Facial Recognition

Using Empirical Mode Decomposition, Multi-Linear Principal Component Analysis and Post Processing using Expectation Maximization Algorithm

Supervised by: Mr. Md. Zahangir Alom

Conducted by

Mabrur Mujib Chowdhury

14341004

School of Engineering and Computer Science

BRAC University

Submitted on: September, 2014

DECLARATION

I hereby declare that this thesis is based on the results found by myself. I have gone through numerous reference papers from researchers of both national and foreign universities, and a full list is mentioned at the end of the paper. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of the Supervisor	Signature of the Author
_____	_____
Mr. Md. Zahangir Alom	Mabrur Mujib Chowdhury

CERTIFICATE

This is to certify that the thesis on "**Facial Recognition using EMD, Multi-linear PCA and Post-Processing using EM-Algorithm"** is a bona-fide record done by **Mabrur Mujib Chowdhury** for the fulfillment of the degree of Bachelor of Science in Computer Science (BSc. in CS) from **BRAC University.**

The thesis has been prepared under my guidance and is a record of bona-fide work carried out successfully.

Supervised by

Md. Zahangir Alom
Lecturer III
Department of Computer Science & Engineering
BRAC University
Dhaka, Bangladesh

ACKNOWLEDGEMENTS

Firstly, I would like to thank the Almighty, without whose blessings none of this would have come into fruition.

It is truly an honor for me to thank those who made my thesis possible.

I owe my deepest gratitude to my supervisor, Md. Zahangir Alom who has repeatedly pushed, educated and inspired me to develop a thorough understanding of the subject. He was there for me at every stumbling block I came across, and clarified all my misconceptions with his knowledge and wisdom, kept my morale intact and made me persevere towards completion of my thesis.

I would also like to thank my parents and my brother. Their undying support from start to finish kept me high-spirited even through times of difficulty, and allowed me to dig deep and fully concentrate on my task. Without their help, I would not have had the mental strength and determination to persistently strive towards achieving my goal.

ABSTRACT

Facial Recognition using Empirical Mode Decomposition, Multi-Linear PCA and Expectation Maximization Algorithm

The field of facial recognition is rapidly growing into a vital part of our everyday lives. The use of facial recognition systems has been extended primarily from security purposes to social networking sites, managing fraud, and improved user experience. Numerous algorithms have been designed to perform facial recognition with greatest accuracy. The use of several preprocessing and post-processing techniques is also known to improve the effectiveness of these recognition algorithms. This paper focuses on a three-tier approach towards facial recognition.

A widely popular recognition algorithm used today is the Principal Component Analysis (PCA). Throughout the years, there have been many improvements and extensions to the original PCA. One such extension is the Multi-linear PCA, which is the algorithm I have used in my study. Studies have shown that results of the recognition algorithm can be greatly improved by applying preprocessing techniques to the images before feeding them into the main recognition algorithm. Therefore, in addition to the Multi-linear PCA, I will be using Empirical Mode Decomposition (EMD) for preprocessing. Furthermore, I plan to run an Expectation Maximization (EM) algorithm which estimates Maximum Likelihood values for information which may be missing from the dataset.

Applying these three strategies simultaneously would allow us to have a more efficient, secure and robust facial recognition system.

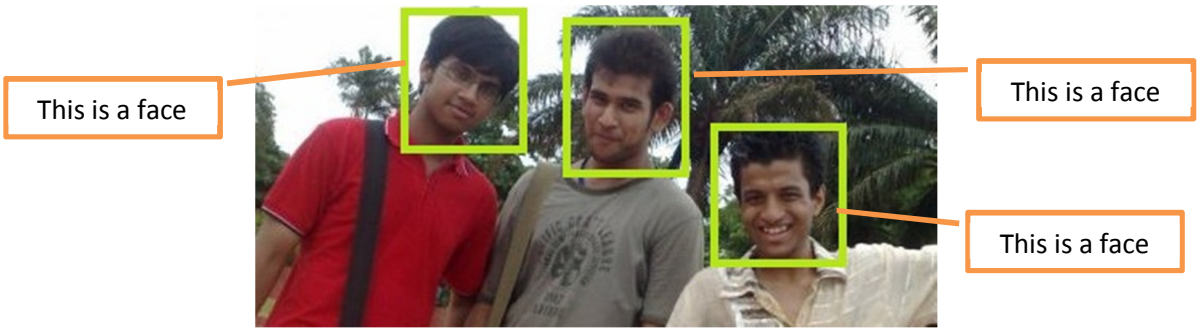
TABLE OF CONTENTS

CHAPTER ONE

Introduction

In order to design a facial recognition system, we require an image as an input to our program. For a face to be detected, the very first thing to verify in the image is to check if it actually contains a face(s). If the image has no face, then there is nothing to recognize, and using facial recognition software will be meaningless. So the first step is to verify that the presence of at least one face in the entire input image. A facial recognition system need not necessarily be able to detect the presence of a face in an image, but typically when we speak of Facial Recognition, the facial detection part is a given.

In my study, I have not focused on Face Detection and my work will focus strictly on 'facial recognition'. I have decided to omit the face detection portion because of its fairly trivial method of implementation. Face Detection is commonly employed using algorithms such as Principal Component Analysis (or extensions thereof), skin color segmentation etc.



Face Detection

The image above shows how a face detector works. It simply identifies faces from an input image. Here, we can see that three faces have been detected, outlines in the green boxes. This feature is readily available in most modern phones and digital cameras.

Facial Recognition

This image roughly shows the type of output we can expect from a Facial Recognition system, provided the faces have been stored in the database and identified before. This manages to identify the faces and based on previously saved information, it recognizes the faces and tells the user who each of the faces belong to.

1.1 Background of the Problem

Throughout the years, slowly everything has become computerized. This is a well-known fact. But technology is not always one hundred percent guaranteed to be foolproof. Still errors occur, and machines are prone to error and discrepancy. Hence, scientists and programmers are constantly developing new and improved algorithms to improve success rates to get as close to maximum efficiency as possible, given constraints of time (computational complexity) and space (efficient utilization of computer memory).

1.2 Motivation

The reasons for conducting the research work were:

(1) I have completed a project on Face Detection while doing my Artificial Intelligence course as part of the University curriculum for obtaining a Bachelor's degree in Computer Science. Successfully completing that project has inspired me to further pursue the field of image processing, and taking one step forward and focus on Facial Recognition

(2) BRAC University could provide me with the facilities and services that I required to undertake this thesis successfully. From the teachers to the technical officers at the laboratories, along with the rich amount of resources I was able to find in the university library, I found everything I needed to accomplish my purpose in completing this research-based thesis.

(3) The computer environment and resources required to complete this thesis were readily available for little or no cost online, which added as further motivation for me to carry on with the thesis.

All three reasons argue most powerfully for the type of research work I have conducted.

1.3 Methodology

Facial recognition is merely one-third of my entire thesis topic. This is because of the preprocessing and post-processing methods used along with the recognition algorithm itself. The entire working methodology can be summarized in a flowchart in the following page, along with a small description of the total process, in brief. The flowchart is fairly simplistic, without getting into any of the technical details, which will be introduced in the coming chapters.

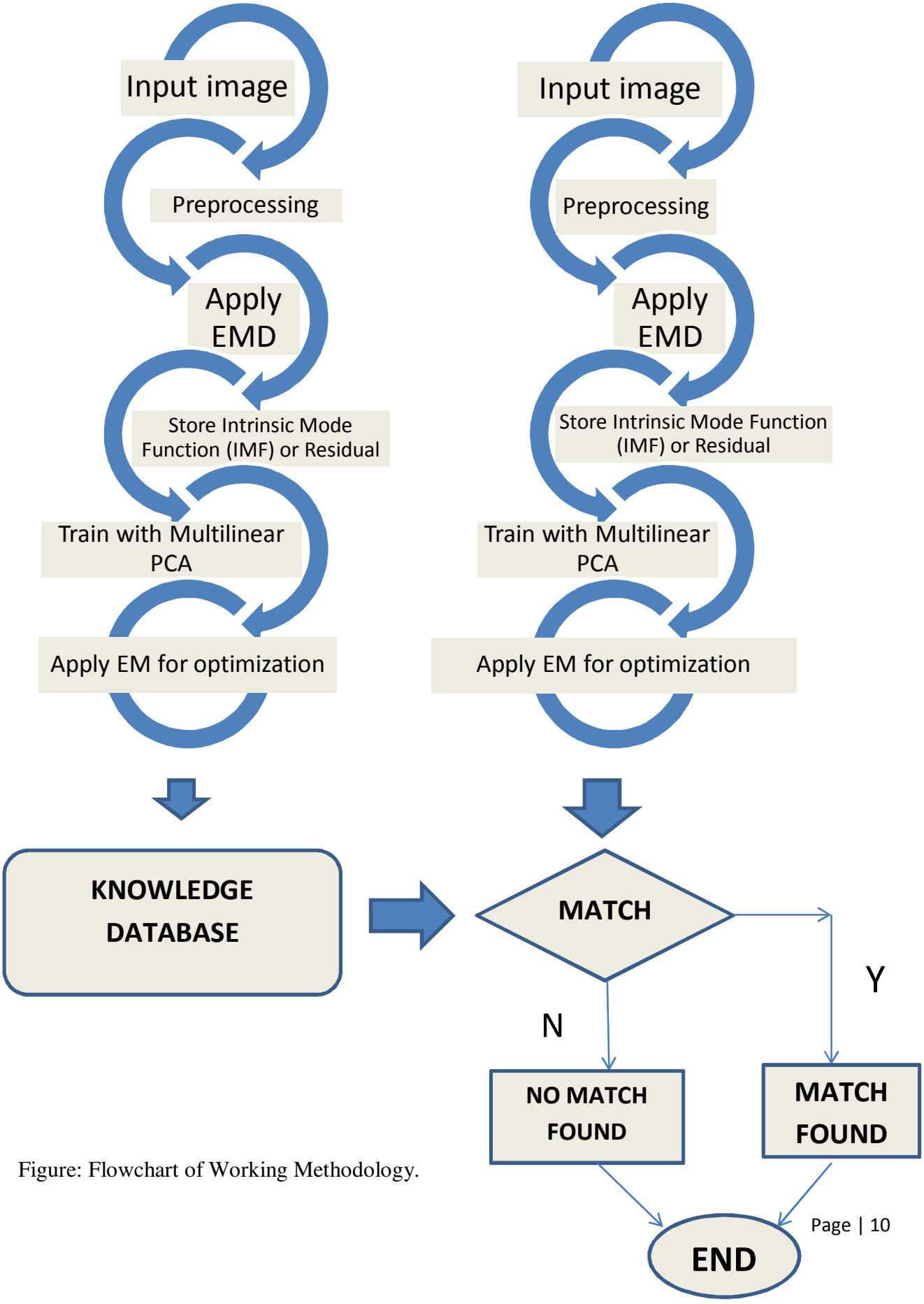


Figure: Flowchart of Working Methodology.

Faces can only be recognized if the face already exists in the database, i.e. the face has been detected before. The column on the left shows the usage of an image which is not currently stored in the database, and is therefore for all intents and purposes, a new image. Feeding that image into the algorithm allows it to be processed. EMD is applied on the image first to improve the sharpness of the image. Then the image is analyzed by the Multi-linear PCA. Applying post-processing techniques after that puts the image into the knowledge database in such a format that it is computationally easier to compare with images that come in later.

Once a second image in put into the algorithm, it goes through the same sequence of processes as the first image, and after it goes through the three phases, the image is compared to all images in the database. The signals are compared, and it reports whether the image is a match with any of the images already in the database, or not a match.

1.4 Outline of this paper

This report is organized into seven chapters. In Chapter 2, I have mentioned the names and brief descriptions of some of the more important reference papers that have benefitted me greatly in completing my thesis. I have gone through a large number of papers (a full list is available at the end of the report), but the papers mentioned in that chapter are of particular importance as they have directly assisted me in my works. In Chapter 3, I discuss the proposed approach towards facial recognition, and the three procedures in question are introduced. This is accompanied by how I intend to fully utilize each process. Chapter 4 takes an in-depth look at the three processes, and describes methods, functions, and each and every step of applying EMD, MPCA and EM into my working respectively. The results of applying these processes are discussed in Chapter 5, along with the software environment in which all the programs were run. Conclusions are drawn in Chapter 6 about the effectiveness of the entire procedure. Future works and improvements are discussed in the final chapter.

CHAPTER-2

Literature Review

Facial recognition is a prominently developed field of computer science, and therefore it is no surprise that I found myself flooded by academic papers regarding the subject. I have gone through numerous of such papers and projects, of which I found many to be directly relevant to my particular area of interest.

Relevant works and papers I have found and used as reference come from various universities and institutions from all over the world. A fully compiled list of references is available at the end of the paper. However, I have handpicked a selective few of those papers to do a review on. These papers in particular taught me the basics and implementation of some of the techniques I have employed into my thesis work.

2.1 Multi-linear Principal Component Analysis for Face Recognition with Fewer Features

by Wanga, Barreto, Wang, Chen, Rishe, Andrian, Adjouadi (*Florida International University)* – In this study, a method is proposed based on multi-linear principal component analysis (MPCA) for face recognition. This method utilized less features than traditional MPCA algorithm without downgrading the performance in recognition accuracy. The experiment results show that the proposed method is more suitable for large dataset, obtaining better computational efficiency. Moreover, when support vector machine is employed as the classification method, the superiority of the proposed algorithm reflects significantly.

2.2 Analyzing Facial Images Using Empirical Mode Decomposition for Illumination Artifact Removal and Improved Face Recognition

by Bhagavatula, Savvides *(Carnegie Mellon University)* – A popular modality of biometrics, facial recognition is effective when used in controlled environments as in those situations where factors such as

camera position, facial expression, and illumination effects are either completely or partially controlled in a beneficial way. Regulation of such factors has an immediate effect on the performance of facial recognition algorithms, in particular illumination effects which cannot be controlled by even the most cooperative of users. This paper describes a method to address illumination effects in the biometric modality of face recognition using the signal processing analysis tool of Empirical Mode Decomposition (EMD) to decompose images into their intrinsic mode function that corresponds to the dominant illumination factors. Using these illumination modes, the facial image is reconstructed without the illumination distortion components to synthesize a more illumination neutral facial image. Then, verification experiments are performed using algorithms such as PCA, FLDA and ACF to demonstrate the fundamental effectiveness of EMD as an illumination compensation method. Results are reported on the CMU PIE database.

2.3 Face Recognition with the Mixture of MDA and MPCA by Shams-Baboli, Kaffashpour-Yazdi, Shams-Baboli, Araghi *(Journal of Basic and Applied Scientific Research)* – Many algorithms had been proposed for face recognition problems in the last few years, but none of them could defeat the enormous changeability of some environmental parameters such as: lighting, scale and pose; and the SSS problem at the same time. This paper proposes a method for improving the robustness of a face recognition algorithm with tensors representation and fusion of the MDA and MPCA. A multi-linear principal component analysis (MPCA) for tensor object feature extraction and a multi-linear discriminant analysis (MDA), to find the best subspaces have been proposed. It should be noted that both of them work with tensor objects so the structure of the objects has been never broken. Therefore a better performance is achieved. For the final decision, these two criteria are combined according to a given combination rule. Two algorithms are proposed for the fusion phase: the K-Nearest Neighbors and the Nearest Mean. Finally, a comprehensive experiment has been provided to demonstrate that this fusion on tensor base algorithm has the potential to outperform the traditional vector based subspace learning methods, especially in the cases with small sample sizes and curse of dimensionality and variability of many environmental parameters such as lighting, pose and scale. Experimental

results approve that fusing of these two tensor based approaches gives better recognition accuracy.

2.4 MPCA: Multi-linear Principal Component Analysis of Tensor Objects by Lu, Konstantinos, Venetsanopoulos *(IEEE Transactions on Neural Networks)* - This paper introduces a multi-linear principal component analysis (MPCA) framework for tensor object feature extraction. Objects of interest in many computer vision and pattern recognition applications, such as 2-D/3-D images and video sequences are naturally described as tensors or multi-linear arrays. The proposed framework performs feature extraction by determining a multi-linear projection that captures most of the original tensor input variation. The solution is iterative in nature and it proceeds by decomposing the original problem to a series of multiple projection sub-problems. As part of this work, methods for subspace dimensionality determination are proposed and analyzed. It is shown that the MPCA framework discussed in this work supplants existing heterogeneous solutions such as the classical principal component analysis (PCA) and its 2-D variant (2-D PCA). Finally, a tensor object recognition system is proposed with the introduction of a discriminative tensor feature selection mechanism and a novel classification strategy, and applied to the problem of gait recognition. Results presented here indicate MPCA's utility as a feature extraction tool. It is shown that even without a fully optimized design; an MPCA-based gait recognition module achieves highly competitive performance and compares favorably to the state-of-the-art gait recognizers.

2.5 EM Algorithms for PCA and SPCA by Roweis (*California Institute of Technology)* - This paper presents an expectation-maximization (EM) algorithm for principal component analysis (PCA). The algorithm allows a few eigenvectors and eigenvalues to be extracted from large collections of high dimensional data. It is computationally very efficient in space and time. It also naturally accommodates missing information. It reports results on synthetic and real data showing that the EM algorithms correctly and efficiently find the leading eigenvectors of the covariance of datasets in a few iterations using up to hundreds of thousands of data points in thousands of dimensions.

2.6 PEM-PCA: A Parallel Expectation-Maximization PCA Face Recognition Architecture by Rujirakul, So-In, Arnonkijpanich *(The Scientific World Journal)* - Principal component analysis or PCA has been traditionally used as one of the feature extraction techniques in face recognition systems yielding high accuracy when requiring a small number of features. However, the covariance matrix and eigenvalue decomposition stages cause high computational complexity, especially for a large database. Thus, this research presents an alternative approach utilizing an Expectation-Maximization algorithm to reduce the determinant matrix manipulation resulting in the reduction of the stages' complexity. To improve the computational time, a novel parallel architecture was employed to utilize the benefits of parallelization of matrix computation during feature extraction and classification stages including parallel preprocessing, and their combinations, so-called a Parallel Expectation Maximization PCA architecture. Comparing to a traditional PCA and its derivatives, the results indicate lower complexity with an insignificant difference in recognition precision leading to high speed face recognition systems, that is, the speed-up over nine and three times over PCA and Parallel PCA.

2.7 Principal Component Analysis with Noise and/or Missing Data by Bailey *(Lawrence Berkeley National Lab)* - The paper presents a method for performing Principal Component Analysis (PCA) on noisy datasets with missing values. Estimates of the measurement error are used to weight the input data such that compared to classic PCA; the resulting eigenvectors are more sensitive to the true underlying signal variations rather than being pulled by heteroskedastic measurement noise. Missing data is simply the limiting case of weight=0. The underlying algorithm is a noise weighted Expectation Maximization (EM) PCA, which has additional benefits of implementation speed and flexibility for smoothing eigenvectors to reduce the noise contribution. We present applications of this method on simulated data and QSO spectra from the Sloan Digital Sky Survey.

There are many other publications, online journals and research papers that I have referred to during the course of the study. The bibliography along with a full list of references can be found at the end of the paper.

CHAPTER 3

PROPOSED SYSTEM IMPLEMENTATION

3.1 Pre-processing using Empirical Mode Decomposition

Empirical Mode Decomposition (EMD) is a fundamental part of the Hilbert-Huang Transform. It is a signal processing analysis tool that addresses the effects of illumination in the biometric modality of facial recognition. Images are converted, or decomposed into their respective Intrinsic Mode Functions (IMFs) which correspond to the dominant illumination factors of the image. Using their illumination modes, the facial image can be reconstructed without the illumination distortion components. Hence, using EMD as a preprocessing tool allows a more illumination-neutral facial image to be synthesized.

The Empirical Mode Decomposition (EMD) can adaptively decompose a complex signal into IMFs that are relevant to intrinsic physical significances, therefore is a powerful tool for multi-scale analysis of non-stationary signals. Towards restoring a frontal-illuminated face from a single image, in this paper we study the usage of EMD for manipulating the illumination issue on face images. We propose an EMD-based algorithm to extract the illumination-insensitive facial features. We also come up with an EMD-based scheme to detect the shadows and to reduce the effects of shadows on face images. By preserving the intrinsic facial features as well as lessening the shadows, it is more likely to restore the frontal-illuminated face image with good visual quality from a single image. Experiments verify the effectiveness of the proposed methods.

3.1.1 Image Acquisition

Image Acquisition Toolbox of MATLAB provides an app and functions and a programmatic interface to help us work with image acquisition hardware. We can automate repetitive tasks, create workflows combined with tasks such as image processing, and create standalone executables that acquire images and video with MATLAB Compiler. The toolbox enables us to customize the acquisition process to include integrating image processing functionality to identify objects, enhance imagery, or construct mosaics and panoramic views as the data is acquired.

Image Acquisition Toolbox automatically detects compatible image and video acquisition devices. Each device connection is encapsulated as an object, providing an interface for configuration and acquisition. We can create multiple connection objects for simultaneous acquisition from as many devices as our PC and imaging hardware support. Image Acquisition Toolbox can be used on Windows®, Linux®, and Macintosh® systems, enabling us to reuse code when connecting to the same camera in different operating systems.

Image Acquisition Toolbox supports several modes, including background acquisition and continuous acquisition, while processing the acquired data. The toolbox automatically buffers data into memory, handles memory and buffer management, and enables acquisition from an ROI. The image acquisition engine is designed to acquire imagery as fast as our camera and computer can support, enabling analysis and processing of high-speed imaging applications.

Data can be acquired in a wide range of data types, including signed or unsigned 8-, 16-, and 32-bit integers and single- or double-precision floating point. The toolbox supports any color space provided by the image acquisition device including RGB, YUV, or grayscale. Raw sensor data in a Bayer pattern can be automatically converted into RGB data.

For the purpose of this thesis, no cameras are used, and images are input manually. Of course, instantaneous facial recognition using cameras may also be achieved, but require some further programming.

3.1.2 Data Acquisition and Representation using EMD

Empirical Mode Decomposition involves a range of mathematical functions used in MATLAB, the main computation process being the Sifting process and Spline Interpolation of signals. A detailed and more canonical form of the Empirical Mode Decomposition process is expressed in the Methodology section of the paper. To summarize the entire procedure in simple terms, Empirical Mode Decomposition can be expressed in the following few steps:

Step 1: Read the image, turn it into source signal and convert it to grayscale

Firstly, we take our image and pass it through the *imread()* function of MATLAB. This allows the image to be read, in the form of a matrix. Then we need to convert it into a grayscale image by the use of a simple MATLAB function. A grayscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white (different from a Binary image), are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.

In MATLAB `I = rgb2gray(RGB)` converts the true-color image `RGB` to the grayscale intensity image I. `rgb2gray` converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

`newmap = rgb2gray(map)` returns a grayscale colormap equivalent to map.

`gpuarrayB = rgb2gray(gpuarrayA)` performs the operation on a GPU. `gpuarrayA` can be either an RGB image or a colormap. `gpuarrayB` is a GPU-Array of the same type as `gpuarrayA`. This syntax requires the Parallel Computing Toolbox of MATLAB.

The purpose of converting the image to grayscale is to eliminate the effect of color (hue and saturation), which lead to greater processing times and more complex computations.

The image signal must then be converted into a source signal. In our case, the source signal is a column vector. Taking the image in will firstly give us a row vector with *n* number of rows for an *n*m* dimension picture. We need to convert the row vector to a column vector. It is a simple process that can be done by adding the suffix *(:)* at the end of the signal. This guarantees we have the image signal converted to a column vector, and is ready to be passed on for further processing.

Step 2: Set the number of IMFs to be found

The objective of performing Empirical Mode Decomposition on the input image is to find out the corresponding Intrinsic Mode Functions (or IMFs) of the image. Ideally, the maximum number of IMFs should be found for each input signal, but limiting the number of IMFs to be found will reduce computational complexities of time and space.

Step 3: Find local extremes of source signal compute envelopes and extract detail

By computing the derivatives of the input signal, we can find the resultant extreme turning points, both maxima and minima. From these points, a mean envelope can be calculated by averaging the maxima with their corresponding minima. If we find that the mean envelop value is equal to zero, this means we may have found an IMF. The next stage involves the extraction of detail by using the mean envelope. The detail can be extracted by subtracting the original signal from the mean envelope. The difference will give us the special features of the original signal, if any.

Step 4: Compare mean envelope with standard deviation

The absolute value of the mean envelope is now compared with the standard deviation of the original signal. If the mean envelope is found to be greater than the standard deviation that

means that the computed envelope is significantly different and must contain special features in the image. Therefore, we must go back to the previous step by replacing the original input signal with the value of the mean envelope. If however the mean envelope is found to be less than the standard deviation, then we can conclude that the detail extracted from the previous step comprises of information that is not useful to our purpose. Therefore, the detail can be classified as an Intrinsic Mode Function.

Step 5: Computation of Residuals and Stopping Condition

This step is reached once we have an Intrinsic Mode Function. The residual information is computed in this step by subtracting the detail signal obtained (the Intrinsic Mode Function that we have found in the previous step) from the original signal it was computed from. Using this residual as the input signal, we go back to step 3 and repeat the process. The process is repeated until the input signal no longer has any extreme points, or until the pre-designated number of IMFs has been computed.

An IMF satisfies two conditions. First, the number of extreme points and the number of zero-crossings must be equal or at most differ by one. Second, at any point, the mean values of the envelopes defined by the local maxima and the local minima respectively must equal zero. Step 3 is collectively referred to as the "sifting process" and is the most computationally expensive and error prone portion of the algorithm due to identification of extreme points and subsequent interpolation.

The primary power of EMD is that once these IMFs have been found, we can easily go back and forth from them to the original data. By simply summing all the IMFs together we will recover the original data accommodating for minor variations due to the interpolation present in the algorithm. Empirical Mode Decomposition also allows us to selectively reconstruct the data, ignoring the IMFs whose contributions to the data are undesirable. For our application, such contributions are those of illumination effects.

If we can use EMD to decompose our original facial images into their IMFs, there is a strong likelihood that the effects of illumination will be isolated to one or more IMFs. Selective reconstruction of facial images using IMFs that do not contain illumination effects will enable us to reconstruct the fundamental nature of the data without the unwanted effects of illumination variation.

3.2 Multi-Linear Principal Component Analysis for Face Recognition

3.2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a widely used dimensionality reduction technique in data analysis. Its popularity comes from three important properties. First, it is the *optimal* (in terms of mean squared error) *linear* scheme for compressing a set of high dimensional vectors into a set of lower dimensional vectors and then reconstructing. Second, the model parameters can be computed *directly* from the data – for example by diagonalizing the sample covariance. Third, compression and decompression are easy operations to perform given the model parameters – they require only matrix multiplications.

Despite these attractive features however, PCA models have several shortcomings. One is that naive methods for finding the principal component directions have trouble with high dimensional data or large numbers of datapoints. Consider attempting to diagonalize the sample covariance matrix of n_ vectors in a space of p dimensions when n and p are several hundred or several thousand. Difficulties can arise both in the form of computational complexity and also data scarcity. Even computing the sample covariance itself is very costly, requiring an n-squared number of operations. In general it is best to avoid altogether computing the sample covariance explicitly. Methods such as the *snap-shot* algorithm do this by assuming that the eigenvectors being searched for are linear combinations of the datapoints; their complexity is n-cubed. In this note, I present a version of the expectation-maximization (EM) algorithm for learning the

principal components of a dataset. The algorithm does not require computing the sample covariance and has a complexity limited by (O)knp operations where k is the number of leading eigenvectors to be learned.

Another shortcoming of standard approaches to PCA is that it is not obvious how to deal properly with missing data. Most of the methods discussed above cannot accommodate missing values and so incomplete points must either be discarded or completed using a variety of ad-hoc interpolation methods. On the other hand, the EM algorithm for PCA enjoys all the benefits of other EM algorithms in terms of estimating the maximum likelihood values for missing information directly in every iteration.

Finally, the PCA model itself suffers from a critical flaw which is independent of the technique used to compute its parameters: it does not define a proper probability model in the space of inputs. This is because the density is not normalized within the principal subspace. In other words, if we perform PCA on some data and then ask how well *new* data are fit by the model, the only criterion used is the squared distance of the new data from their projections into the principal subspace. A datapoint far away from the training data but nonetheless near the principal subspace will be assigned a high "pseudo-likelihood" or low error. Similarly, it is not possible to generate "fantasy" data from a PCA model.

3.2.2 Multi-Linear Principal Component Analysis (MPCA)

Multi-linear PCA is a mathematical procedure that uses multiple orthogonal transformations to convert a set of multidimensional objects into another set of multidimensional objects of lower dimensions. There is one orthogonal (linear) transformation for each dimension (mode): hence *multi-linear*. This transformation aims to capture as high a variance as possible, accounting for as much of the variability in the data as possible, subject to the constraint of mode-wise orthogonality.

MPCA is a multi-linear extension of principal component analysis (PCA). The major difference is that PCA needs to reshape a multidimensional object into a vector, while MPCA operates

directly on multidimensional objects through mode-wise processing. E.g., for 100x100 images, PCA operates on vectors of 10000x1 while MPCA operates on vectors of 100x1 in two modes. For the same amount of dimension reduction, PCA needs to estimate 49 * (10000 / (100 x 2) -1) times more parameters than MPCA in this example. Thus, MPCA is more efficient and better conditioned in practice.

3.2.3 The MPCA Algorithm

MPCA is a basic algorithm for dimension reduction via multi-linear subspace learning. In wider scope, it belongs to tensor-based computation. Its origin can be traced back to the Tucker decomposition in 1960s and it is closely related to higher-order singular value decomposition, (HOSVD) and to the best rank - (R1, R2…RN) approximation of higher-order tensors. The algorithm is summarized in the pseudocode below.

Input: A set of tensor samples $\{\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m = 1, \ldots, M\}$.

Output: Low-dimensional representations $\{\mathcal{Y}_m \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}, m = 1, \ldots, M\}$ of the input tensor samples with maximum variation captured.

Algorithm:

Step 1 (Preprocessing): Center the input samples as $\{\tilde{\mathcal{X}}_m = \mathcal{X}_m - \bar{\mathcal{X}}, m = 1, \ldots, M\}$, where $\bar{\mathcal{X}} = \frac{1}{M}\sum_{m=1}^{M} \mathcal{X}_m$ is the sample mean.

Step 2 (Initialization): Calculate the eigen-decomposition of $\mathbf{\Phi}^{(n)*} = \sum_{m=1}^{M} \tilde{\mathbf{X}}_{m(n)} \cdot \tilde{\mathbf{X}}_{m(n)}^T$ and set $\tilde{\mathbf{U}}^{(n)}$ to consist of the eigenvectors corresponding to the most significant P_n eigenvalues, for $n = 1, \ldots, N$.

Step 3 (Local optimization):

- Calculate $\{\tilde{\mathcal{Y}}_m = \tilde{\mathcal{X}}_m \times_1 \tilde{\mathbf{U}}^{(1)^T} \times_2 \tilde{\mathbf{U}}^{(2)^T} \ldots \times_N \tilde{\mathbf{U}}^{(N)^T}, m = 1, \ldots, M\}$.
- Calculate $\Psi_{\mathcal{Y}_0} = \sum_{m=1}^{M} \| \tilde{\mathcal{Y}}_m \|_F^2$ (the mean $\bar{\mathcal{Y}}$ is all zero since $\tilde{\mathcal{X}}_m$ is centered).
- For $k = 1 : K$
  - For $n = 1 : N$
    - Set the matrix $\tilde{\mathbf{U}}^{(n)}$ to consist of the P_n eigenvectors of the matrix $\mathbf{\Phi}^{(n)}$, as defined in (5), corresponding to the largest P_n eigenvalues.
  - Calculate $\{\tilde{\mathcal{Y}}_m, m = 1, \ldots, M\}$ and $\Psi_{\mathcal{Y}_k}$.
  - If $\Psi_{\mathcal{Y}_k} - \Psi_{\mathcal{Y}_{k-1}} < \eta$, break and go to Step 4.

Step 4 (Projection): The feature tensor after projection is obtained as $\{\mathcal{Y}_m = \mathcal{X}_m \times_1 \tilde{\mathbf{U}}^{(1)^T} \times_2 \tilde{\mathbf{U}}^{(2)^T} \ldots \times_N \tilde{\mathbf{U}}^{(N)^T}, m = 1, \ldots, M\}$.

Fig. 3. Pseudocode implementation of the proposed MPCA algorithm.

MPCA performs feature extraction by determining a multi-linear projection that captures most of the original tensorial input variations. As in PCA, MPCA works on centered data. The MPCA solution follows the alternating least square (ALS) approach. Thus, is iterative in nature and it proceeds by decomposing the original problem to a series of multiple projection sub-problems. Each sub-problem is a classical PCA problem, which can be easily solved.

It should be noted that while PCA with orthogonal transformations produces uncorrelated features/variables, this is not the case for MPCA. Due to the nature of tensor-to-tensor transformation, MPCA features are not uncorrelated in general although the transformation in each mode is orthogonal.

3.3 Expectation Maximization (EM) for performance enhancement

An **expectation–maximization** (**EM**) **algorithm** is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the *E* step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

3.3.1 EM Algorithm

The EM algorithm is used to find the maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent

variables in addition to unknown parameters and known data observations. That is, either there are missing values among the data, or the model can be formulated more simply by assuming the existence of additional unobserved data points. For example, a mixture model can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component that each data point belongs to.

Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values — viz. the parameters and the latent variables — and simultaneously solving the resulting equations. In statistical models with latent variables, this usually is not possible. Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice-versa, but substituting one set of equations into the other produces an unsolvable equation.

The EM algorithm proceeds from the observation that the following is a way to solve these two sets of equations numerically. One can simply pick arbitrary values for one of the two sets of unknowns, use them to estimate the second set, then use these new values to find a better estimate of the first set, and then keep alternating between the two until the resulting values both converge to fixed points. It's not obvious that this will work at all, but in fact it can be proven that in this particular context it does, and that the derivative of the likelihood is (arbitrarily close to) zero at that point, which in turn means that the point is either a maximum or a saddle point. In general there may be multiple maxima, and there is no guarantee that the global maximum will be found. Some likelihood estimates also have singularities in them, i.e. nonsensical maxima. For example, one of the "solutions" that may be found by EM in a mixture model involves setting one of the components to have zero variance and the mean parameter for the same component to be equal to one of the data points.

Given a statistical model consisting of a set $\mathbf{X}$ of observed data, a set of unobserved latent data or missing values $\mathbf{Z}$, and a vector of unknown parameters $\boldsymbol{\theta}$, along with a likelihood

function $L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z}) = p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data

$$L(\boldsymbol{\theta}; \mathbf{X}) = p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

However, this quantity is often intractable (e.g. if $\mathbf{Z}$ is a sequence of events, so that the number of values grows exponentially with the sequence length, making the exact calculation of the sum extremely difficult).

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps:

3.3.2 Expectation Step

Step 1:

Expectation step (E step): Calculate the expected value of the log likelihood function, with respect to the conditional distribution of $\mathbf{Z}$ given $\mathbf{X}$ under the current estimate of the parameters $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}}\left[\log L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z})\right]$$

3.3.3 Maximization Step

Step 2:

Maximization step (M step): Find the parameter that maximizes this quantity:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$$

Note that in typical models to which EM is applied:

1. The observed data points $\mathbf{X}$ may be discrete (taking values in a finite or countably infinite set) or continuous (taking values in an uncountably infinite set). There may in fact be a vector of observations associated with each data point.
2. The missing values (aka latent variables) $\mathbf{Z}$ are discrete, drawn from a fixed number of values, and there is one latent variable per observed data point.
3. The parameters are continuous, and are of two kinds: Parameters that are associated with all datapoints and parameters associated with a particular value of a latent variable (i.e. associated with all data points whose corresponding latent variable has a particular value).

However, it is possible to apply EM to other sorts of models.

The motivation is as follows. If we know the value of the parameters $\boldsymbol{\theta}$, we can usually find the value of the latent variables $\mathbf{Z}$ by maximizing the log-likelihood over all possible values of $\mathbf{Z}$, either simply by iterating over $\mathbf{Z}$ or through an algorithm such as the Viterbi algorithm for hidden Markov models. Conversely, if we know the value of the latent variables $\mathbf{Z}$, we can find an estimate of the parameters $\boldsymbol{\theta}$ fairly easily, typically by simply grouping the observed data points according to the value of the associated latent variable and averaging the values, or some function of the values, of the points in each group. This suggests an iterative algorithm, in the case where both $\boldsymbol{\theta}$ and $\mathbf{Z}$ are unknown:

1. First, initialize the parameters $\boldsymbol{\theta}$ to some random values.
2. Compute the best value for $\mathbf{Z}$ given these parameter values.
3. Then, use the just-computed values of $\mathbf{Z}$ to compute a better estimate for the parameters $\boldsymbol{\theta}$. Parameters associated with a particular value of $\mathbf{Z}$ will use only those data points whose associated latent variable has that value.
4. Iterate steps 2 and 3 until convergence.

The algorithm as just described monotonically approaches a local minimum of the cost function, and is commonly called *hard EM*. The *k*-means algorithm is an example of this class of algorithms.

However, we can do somewhat better by, rather than making a hard choice for $\mathbf{Z}$ given the current parameter values and averaging only over the set of data points associated with a particular value of $\mathbf{Z}$ instead determining the probability of each possible value of $\mathbf{Z}$ for each data point, and then using the probabilities associated with a particular value of $\mathbf{Z}$ to compute a weighted average over the entire set of data points. The resulting algorithm is commonly called *soft EM*, and is the type of algorithm normally associated with EM. The counts used to compute these weighted averages are called *soft counts* (as opposed to the *hard counts* used in a hard-EM-type algorithm such as *k*-means). The probabilities computed for $\mathbf{Z}$ are posterior probabilities and are what is computed in the E step. The soft counts used to compute new parameter values are what is computed in the M step.

CHAPTER 4

WORKING PROCESS

4.1 Empirical Mode Decomposition

The EMD was first introduced by Huang et al. as a tool to adaptively decompose a signal into a collection of Intrinsic Mode Functions (IMFs). It relies on a fully data-driven mechanism that does not require a-priori known basis. Therefore, EMD is especially well suited for analyzing non-linear and non-stationary signals like biometric signals. A function is defined as an IMF if the number of its extreme equals the number of zero-crossings and if it has a zero local mean. What is important is that IMFs may stand for different intrinsic physical significances hided in the original signal. The EMD is to decompose a signal into such a set of IMFs. In mathematical terms, the EMD for a signal f can be formulated as:

$$f = \sum_{k=1}^{K} d_k + r$$

where, d_k are the IMFs of different frequencies (from higher to lower frequency with respect to the increasing of k, and r is the residue. An EMD may be implemented by using the following algorithm:

(i) Find all local extrema of f;
(ii) Interpolate between all minima (maxima) to get an envelope e_{min} (e_{max}) and compute the mean envelope $m(t) = (e_{min}(t) + e_{max}(t))/2$;
(iii) Extract detail $d(t) = f(t) - m(t)$;
(iv) Check if $|m(t)| < \varepsilon$ for all t; if so, d is an IMF; else, repeat steps (i)–(iii) via replacing f by d;
(v) Calculate residual $r(t) = f(t) - d(t)$;
(vi) Go to step (i) with using r as f;
(vii) Repeat until signal has no extrema.

The EMD of images relies on proper spline interpolation in two dimensions. To find the first IMF it is necessary to start with the input image itself as the input signal $in_{11}(\mathrm{m}, \mathrm{n}) = x(m, n)$. The first index is the IMF number, $l = 1...L$, and the second index is the iteration number, $k = 1...K$, in the sifting process. m and n represent two spatial dimensions. To find the next IMF, the residue corresponding to the previously found IMF is then used as input signal $in_{21}(m, n) = r_1(m, n)$. The sifting process to find the IMFs of a signal $x(m, n)$ follows the next steps:

(a) We find the positions and amplitudes of all local maxima and all local minima in the input signal

(b) We then create the upper and the lower envelope by spline interpolation of the local maxima and the local minima, and denote the envelopes $e_{max}(m, n)$ and $e_{min}(m, n)$ respectively.

(c) For each position (m, n) we then calculate the mean of the upper envelope and the lower envelope:

$$em_{lk}(m, n) = \tfrac{1}{2} \times (e_{max}(m, n) + e_{min}(m, n))$$

The signal $em_{lk}(m, n)$ is referred to as the envelope mean.

(d) Then we subtract the envelope mean signal from the input signal:

$$h_{lk}(m, n) = in_{lk}(m, n) - em_{lk}(m, n)$$

This is an iteration of the sifting process. The next step is to check if the signal $h_{lk}(m, n)$ is the IMF or not. The process stops when the envelope mean signal is close enough to zero:

$$|em_{lk}(m,n)| < \epsilon, \text{ for all } (m,n)$$

Forcing the envelope mean to zero will give the wanted symmetry of the envelope and the correct relation between the number of zero crossings and the number of extremes that define the IMF.

(e) We need to check if the mean signal is close enough to zero, based upon the stop criterion. If not, repeat the process from step 1 with the resulting signal from step (d) as the input signal, sufficient number of times.

$$in_{l(k+1)}(m, n) = h_{lK}(m, n)$$

When the stop criterion is met, $k = K$, the IMF is defined as the last result of (d).

$$c_l(m, n) = h_{lK}(m, n)$$

After the IMF $c_l(m, n)$ is found, we can then define the residue $r_1(m, n)$ as:

$$r_1(m, n) = in_{l1}(m, n) - c_l(m, n)$$

(f) The next IMF is found by starting over from step 1, now with the residue as the input signal

$$in_{(l+1)1}(m, n) = r_l(m, n)$$

Steps from (a) to (f) can be repeated for all the subsequent r_j. The EMD is completed when the residue, ideally, does not contain any extreme points. This means that it is either a constant or a monotonic function. The signal can be expressed as the sum of IMFs and the last residue.

$$x(m,n) = r_L(m,n) + \sum_{j=1}^{L} c_j(m,n)$$

4.2 Multi-Linear PCA

MPCA is a basic algorithm for dimension reduction via multi-linear subspace learning. In wider scope, it belongs to tensor-based computation. Its origin can be traced back to the Tucker decomposition in 1960s and it is closely related to higher-order singular value decomposition, (HOSVD) and to the best rank - (R1, R2…RN) approximation of higher-order tensors. The algorithm is summarized in the pseudocode below.

Input: A set of tensor samples $\{\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m = 1, \ldots, M\}$.

Output: Low-dimensional representations $\{\mathcal{Y}_m \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}, m = 1, \ldots, M\}$ of the input tensor samples with maximum variation captured.

Algorithm:

Step 1 (Preprocessing): Center the input samples as $\{\tilde{\mathcal{X}}_m = \mathcal{X}_m - \bar{\mathcal{X}}, m = 1, \ldots, M\}$, where $\bar{\mathcal{X}} = \frac{1}{M}\sum_{m=1}^{M} \mathcal{X}_m$ is the sample mean.

Step 2 (Initialization): Calculate the eigen-decomposition of $\mathbf{\Phi}^{(n)*} = \sum_{m=1}^{M} \tilde{\mathbf{X}}_{m(n)} \cdot \tilde{\mathbf{X}}_{m(n)}^{T}$ and set $\tilde{\mathbf{U}}^{(n)}$ to consist of the eigenvectors corresponding to the most significant P_n eigenvalues, for $n = 1, \ldots, N$.

Step 3 (Local optimization):

- Calculate $\{\tilde{\mathcal{Y}}_m = \tilde{\mathcal{X}}_m \times_1 \tilde{\mathbf{U}}^{(1)^T} \times_2 \tilde{\mathbf{U}}^{(2)^T} \ldots \times_N \tilde{\mathbf{U}}^{(N)^T}, m = 1, \ldots, M\}$.
- Calculate $\Psi_{\mathcal{Y}_0} = \sum_{m=1}^{M} \| \tilde{\mathcal{Y}}_m \|_F^2$ (the mean $\bar{\tilde{\mathcal{Y}}}$ is all zero since $\tilde{\mathcal{X}}_m$ is centered).
- For $k = 1 : K$
  - For $n = 1 : N$
    - Set the matrix $\tilde{\mathbf{U}}^{(n)}$ to consist of the P_n eigenvectors of the matrix $\mathbf{\Phi}^{(n)}$, as defined in (5), corresponding to the largest P_n eigenvalues.
  - Calculate $\{\tilde{\mathcal{Y}}_m, m = 1, \ldots, M\}$ and $\Psi_{\mathcal{Y}_k}$.
  - If $\Psi_{\mathcal{Y}_k} - \Psi_{\mathcal{Y}_{k-1}} < \eta$, break and go to Step 4.

Step 4 (Projection): The feature tensor after projection is obtained as $\{\mathcal{Y}_m = \mathcal{X}_m \times_1 \tilde{\mathbf{U}}^{(1)^T} \times_2 \tilde{\mathbf{U}}^{(2)^T} \ldots \times_N \tilde{\mathbf{U}}^{(N)^T}, m = 1, \ldots, M\}$.

Fig. 3. Pseudocode implementation of the proposed MPCA algorithm.

MPCA performs feature extraction by determining a multi-linear projection that captures most of the original tensorial input variations. As in PCA, MPCA works on centered data. The MPCA solution follows the alternating least square (ALS) approach. Thus, is iterative in nature and it proceeds by decomposing the original problem to a series of multiple projection sub-problems. Each sub-problem is a classical PCA problem, which can be easily solved.

It should be noted that while PCA with orthogonal transformations produces uncorrelated features/variables, this is not the case for MPCA. Due to the nature of tensor-to-tensor transformation, MPCA features are not uncorrelated in general although the transformation in each mode is orthogonal.

Given:

```
M training images, sized N pixels wide by N pixels tall
c recognition images, also sized N by N pixels
Mp = desired number of principal components
```

Feature Extraction:

```
% merge column vector for each training face
X = [x1 x2 ... xm]
% compute the average face
me = mean(X,2)
A = X - [me me ... me]

% avoids N^2 by N^2 matrix computation of [V,D]=eig(A*A')
% only computes M columns of U: A=U*E*V'
[U,E,V] = svd(A,0)

eigVals = diag(E)
lmda = eigVals(1:Mp)
% pick face-space principal components (eigenfaces)
P = U(:,1:Mp)

% store weights of training data projected into eigenspace
train_wt = P'*A
```

Nearest-Neighbor Classification:

```
% A2 created from the recog data (in similar manner to A)
recog_wt = P'*A2

% euclidean distance for ith recog face, jth train face
euDis(i,j) = sqrt((recog_wt(:,j)-train_wt(:,i)).^2)
```

4.3 Expectation Maximization Algorithm

The EM algorithm is used to find the maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent variables in addition to unknown parameters and known data observations. That is, either there are missing values among the data, or the model can be formulated more simply by assuming the existence of additional unobserved data points. For example, a mixture model can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component that each data point belongs to.

Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values — viz. the parameters and the latent variables — and simultaneously solving the resulting equations. In statistical models with latent variables, this usually is not possible. Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice-versa, but substituting one set of equations into the other produces an unsolvable equation.

The EM algorithm proceeds from the observation that the following is a way to solve these two sets of equations numerically. One can simply pick arbitrary values for one of the two sets of unknowns, use them to estimate the second set, then use these new values to find a better estimate of the first set, and then keep alternating between the two until the resulting values both converge to fixed points. It's not obvious that this will work at all, but in fact it can be proven that in this particular context it does, and that the derivative of the likelihood is (arbitrarily close to) zero at that point, which in turn means that the point is either a maximum or a saddle point. In general there may be multiple maxima, and there is no guarantee that the global maximum will be found. Some likelihood estimates also have singularities in them, i.e. nonsensical maxima. For example, one of the "solutions" that may be found by EM in a mixture model involves setting one of the components to have zero variance and the mean parameter for the same component to be equal to one of the data points.

Given a statistical model consisting of a set $\mathbf{X}$ of observed data, a set of unobserved latent data or missing values $\mathbf{Z}$, and a vector of unknown parameters $\boldsymbol{\theta}$, along with a likelihood function $L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z}) = p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by the marginal likelihood of the observed data

$$L(\boldsymbol{\theta}; \mathbf{X}) = p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

However, this quantity is often intractable (e.g. if $\mathbf{Z}$ is a sequence of events, so that the number of values grows exponentially with the sequence length, making the exact calculation of the sum extremely difficult).

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps:

Expectation Step

Step 1:

Expectation step (E step): Calculate the expected value of the log likelihood function, with respect to the conditional distribution of $\mathbf{Z}$ given $\mathbf{X}$ under the current estimate of the parameters $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathrm{E}_{\mathbf{Z}|\mathbf{X},\boldsymbol{\theta}^{(t)}}\left[\log L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z})\right]$$

Maximization Step

Step 2:

Maximization step (M step): Find the parameter that maximizes this quantity:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$$

Note that in typical models to which EM is applied:

4. The observed data points $\mathbf{X}$ may be discrete (taking values in a finite or countably infinite set) or continuous (taking values in an uncountably infinite set). There may in fact be a vector of observations associated with each data point.
5. The missing values (aka latent variables) $\mathbf{Z}$ are discrete, drawn from a fixed number of values, and there is one latent variable per observed data point.

6. The parameters are continuous, and are of two kinds: Parameters that are associated with all datapoints and parameters associated with a particular value of a latent variable (i.e. associated with all data points whose corresponding latent variable has a particular value).

However, it is possible to apply EM to other sorts of models.

The motivation is as follows. If we know the value of the parameters $\boldsymbol{\theta}$, we can usually find the value of the latent variables $\mathbf{Z}$ by maximizing the log-likelihood over all possible values of $\mathbf{Z}$, either simply by iterating over $\mathbf{Z}$ or through an algorithm such as the Viterbi algorithm for hidden Markov models. Conversely, if we know the value of the latent variables $\mathbf{Z}$, we can find an estimate of the parameters $\boldsymbol{\theta}$ fairly easily, typically by simply grouping the observed data points according to the value of the associated latent variable and averaging the values, or some function of the values, of the points in each group. This suggests an iterative algorithm, in the case where both $\boldsymbol{\theta}$ and $\mathbf{Z}$ are unknown:

5. First, initialize the parameters $\boldsymbol{\theta}$ to some random values.
6. Compute the best value for $\mathbf{Z}$ given these parameter values.
7. Then, use the just-computed values of $\mathbf{Z}$ to compute a better estimate for the parameters $\boldsymbol{\theta}$. Parameters associated with a particular value of $\mathbf{Z}$ will use only those data points whose associated latent variable has that value.
8. Iterate steps 2 and 3 until convergence.

The algorithm as just described monotonically approaches a local minimum of the cost function, and is commonly called *hard EM*. The *k*-means algorithm is an example of this class of algorithms.

However, we can do somewhat better by, rather than making a hard choice for $\mathbf{Z}$ given the current parameter values and averaging only over the set of data points associated with a

particular value of $\mathbf{Z}$ instead determining the probability of each possible value of $\mathbf{Z}$ for each data point, and then using the probabilities associated with a particular value of $\mathbf{Z}$ to compute a weighted average over the entire set of data points. The resulting algorithm is commonly called *soft EM*, and is the type of algorithm normally associated with EM. The counts used to compute these weighted averages are called *soft counts* (as opposed to the *hard counts* used in a hard-EM-type algorithm such as *k*-means). The probabilities computed for $\mathbf{Z}$ are posterior probabilities and are what is computed in the E step. The soft counts used to compute new parameter values are what is computed in the M step.

CHAPTER 5

EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Environmental Setup

The controlled environment has been setup for operation in MATLAB R2012a 64-bit version.

Operating System: Windows 7 Ultimate, 64-bit

RAM: 6GB DDR3

CPU details: Intel Core 2 Duo T6670 2.1GHz

GPU details: NVIDIA GeForce CUDA 310M, 512MB dedicated memory

5.2 Database

MPCA has been tested using the FERET and USF17 Databases. For simplicity of analysis, the programs have been trained using a set of 20 images of 10 different people, and for testing the recognition rates, 10 of those images have been used.

5.3 Results and Discussion

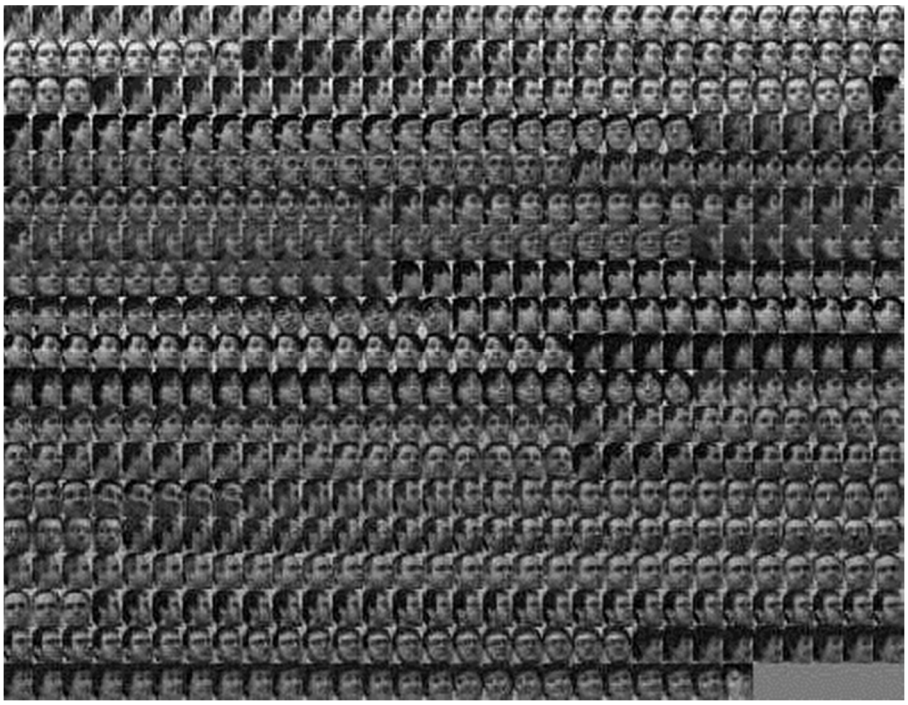
The experiment has been performed in two stages. The Expectation-Maximization Algorithm has been embedded into the Multi-linear PCA code, and as a result acts as a single result. We compare facial recognition rates of a selected sample database once by applying no preprocessing procedure, and once by applying preprocessing using Empirical Mode Decomposition.

The results of our test show that out of 10 test images, we have managed to obtain 7 successful recognitions by implementing MPCA along with EM Algorithm with no preprocessing. Once we apply preprocessing using EMD, we manage to recognize 8 facial images successfully. This improves the success rate of recognition by 10%.

In a formal study, the software was tested against two databases: ALAN & UMIST.

1. UMIST was created by Daniel B. Graham, with a purpose of collecting a controlled set of images that vary pose uniformly from frontal to side view. The UMIST database has 565 total images of 20 people.
2. ALAN was created by the author of the study, Alan C. Brooks, by collecting facial snapshots of people taken at different times. These snapshots were then pre-processed by hand using the gimp to align, normalize lighting, and remove background. This database has 47 total images of 14 people.

The UMIST database images (displayed below) have uniform lighting and pose varying from side to frontal.



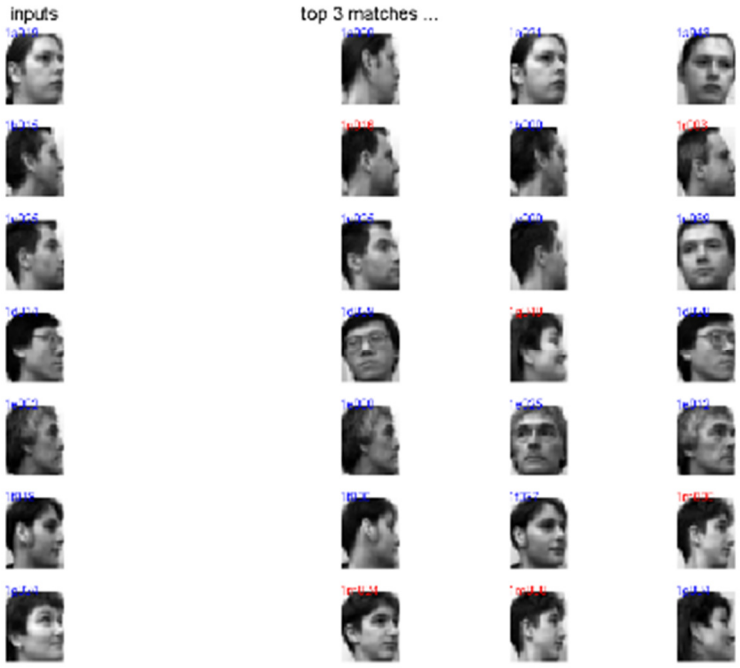
UMIST Results Using PCA in Sparsely Sampled Database

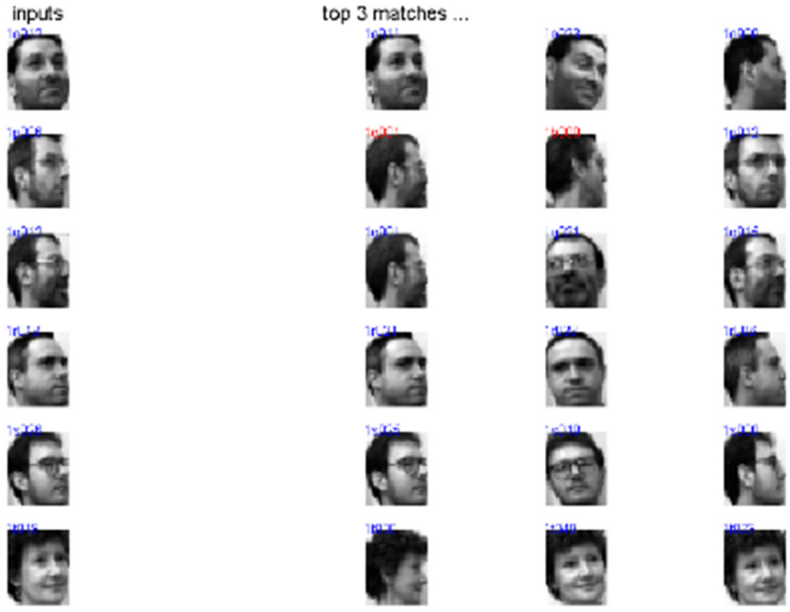
For these results, 20 recognition faces (one for each person) were randomly picked from the database, leaving 545 photos to use as training faces. Mp, the number of principal components to use, was chosen as 20.

UMIST Results Using MPCA in Sparsely Sampled Database

For these results, 20 recognition faces (one for each person) were randomly picked from the database, then 60 more photos were used as training faces. Three training faces were picked for each person: a frontal, side, and 45-degree view.

The resulting plots follow.





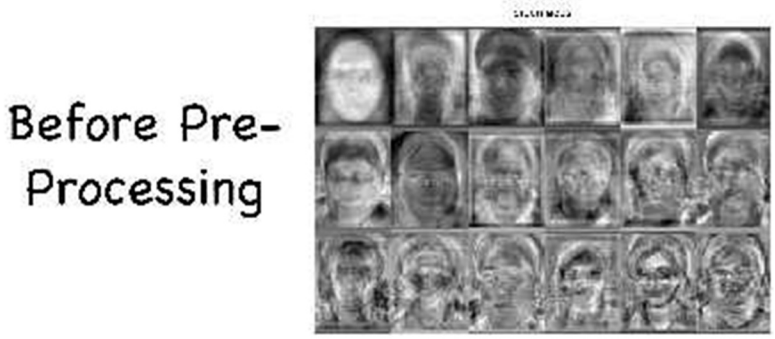
Out of the 20 faces, 16 were correctly classified in the 1st match. Also notice that this approach is rather pose invariant — it often (13 times) picks out all 3 training images from the database.

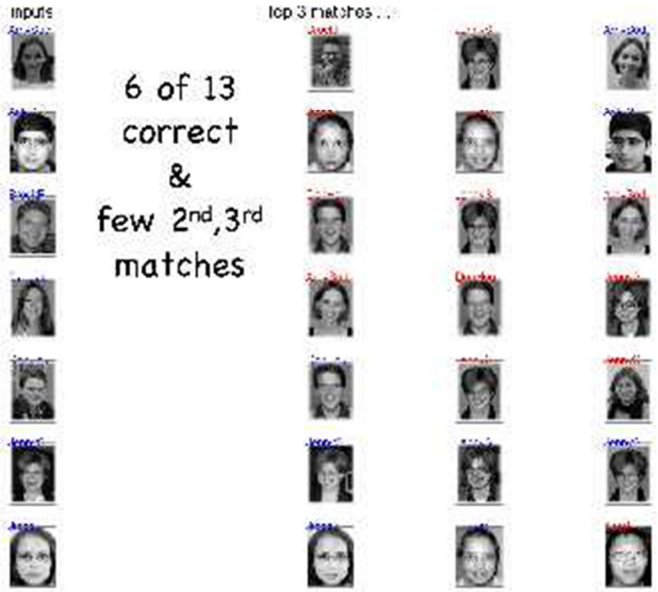
For comparison, results are compared to show the same setup run using the PCA algorithm. Note that 14 of the 20 faces are correctly classified, and 3 correct images are *never* found.

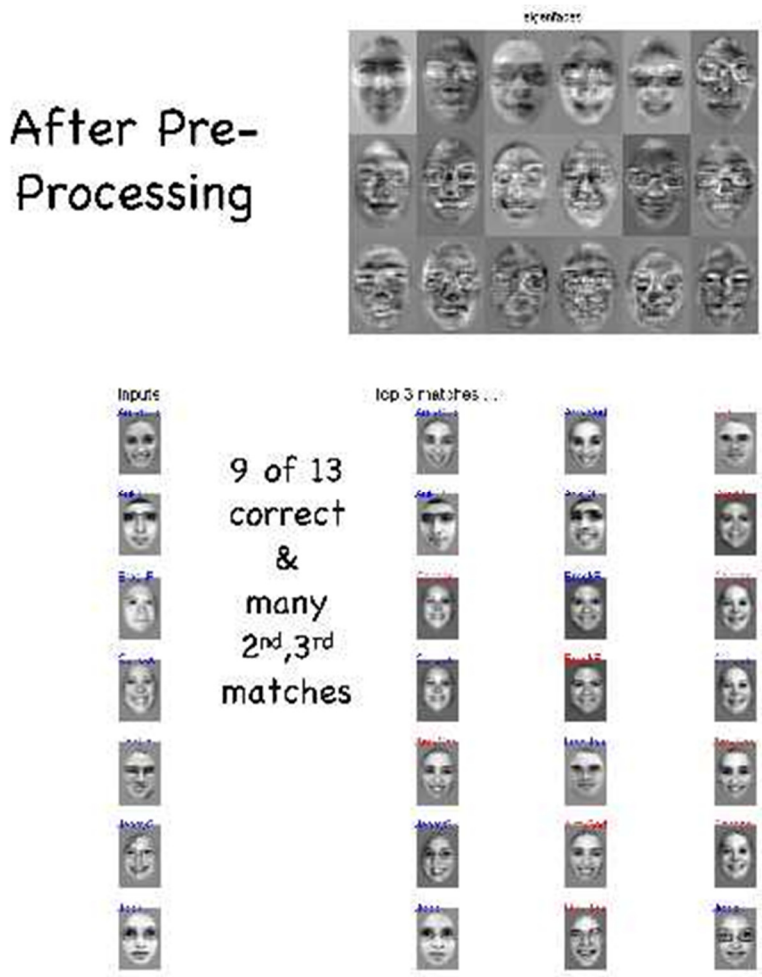
Clearly, the MPCA algorithm performs better under pose variation when only a few samples across pose are available in the training set, as compared to the PCA alone.

Now, we apply EMD preprocessing before running MPCA to observe the results. Pre-processing was used to attempt to remove differences among images in lighting (by normalizing skin tone), scale, alignment, and background.

For these results, 11 recognition faces (one for each person) were randomly picked from the database. Then the remaining 26 photos were used as training faces. The results before and after pre-processing are displayed.







While performance was still not perfect, the EMD pre-processing improved the number of correct classifications from 6 to 9 (of 13). MPCA performance therefore shows improved performance for this database.

5.4 Performance Analysis

In order to summarize the overall performance of MPCA before and after performing EMD preprocessing, we need to look at three aspects separately.

They are- 1) Computational Complexity

2) Effectiveness across several poses

3) Sensitivity to lighting conditions

And the results can be summarized in the table below:

	MPCA with EMD	MPCA
Computational Complexity	Involves preprocessing, so slightly more complex	Relatively simpler method of implementation and fewer calculations
Effectiveness across several poses	Good, even with limited data	Decent performance in a sparse database
Sensitivity to lighting conditions	Very little	Much more dependent on lighting conditions

CHAPTER 6

CONCLUSION

The results of the experiment conclusively show that applying Empirical Mode Decomposition to the images before applying Multi-Linear Principal Component Analysis on it significantly boosts recognition rates. Furthermore, running the EM Algorithm alongside with the MPCA Algorithm also contributes to successful recognition, although with slightly less effectiveness as compared to the EMD preprocessing.

CHAPTER 7

FUTURE WORKS

The research done in this paper can be built and improved in several ways. Each of the three stages of this paper has more advanced versions available. Some may be computationally more expensive, but yield much better results. I have looked into alternative approaches to having a more efficient facial recognition system. These are basically enhanced versions of the processes implemented for this thesis.

Empirical Mode Decomposition can be improved upon by using **Ensemble Empirical Mode Decomposition** (EEMD). This new approach consists of sifting an ensemble of white noise-added signal (data) and treats the mean as the final true result. Finite, not infinitesimal, amplitude white noise is necessary to force the ensemble to exhaust all possible solutions in the sifting process, thus making the different scale signals to collate in the proper intrinsic mode functions (IMF) dictated by the dyadic filter banks. As EEMD is a time–space analysis method, the added white noise is averaged out with sufficient number of trials; the only persistent part that survives the averaging process is the component of the signal (original data), which is then treated as the true and more physical meaningful answer. The effect of the added white noise is to provide a uniform reference frame in the time–frequency space; therefore, the added noise collates the portion of the signal of comparable scale in one IMF. With this ensemble mean, one can separate scales naturally without any a priori subjective criterion selection as in the intermittence test for the original EMD algorithm. This new approach utilizes the full advantage of the statistical characteristics of white noise to perturb the signal in its true solution neighborhood, and to cancel itself out after serving its purpose; therefore, it represents a substantial improvement over the original EMD and is a truly noise-assisted data analysis (NADA) method.

MPCA can be improved by using **Uncorrelated Multi-Linear Principal Component Analysis** (UMPCA). Through successive variance maximization, UMPCA seeks a tensor-to-vector

projection that captures most of the variation in the original tensorial input while producing uncorrelated features. The solution consists of sequential iterative steps based on the alternating projection method. In addition to deriving the UMPCA framework, this work offers a way to systematically determine the maximum number of uncorrelated multi-linear features that can be extracted by the method. Experimental results suggest that UMPCA is particularly effective in determining the low-dimensional projection space needed in such recognition tasks.

There is also an extension of the EM algorithm available, called the **Alpha-EM Algorithm**. The Q-function used in the EM algorithm is based on the log likelihood. Therefore, it is regarded as the log-EM algorithm. The use of the log likelihood can be generalized to that of the α-log likelihood ratio. Then, the α-log likelihood ratio of the observed data can be exactly expressed as equality by using the Q-function of the α-log likelihood ratio and the α-divergence. Obtaining this Q-function is a generalized E step. Its maximization is a generalized M step. This pair is called the α-EM algorithm which contains the log-EM algorithm as its subclass. Thus, the α-EM algorithm by Yasuo Matsuyama is an exact generalization of the log-EM algorithm. No computation of gradient or Hessian matrix is needed. The α-EM shows faster convergence than the log-EM algorithm by choosing an appropriate α. The α-EM algorithm leads to a faster version of the Hidden Markov model estimation algorithm α-HMM.

These are areas which are open for research, as there is no guarantee that applying either process will in fact lead to better facial recognition.

BIBLIOGRAPHY AND REFERENCES

A. Linderhed, *Adaptive Image Compression with Wavelet Packets and Empirical Mode Decomposition,* PhD thesis, Linköping University, Image Coding Group, 2004.

M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, 1991.

M. Turk and A. Pentland, "Face recognition using eigenfaces," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1991, pp. 586-591.

P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711-720, July 1997.

R. Gross, I. Matthews, S. Baker, "Appearance-based face recognition and light-fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 4, pp. 449-465, April 2004.

Rahman, M. A comparative study on face recognition techniques and neural networks. *George Mason University*.

Gottumukkal, R., & Asari, V. (2004). An improved face recognition technique based on modular pca approach. *Pattern Recognition Letters*, *25*.

Savvides, M., & Bhagavatula, R. Analyzing facial images using empirical mode decomposition for illumination artifact removal and improved face recognition. *Carnegie Mellon University*.

Roweis, S. Em algorithms for pca and spca. .

Barisic, B. et.al. Simple iterative algorithm for image enhancement. *University of Split*.

Multi-linear Principal Component Analysis for face recognition with fewer features by Jin Wanga, Armando Barreto, Lu Wang, Yu Chen, Naphtali Rishe, Jean Andrian, Malek Adjouadi – *Florida International University*

PEM-PCA: A Parallel Expectation-Maximization PCA Face Recognition Architecture by Kanokmon Rujirakul, Chakchai So-In, Banchar Arnonkijpanich – *The Scientific World Journal (2014)*

Shams-Baboli, A. et. al. (2013). Face recognition with the mixture of mda and mpca. *Journal of Basic and Applied Scientific Research*.

Xie, X. (2013). Illumination preprocessing for face image sbased on empirical mode decomposition. *Journal of Basic and Applied Scientific Research*.

Lu, H. et.al. (2008). Mpca: Multilinear principal component analysis of tensor objects. *IEEE Transactions on Neural Networks*, *19*(1).

Wang, J. et.al.(2010). Multilinear principal component analysis for face recognition with fewer features. *Neurocomputing, Elsevier*.

Rujirakul, K. et.al. (2014). Pem-pca: A parallel expectation-maximization pca face recognition architecture. *The Scientific World Journal*.

Bailey, S. Principal component analysis with noisy and/or missing data. *Lawrence Berkeley National Lab*.

B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696-710, 1997 (*on-line*).

S. McKenna, Y. Raja, and S. Gong, "Tracking color objects using adaptive mixture models", *Image and Vision Computing*, vol. 17, pp. 225-231, 1999 (*on-line*).

C. Stauffer and E. Grimson, "Adaptive background mixture models for real-time tracking", *IEEE Computer Vision and Pattern Recognition Conference*, Vol. 2, pp. 246-252, 1998 (*on-line*).