Performance measure of pca and dct for images

Generally, in Image Processing the transformation is the basic technique that we apply in order to study the characteristics of the Image under scan. Under this process here we present a method in which we are analyzing the performance of the two methods namely, PCA and DCT. In this thesis we are going to analyze the system by first training the set for particular no. Of images and then analyzing the performance for the two methods by calculating the error in this two methods.

This thesis referred and tested the PCA and DCT transformation techniques.

PCA is a technique which involves a procedure which mathematically transforms number of probably related parameters into smaller number of parameters whose values don’t change called principal components. The primary principal component accounts for much variability in the data, and each succeeding component accounts for much of the remaining variability. Depending on the application field, it is also called the separate Karhunen-Loève transform (KLT), the Hotelling transform or proper orthogonal decomposition (POD).

DCT expresses a series of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies.

Transformations are important to numerous applications in science and engineering, from lossy compression of audio and images (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations.

CHAPTER 1

INTRODUCTION

1. 1 Introduction

Over the past few years, several face recognition systems have been proposed based on principal components analysis (PCA) [14, 8, 13, 15, 1, 10, 16, 6]. Although the details vary, these systems can all be described in terms of the same preprocessing and run-time steps. During preprocessing, they register a gallery of m training images to each other and unroll each image into a vector of n pixel values. Next, the mean image for the gallery is subtracted from each and the resulting " centered" images are placed in a gallery matrix M. Element [i; j] of M is the ith pixel from the jth image. A covariance matrix W = MMT characterizes the distribution of the m images in n. A subset of the Eigenvectors of W are used as the basis vectors for a subspace in which to compare gallery and novel probe images. When sorted by decreasing Eigenvalue, the full set of unit length Eigenvectors represent an orthonormal basis where the first direction corresponds to the direction of maximum variance in the images, the second the next largest variance, etc. These basis vectors are the Principle Components of the gallery images. Once the Eigenspace is computed, the centered gallery images are projected into this subspace. At run-time, recognition is accomplished by projecting a centered probe image into the subspace and the nearest gallery image to the probe image is selected as its match. There are many differences in the systems referenced. Some systems assume that the images are registered prior to face recognition [15, 10, 11, 16]; among the rest, a variety of techniques are used to identify facial features and register them to each

other. Different systems may use different distance measures when matching probe images to the nearest gallery image. Different systems select different numbers of Eigenvectors (usually those corresponding to the largest k Eigenvalues) in order to compress the data and to improve accuracy by eliminating Eigenvectors corresponding to noise rather than meaningful variation. To help evaluate and compare individual steps of the face recognition process, Moon and Phillips created the FERET face database, and performed initial comparisons of some common distance measures for otherwise identical systems [10, 11, 9]. This paper extends their work, presenting further comparisons of distance measures over the FERET database and examining alternative way of selecting subsets of Eigenvectors. The Principal Component Analysis (PCA) is one of the most successful techniques that have been used in image recognition and compression. PCA is a statistical method under the broad title of factor analysis. The purpose of PCA is to reduce the large dimensionality of the data space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which are needed to describe the data economically. This is the case when there is a strong correlation between observed variables. The jobs which PCA can do are prediction, redundancy removal, feature extraction, data compression, etc. Because PCA is a classical technique which can do something in the linear domain, applications having linear models are suitable, such as signal processing, image processing, system and control theory, communications, etc. Face recognition has many applicable areas. Moreover, it can be categorized into face identification, face classification, or sex determination. The most useful applications contain crowd surveillance, video content indexing, personal

identification (ex. driver’s license), mug shots matching, entrance security, etc. The main idea of using PCA for face recognition is to express the large 1-D vector of pixels constructed from 2-D facial image into the compact principal components of the feature space. This can be called eigen space projection. Eigen space is calculated by identifying the eigenvectors of the covariance matrix derived from a set of facial images(vectors). The details are described in the following section.

PCA computes the basis of a space which is represented by its training vectors. These basis vectors, actually eigenvectors, computed by PCA are in the direction of the largest variance of the training vectors. As it has been said earlier, we call them eigenfaces. Each eigenface can be viewed a feature. When a particular face is projected onto the face space, its vector into the face space describe the importance of each of those features in the face. The face is expressed in the face space by its eigenface coefficients (or weights). We can handle a large input vector, facial image, only by taking its small weight vector in the face space. This means that we can reconstruct the original face with some error, since the dimensionality of the image space is much larger than that of face space.

A face recognition system using the Principal Component Analysis (PCA) algorithm. Automatic face recognition systems try to find the identity of a given face image according to their memory. The memory of a face recognizer is generally simulated by a training set. In this project, our training set consists of the features extracted from known face images of different persons. Thus, the task of the face recognizer is to find the most similar feature vector among the training set to the feature vector of a given

test image. Here, we want to recognize the identity of a person where an image of that person (test image) is given to the system. You will use PCA as a feature extraction algorithm in this project. In the training phase, you should extract feature vectors for each image in the training set. Let A be a training image of person A which has a pixel resolution of M £ N (M rows, N columns). In order to extract PCA features of A, you will first convert the image into a pixel vector ÁA by concatenating each of the M rows into a single vector. The length (or, dimensionality) of the vector ÁA will be M £N. In this project, you will use the PCA algorithm as a dimensionality reduction technique which transforms the vector ÁA to a vector ! A which has a imensionality d where d ¿ M £ N. For each training image i, you should calculate and store these feature vectors ! i. In the recognition phase (or, testing phase), you will be given a test image j of a known person. Let ®j be the identity (name) of this person. As in the training phase, you should compute the feature vector of this person using PCA and obtain ! j . In order to identify j , you should compute the similarities between ! j and all of the feature vectors ! i’s in the training set. The similarity between feature vectors can be computed using Euclidean distance. The identity of the most similar ! i will be the output of our face recognizer. If i = j, it means that we have correctly identified the person j, otherwise if i 6= j, it means that we have misclassified the person j.

1. 2 Thesis structure:

This thesis work is divided into five chapters as follows.

Chapter 1: Introduction

This introductory chapter is briefly explains the procedure of transformation in the Face Recognition and its applications. And here we explained the scope of this research. And finally it gives the structure of the thesis for friendly usage.

Chapter 2: Basis of Transformation Techniques.

This chapter gives an introduction to the Transformation techniques. In this chapter we have introduced two transformation techniques for which we are going to perform the analysis and result are used for face recognition purpose

Chapter 3: Discrete Cosine Transformation

In this chapter we have continued the part from chapter 2 about transformations. In this other method ie., DCT is introduced and analysis is done

Chapter 4: Implementation and results

This chapter presents the simulated results of the face recognition analysis using MATLAB. And it gives the explanation for each and every step of the design of face recognition analysis and it gives the tested results of the transformation algorithms.

Chapter 5: Conclusion and Future work

This is the final chapter in this thesis. Here, we conclude our research and discussed about the achieved results of this research work and suggested future work for this research.

CHAPTER 2

BASICs of Image Transform Techniques

2. 1 Introduction:

Now a day's Image Processing has been gained so much of importance that in every field of science we apply image processing for the purpose of security as well as increasing demand for it. Here we apply two different transformation techniques in order study the performance which will be helpful in the detection purpose. The computation of the performance of the image given for testing is performed in two steps:

PCA (Principal Component Analysis)

DCT (Discrete Cosine Transform)

2. 2 Principal Component Analysis:

PCA is a technique which involves a procedure which mathematically transforms number of possibly correlated variables into smaller number of uncorrelated variables called principal components. The first principal component accounts for much variability in the data, and each succeeding component accounts for much of the remaining variability. Depending on the application field, it is also called the discrete Karhunen-Loève transform (KLT), the Hotelling transform or proper orthogonal decomposition (POD).

Now PCA is mostly used as a tool in exploration of data analysis and for making prognostic models. PCA also involves calculation for the Eigen value decomposition of a data covariance matrix or singular value decomposition of a data matrix, usually after mean centring the data from each attribute.

The results of this analysis technique are usually shown in terms of component scores and also as loadings.

PCA is real Eigen based multivariate analysis. Its action can be termed in terms of as edifying the inner arrangement of the data in a shape which give details of the mean and variance in the data. If there is any multivariate data then it’s visualized as a set if coordinates in a multi dimensional data space, this algorithm allows the users having pictures with a lower aspect reveal a shadow of object in view from a higher aspect view which reveals the true informative nature of the object.

PCA is very closely related to aspect analysis, some statistical software packages purposely conflict the two techniques. True aspect analysis makes different assumptions about the original configuration and then solves eigenvectors of a little different medium.

2. 2. 1 PCA Implementation:

PCA is mathematically defined as an orthogonal linear transformation technique that transforms data to a new coordinate system, such that the greatest variance from any projection of data comes to lie on the first coordinate, the second greatest variance on the second coordinate, and so on. PCA is theoretically the optimum transform technique for given data in least square terms.

For a data matrix, XT, with zero empirical mean ie., the empirical mean of the distribution has been subtracted from the data set, where each row represents a different repetition of the experiment, and each column gives the results from a particular probe, the PCA transformation is given by:

Where the matrix Î£ is an m-by-n diagonal matrix, where diagonal elements ae non-negative and W Î£ VT is the singular value decomposition of X.

Given a set of points in Euclidean space, the first principal component part corresponds to the line that passes through the mean and minimizes the sum of squared errors with those points. The second principal component corresponds to the same part after all the correlation terms with the first principal component has been subtracted from the points. Each Eigen value indicates the part of the variance ie., correlated with each eigenvector. Thus, the sum of all the Eigen values is equal to the sum of squared distance of the points with their mean divided by the number of dimensions. PCA rotates the set of points around its mean in order to align it with the first few principal components. This moves as much of the variance as possible into the first few dimensions. The values in the remaining dimensions tend to be very highly correlated and may be dropped with minimal loss of information. PCA is used for dimensionality reduction. PCA is optimal linear transformation technique for keeping the subspace which has largest variance. This advantage comes with the price of greater computational requirement. In discrete cosine transform, Non-linear dimensionality reduction techniques tend to be more computationally demanding in comparison with PCA.

Mean subtraction is necessary in performing PCA to ensure that the first principal component describes the direction of maximum variance. If mean subtraction is not performed, the first principal component will instead correspond to the mean of the data. A mean of zero is needed for finding a basis that minimizes the mean square error of the approximation of the data.

Assuming zero empirical mean (the empirical mean of the distribution has been subtracted from the data set), the principal component w1 of a data set x can be defined as:

With the first k âˆ’ 1 component, the kth component can be found by subtracting the first k âˆ’ 1 principal components from x:

and by substituting this as the new data set to find a principal component in

The other transform is therefore equivalent to finding the singular value decomposition of the data matrix X,

and then obtaining the space data matrix Y by projecting X down into the reduced space defined by only the first L singular vectors, WL:

The matrix W of singular vectors of X is equivalently the matrix W of eigenvectors of the matrix of observed covariance's C = X XT,

The eigenvectors with the highest eigen values correspond to the dimensions that have the strongest correlation in the data set (see Rayleigh quotient).

PCA is equivalent to empirical orthogonal functions (EOF), a name which is used in meteorology.

An auto-encoder neural network with a linear hidden layer is similar to PCA. Upon convergence, the weight vectors of the K neurons in the hidden layer will form a basis for the space spanned by the first K principal components. Unlike PCA, this technique will not necessarily produce orthogonal vectors.

PCA is a popular primary technique in pattern recognition. But it’s not optimized for class separability. An alternative is the linear discriminant analysis, which does take this into account.

2. 2. 2 PCA Properties and Limitations

PCA is theoretically the optimal linear scheme, in terms of least mean square error, for compressing a set of high dimensional vectors into a set of lower dimensional vectors and then reconstructing the original set. It is a non-parametric analysis and the answer is unique and independent of any hypothesis about data probability distribution. However, the latter two properties are regarded as weakness as well as strength, in that being non-parametric, no prior knowledge can be incorporated and that PCA compressions often incur loss of information.

The applicability of PCA is limited by the assumptions[5] made in its derivation. These assumptions are:

We assumed the observed data set to be linear combinations of certain basis. Non-linear methods such as kernel PCA have been developed without assuming linearity.

PCA uses the eigenvectors of the covariance matrix and it only finds the independent axes of the data under the Gaussian assumption. For non-Gaussian or multi-modal Gaussian data, PCA simply de-correlates the axes. When PCA is used for clustering, its main limitation is that it does not account for class separability since it makes no use of the class label of the feature vector. There is no guarantee that the directions of maximum variance will contain good features for discrimination.

PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance. It is only when we believe that the observed data has a high signal-to-noise ratio that the principal components with larger variance correspond to interesting dynamics and lower ones correspond to noise.

2. 2. 3 Computing PCA with covariance method

Following is a detailed description of PCA using the covariance method . The goal is to transform a given data set X of dimension M to an alternative data set Y of smaller dimension L. Equivalently; we are seeking to find the matrix Y, where Y is the KLT of matrix X:

Organize the data set

Suppose you have data comprising a set of observations of M variables, and you want to reduce the data so that each observation can be described with only L variables, L < M. Suppose further, that the data are arranged as a set of N data vectors with each representing a single grouped observation of the M variables.

Write as column vectors, each of which has M rows.

Place the column vectors into a single matrix X of dimensions M Ã- N.

Calculate the empirical mean

Find the empirical mean along each dimension m = 1, ..., M.

Place the calculated mean values into an empirical mean vector u of dimensions M Ã- 1.

Calculate the deviations from the mean

Mean subtraction is an integral part of the solution towards finding a principal component basis that minimizes the mean square error of approximating the data. Hence we proceed by centering the data as follows:

Subtract the empirical mean vector u from each column of the data matrix X.

Store mean-subtracted data in the M Ã- N matrix B.

where h is a 1 Ã- N row vector of all 1s:

Find the covariance matrix

Find the M Ã- M empirical covariance matrix C from the outer product of matrix B with itself:

where

is the expected value operator,

is the outer product operator, and

is the conjugate transpose operator.

Please note that the information in this section is indeed a bit fuzzy. Outer products apply to vectors, for tensor cases we should apply tensor products, but the covariance matrix in PCA, is a sum of outer products between its sample vectors, indeed it could be represented as B. B*. See the covariance matrix sections on the discussion page for more information.

Find the eigenvectors and eigenvalues of the covariance matrix

Compute the matrix V of eigenvectors which diagonalizes the covariance matrix C:

where D is the diagonal matrix of eigenvalues of C. This step will typically involve the use of a computer-based algorithm for computing eigenvectors and eigenvalues. These algorithms are readily available as sub-components of most matrix algebra systems, such as MATLAB[7][8], Mathematica[9], SciPy, IDL(Interactive Data Language), or GNU Octave as well as OpenCV.

Matrix D will take the form of an M Ã- M diagonal matrix, where

is the mth eigenvalue of the covariance matrix C, and

Matrix V, also of dimension M Ã- M, contains M column vectors, each of length M, which represent the M eigenvectors of the covariance matrix C.

The eigenvalues and eigenvectors are ordered and paired. The mth eigenvalue corresponds to the mth eigenvector.

Rearrange the eigenvectors and eigenvalues

Sort the columns of the eigenvector matrix V and eigenvalue matrix D in order of decreasing eigenvalue.

Make sure to maintain the correct pairings between the columns in each matrix.

Compute the cumulative energy content for each eigenvector

The eigenvalues represent the distribution of the source data’s energy among each of the eigenvectors, where the eigenvectors form a basis for the data. The cumulative energy content g for the mth eigenvector is the sum of the energy content across all of the eigenvalues from 1 through m:

Select a subset of the eigenvectors as basis vectors

Save the first L columns of V as the M Ã- L matrix W:

where

Use the vector g as a guide in choosing an appropriate value for L. The goal is to choose a value of L as small as possible while achieving a reasonably high value of g on a percentage basis. For example, you may want to choose L so that the cumulative energy g is above a certain threshold, like 90 percent. In this case, choose the smallest value of L such that

Convert the source data to z-scores

Create an M Ã- 1 empirical standard deviation vector s from the square root of each element along the main diagonal of the covariance matrix C:

Calculate the M Ã- N z-score matrix:

(divide element-by-element)

Note: While this step is useful for various applications as it normalizes the data set with respect to its variance, it is not integral part of PCA/KLT!

Project the z-scores of the data onto the new basis

The projected vectors are the columns of the matrix

W* is the conjugate transpose of the eigenvector matrix.

The columns of matrix Y represent the Karhunen-Loeve transforms (KLT) of the data vectors in the columns of matrix X.

2. 2. 4 PCA Derivation

Let X be a d-dimensional random vector expressed as column vector. Without loss of generality, assume X has zero mean. We want to find a Orthonormal transformation matrix P such that

with the constraint that

is a diagonal matrix and

By substitution, and matrix algebra, we obtain:

We now have:

Rewrite P as d column vectors, so

and as:

Substituting into equation above, we obtain:

Notice that in , Pi is an eigenvector of the covariance matrix of X. Therefore, by finding the eigenvectors of the covariance matrix of X, we find a projection matrix P that satisfies the original constraints.

CHAPTER 3

DISCRETE Cosine transform

3. 1 Introduction:

A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in engineering, from lossy compression of audio and images, to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications: for compression, it turns out that cosine functions are much more efficient, whereas for differential equations the cosines express a particular choice of boundary conditions.

In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. There are eight standard DCT variants, of which four are common.

The most common variant of discrete cosine transform is the type-II DCT, which is often called simply “ the DCT”; its inverse, the type-III DCT, is correspondingly often called simply “ the inverse DCT” or “ the IDCT”. Two related transforms are the discrete sine transforms (DST), which is equivalent to a DFT of real and odd functions, and the modified discrete cosine transforms (MDCT), which is based on a DCT of overlapping data.

3. 2 DCT forms:

Formally, the discrete cosine transform is a linear, invertible function F : RN -> RN, or equivalently an invertible N Ã- N square matrix. There are several variants of the DCT with slightly modified definitions. The N real numbers x0, ..., xN-1 are transformed into the N real numbers X0, ..., XN-1 according to one of the formulas:

DCT-I

Some authors further multiply the x0 and xN-1 terms by âˆš2, and correspondingly multiply the X0 and XN-1 terms by 1/âˆš2. This makes the DCT-I matrix orthogonal, if one further multiplies by an overall scale factor of , but breaks the direct correspondence with a real-even DFT.

The DCT-I is exactly equivalent, to a DFT of 2N â^' 2 real numbers with even symmetry. For example, a DCT-I of N= 5 real numbers abcde is exactly equivalent to a DFT of eight real numbers abcdedcb, divided by two.

Note, however, that the DCT-I is not defined for N less than 2.

Thus, the DCT-I corresponds to the boundary conditions: xn is even around n= 0 and even around n= N-1; similarly for Xk.

DCT-II

The DCT-II is probably the most commonly used form, and is often simply referred to as “ the DCT”.

This transform is exactly equivalent to a DFT of 4N real inputs of even symmetry where the even-indexed elements are zero. That is, it is half of the

DFT of the 4N inputs yn, where y2n = 0, y2n + 1 = xn for , and y4N âˆ’ n = yn for 0 < n < 2N.

Some authors further multiply the X0 term by 1/âˆš2 and multiply the resulting matrix by an overall scale factor of . This makes the DCT-II matrix orthogonal, but breaks the direct correspondence with a real-even DFT of half-shifted input.

The DCT-II implies the boundary conditions: xn is even around n=-1/2 and even around n= N-1/2; Xk is even around k= 0 and odd around k= N.

DCT-III

Because it is the inverse of DCT-II (up to a scale factor, see below), this form is sometimes simply referred to as “ the inverse DCT” (“ IDCT”).

Some authors further multiply the x0 term by âˆš2 and multiply the resulting matrix by an overall scale factor of , so that the DCT-II and DCT-III are transposes of one another. This makes the DCT-III matrix orthogonal, but breaks the direct correspondence with a real-even DFT of half-shifted output.

The DCT-III implies the boundary conditions: xn is even around n= 0 and odd around n= N; Xk is even around k=-1/2 and even around k= N-1/2.

DCT-IV

The DCT-IV matrix becomes orthogonal if one further multiplies by an overall scale factor of .

A variant of the DCT-IV, where data from different transforms are overlapped, is called the modified discrete cosine transform (MDCT) (Malvar, 1992).

The DCT-IV implies the boundary conditions: xn is even around n=-1/2 and odd around n= N-1/2; similarly for Xk.

DCT V-VIII

DCT types I-IV are equivalent to real-even DFTs of even order, since the corresponding DFT is of length 2(Nâˆ’1) (for DCT-I) or 4N (for DCT-II/III) or 8N (for DCT-VIII). In principle, there are actually four additional types of discrete cosine transform, corresponding essentially to real-even DFTs of logically odd order, which have factors of N±½ in the denominators of the cosine arguments.

Equivalently, DCTs of types I-IV imply boundaries that are even/odd around either a data point for both boundaries or halfway between two data points for both boundaries. DCTs of types V-VIII imply boundaries that even/odd around a data point for one boundary and halfway between two data points for the other boundary.

However, these variants seem to be rarely used in practice. One reason, perhaps, is that FFT algorithms for odd-length DFTs are generally more complicated than FFT algorithms for even-length DFTs (e. g. the simplest radix-2 algorithms are only for even lengths), and this increased intricacy carries over to the DCTs as described below.

Inverse transforms

Using the normalization conventions above, the inverse of DCT-I is DCT-I multiplied by 2/(N-1). The inverse of DCT-IV is DCT-IV multiplied by 2/N. The inverse of DCT-II is DCT-III multiplied by 2/N and vice versa.

Like for the DFT, the normalization factor in front of these transform definitions is merely a convention and differs between treatments. For example, some authors multiply the transforms by so that the inverse does not require any additional multiplicative factor. Combined with appropriate factors of âˆš2 (see above), this can be used to make the transform matrix orthogonal.

Multidimensional DCTs

Multidimensional variants of the various DCT types follow straightforwardly from the one-dimensional definitions: they are simply a separable product (equivalently, a composition) of DCTs along each dimension.

For example, a two-dimensional DCT-II of an image or a matrix is simply the one-dimensional DCT-II, from above, performed along the rows and then along the columns (or vice versa). That is, the 2d DCT-II is given by the formula (omitting normalization and other scale factors, as above):

Two-dimensional DCT frequencies

Technically, computing a two- (or multi-) dimensional DCT by sequences of one-dimensional DCTs along each dimension is known as a row-column algorithm. As with multidimensional FFT algorithms, however, there exist

other methods to compute the same thing while performing the computations in a different order.

The inverse of a multi-dimensional DCT is just a separable product of the inverse(s) of the corresponding one-dimensional DCT(s), e. g. the one-dimensional inverses applied along one dimension at a time in a row-column algorithm.

The image to the right shows combination of horizontal and vertical frequencies for an 8 x 8 (N1 = N2 = 8) two-dimensional DCT. Each step from left to right and top to bottom is an increase in frequency by 1/2 cycle. For example, moving right one from the top-left square yields a half-cycle increase in the horizontal frequency. Another move to the right yields two half-cycles. A move down yields two half-cycles horizontally and a half-cycle vertically. The source data (8×8) is transformed to a linear combination of these 64 frequency squares.

Chapter 4

IMPLEMENTATION AND RESULTS

4. 1 Introduction:

In previous chapters (chapter 2 and chapter 3), we get the theoretical knowledge about the Principal Component Analysis and Discrete Cosine Transform. In our thesis work we have seen the analysis of both transform. To execute these tasks we chosen a platform called “ MATLAB”, stands for matrix laboratory. It is an efficient language for Digital image processing. The image processing toolbox in MATLAB is a collection of different MATAB

functions that extend the capability of the MATLAB environment for the solution of digital image processing problems. [13]

4. 2 Practical implementation of Performance analysis:

As discussed earlier we are going to perform analysis for the two transform methods, to the images as,

<