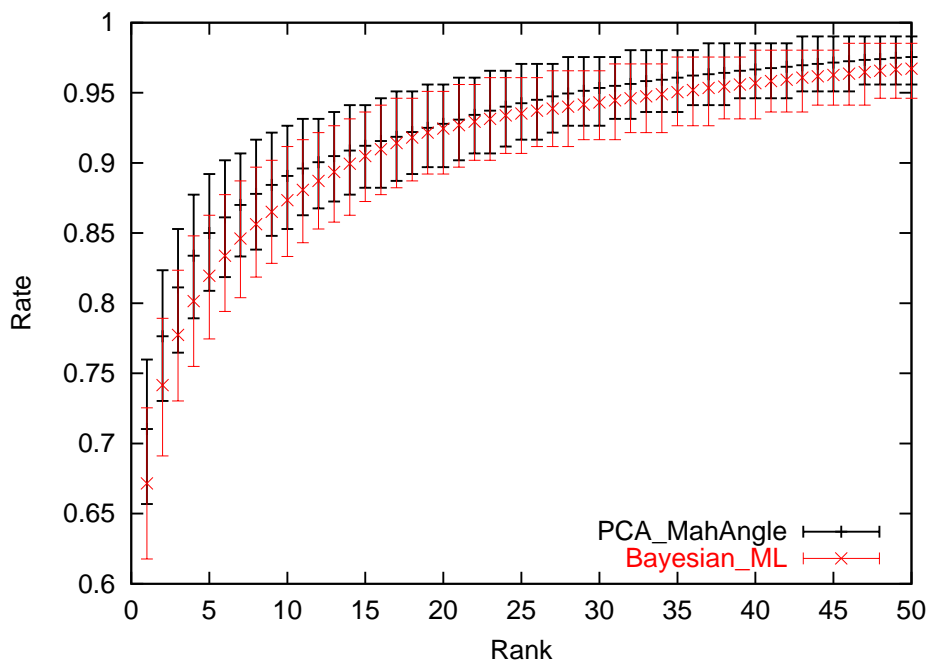


Figure 7: This figure compares our PCA and Bayesian algorithms. The rank curves include 95% error bars that were estimated by csuPermute. The error bars show that there is no significant difference between PCA and Maximum Likelihood when trained on the FERET training set.

## 7 Acknowledgments

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

## References

- [1] J. Ross Beveridge, Kai She, Bruce Draper, and Geof H. Givens. 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*, pages 535 – 542, December 2001.
- [2] Ross Beveridge. Evaluation of face recognition algorithms web site. <http://cs.colostate.edu/evalfacerec>.
- [3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, second edition edition, 2001.
- [4] FERET Database. <http://www.itl.nist.gov/iad/humanid/feret/>. NIST, 2001.
- [5] J. Ross Beveridge. The Geometry of LDA and PCA Classifiers Illustrated with 3D Examples. Technical Report CS-01-101, Computer Science, Colorado State University, 2001.
- [6] M. A. Turk and A. P. Pentland. Face Recognition Using Eigenfaces. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 586 – 591, June 1991.
- [7] B. Moghaddam, C. Nastar, and A. Pentland. A bayesian similarity measure for direct image matching. *ICPR*, B:350–358, 1996.
- [8] Kazunori Okada, Johannes Steffens, Thomas Maurer, Hai Hong, Egor Elagin, Hartmut Neven, and Christoph von der Malsburg. The Bochum/USC Face Recognition System And How it Fared in the FERET Phase III test. In H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman Soulié, and T. S. Huang, editors, *Face Recognition: From Theory to Applications*, pages 186–205. Springer-Verlag, 1998.
- [9] P.J. Phillips, H.J. Moon, S.A. Rizvi, and P.J. Rauss. The FERET Evaluation Methodology for Face-Recognition Algorithms. *T-PAMI*, 22(10):1090–1104, October 2000.
- [10] Ross J. Micheals and Terry Boulton. Efficient evaluation of classification and recognition systems. In *IEEE Computer Vision and Pattern Recognition 2001*, page (to appear), December 2001.
- [11] Wendy S. Yambor. Analysis of pca-based and fisher discriminant-based image recognition algorithms. Master’s thesis, Colorado State University, 2000.
- [12] W. Zhao, R. Chellappa, and A. Krishnaswamy. Discriminant analysis of principal components for face recognition. In *In Wechsler, Philips, Bruce, Fogelman-Soulie, and Huang, editors, Face Recognition: From Theory to Applications*, pages 73–85, 1998.