A REPORT ON

" BIOMETRICS: FACIAL RECOGNITION "

Prepared in partial fulfillment of the Laboratory Project
(EEE F366) Course
At
BITS Pilani

Submitted to: **Dr. Pawan Ajmera**

Assistant Professor

Department of Electrical and Electronics Engineering

Birla Institute of Technology and Science Pilani, Pilani campus

Submitted by: **KAPIL AGRAWAL (2014B3A30579P)**

ACKNOWLEDGEMENT

I would like to thank my teacher and mentor Dr. Pawan Ajmera for giving me an opportunity to work on this interesting topic “Biometrics: Facial Recognition “under his guidance.

INTRODUCTION

As one of several methods of what are called “biometric” identification systems, facial recognition examines physical features of a person’s body in an attempt to uniquely distinguish one person from all the others. Other forms of this type of work include the very common fingerprint matching, retina scanning, iris scanning and even voice recognition.
Facial recognition technology is the least intrusive and the fastest of biometric technology. It works with the most obvious individual identifier of the human face. With the growing security needs and technological advances, the extraction of information has been greatly simplified. This project aims to build an application based on the recognition of faces using different algorithms. This project implements a face recognition framework in Python programming language. The basic objective is to identify the face of any person in the trained dataset, when given test image. It involves two main steps. First to identify the distinguishing factors of the image n that stores them and the second step to compare them with the existing images and return the data related to that image. The facial features that can be used are mean, standard deviation, light intensity, etc. But, using simpler features is too common and the accuracy of the face model is not that high. Hence, we need to use complex features like Principle component analysis, Fisher face feature, etc. The main algorithms used for face detection in this project is Fisher face.

Keywords
Face Recognition, Python, Fisher face Algorithm

PROBLEM STATEMENT:

The main problem can be stated simply as: Given a set of many face images labeled with the person's identity (the training set) and an unlabeled set of face images from the same group of people (the test set), we need to identify the name of each person in the test images.

METHODOLOGY

Obtaining dataset: I used the dataset obtained online from this link – http://www.anefian.com/research/face_reco.htm . I used the dataset available on this link for the whole training purpose and for the test purpose too. This dataset is enough big, so that we can scale our code easily on a larger dataset.

Classifier used: A classifier is needed to classify the images correctly. I chose the k nearest classifier. Pseudo code for the KNN classifier looks like this-

For every point in our dataset:

- *Calculate the distance between test point and the current point*
- *Sort the distances in increasing order*
- *Take k items with lowest distances to test point*
- *Find the majority class among these items*
- *Return the majority class as our prediction for the class of test point*

The code for classifier is as follows :

classifier.py

```
import operator as op

import numpy as np
from sklearn import svm

from facerec.distance import EuclideanDistance
from facerec.util import asRowMatrix


class AbstractClassifier(object):

    def compute(self,X,y):
        raise NotImplementedError("Every AbstractClassifier must implement the compute method.")

    def predict(self,X):
        raise NotImplementedError("Every AbstractClassifier must implement the predict method.")

    def update(self,X,y):
        raise NotImplementedError("This Classifier is cannot be updated.")

class NearestNeighbor(AbstractClassifier):

    """
    Implements a k-Nearest Neighbor Model with a generic distance metric.
    """
    def __init__(self, dist_metric=EuclideanDistance(), k=1):
        AbstractClassifier.__init__(self)
        self.k = k
        self.dist_metric = dist_metric
        self.X = []
        self.y = np.array([], dtype=np.int32)

    def update(self, X, y):
        """
        Updates the classifier.
        """
        self.X.append(X)
        self.y = np.append(self.y, y)
```

Validation: The accuracy and precision of the model can be checked through the performance of our classifier. I used K-fold cross validation technique, which basically separates the data into training and test sets according to the value of k chosen.
The screenshot of the validation code and results obtained is:

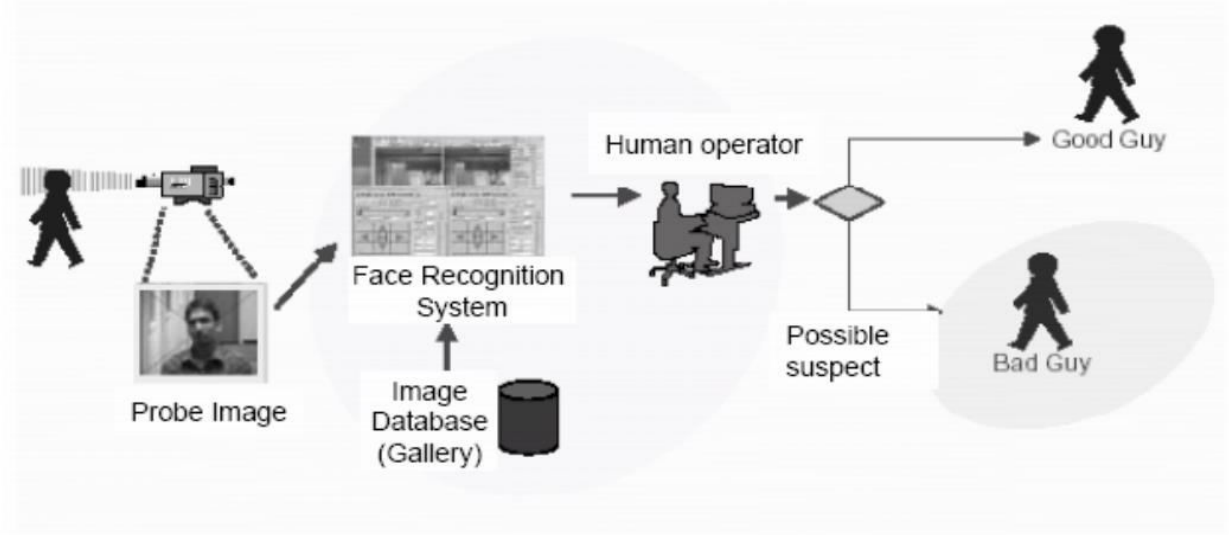
validation.py

```
class SimpleValidation(ValidationStrategy):
    """Implements a simple Validation, which allows you to partition the data yourself.
    """
    def __init__(self, model):
        """
        Args:
            model [PredictableModel] model to perform the validation on
        """
        super(SimpleValidation, self).__init__(model=model)
        self.logger = logging.getLogger("facerec.validation.SimpleValidation")

    def validate(self, Xtrain, ytrain, Xtest, ytest, description="ExperimentName"):
        """

        Performs a validation given training data and test data. User is responsible for non-overlapping assignment of indices.

        Args:
            X [dim x num_data] input data to validate on
            y [1 x num_data] classes
        """
        self.logger.info("Simple Validation.")

        self.model.compute(Xtrain, ytrain)

        self.logger.debug("Model computed.")

        true_positives, false_positives, true_negatives, false_negatives = (0,0,0,0)
        count = 0
        for i in ytest:
            self.logger.debug("Predicting %s/%s." % (count, len(ytest)))
            prediction = self.model.predict(Xtest[i])[0]
            if prediction == ytest[i]:
                true_positives = true_positives + 1
            else:
```

```
Kapils-MacBook-Pro:scripts kapil$ python3 simple_example.py ../../../gt_db
2017-12-06 14:14:42,432 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 1/8.
2017-12-06 14:16:26,774 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 2/8.
2017-12-06 14:17:44,164 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 3/8.
2017-12-06 14:19:07,094 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 4/8.
2017-12-06 14:20:27,246 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 5/8.
2017-12-06 14:21:40,030 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 6/8.
2017-12-06 14:22:52,561 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 7/8.
2017-12-06 14:24:05,442 - facerec.validation.KFoldCrossValidation - INFO - Processing fold 8/8.
PredictableModel (feature=Fisherfaces (num_components=19), classifier=NearestNeighbor (k=1, dist_metric=EuclideanDistance))
ValidationResult (Description=ExperimentName, Precision=98.75%, Accuracy=98.75%)
Kapils-MacBook-Pro:scripts kapil$
```

The face recognition accuracy obtained on the test images is 98.75%, which is well satisfactory.

How any Face Recognition system works?



A face recognition system is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database.

Initially, we have our dataset of images with us. Then, we need to develop a face recognition system, which involves special image features and the us, e of machine learning. We then give an input image as a test to our system, which is later, used to calculate its image features and the whole feature extraction is done for that image. We also do the feature extraction and computation of all the images in our dataset which are used as training. After this, we use some distance metric and calculate the point, which is closest in distance between the test image and the stored features of the trained images. Then, the classifier used outputs the calculated label for that test image.

Principle Component Analysis:

• Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components.

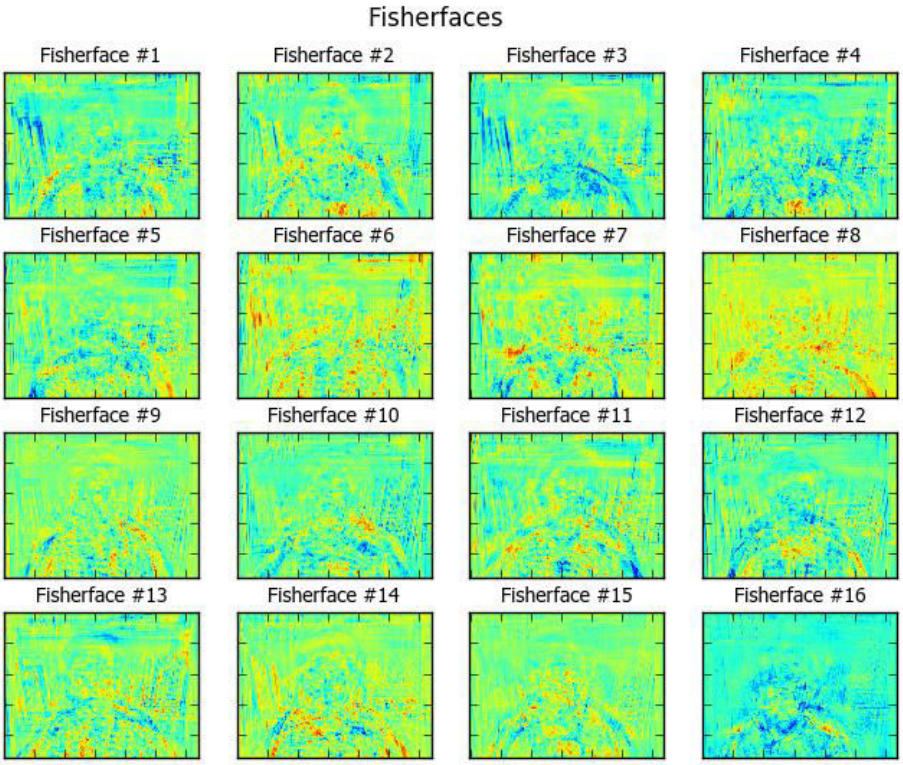
• The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

The Eigenface is the first method considered as a successful technique of face recognition. The Eigenface method uses Principal Component Analysis (PCA) to linearly project the image space to a low dimensional feature space.

Fisher face feature:

The Fisher face method is an enhancement of the Eigenface method that it uses Fisher's Linear Discriminant Analysis (FLDA or LDA) for the dimensionality reduction. The LDA maximizes the ratio of between-class scatter to that of within-class scatter; therefore, it works better than PCA for purpose of discrimination. The Fisher face is especially useful when facial images have large variations in illumination and facial expression.
Fisher face wants to maximize the mean distance of different classes while minimize the variance within class. They get face models that are more useful in discrimination.

Let's see an example of how the fisher face feature looks like when I run the final code on the used dataset:



Principle component Analysis (PCA) vs Linear Discriminant Analysis (LDA) :

- Usually a PCA is done before an LDA and the LDA are much better in separating data in the case of facial recognition.
- LDA is supervised algorithm, whereas PCA is unsupervised algorithm,.
- Fisher face Algorithm which uses LDA is more robust to variations such as lighting direction and facial expressions, when compared to Eigenface algorithm which uses PCA approach.

CONCLUSION

The main limitations of the Eigenface method using PCA algorithm is that it does not take labeling into consideration and when the images have just small variations, it does not work well in separating it more far apart. Whereas, fisher face method using LDA algorithm uses the training labels is more efficient and accurate while it comes to classify and recognized the similar images. Fisher face works very good fir Intra-class and solves most of the limitations of PCA. The accuracy of the Model depends on the value of k chosen in the K-fold cross validation technique. The accuracy obtained is around **98%**, which is quiet good and tells us that Fisher face is better able to recognize the images of our dataset.

REFERENCES

- http://ijarece.org/wp-content/uploads/2013/08/IJARECE-VOL-2-ISSUE-2-149-154.pdf

- https://en.wikipedia.org/wiki/Facial_recognition_system
- https://www.iti.gr/iti/files/document/seminars/FR2.pdf
- http://ieeexplore.ieee.org/document/959216/