# Face Recognition System with Face Detection

*Report submitted in fulfillment of the requirements for the Exploratory Project of*

**Summer Workshop on Machine Learning and Deep Learning for Computer Vision**

*by*

**Debdatta Chatterjee**

*Under the guidance of*

**Dr. Tanima Dutta**

**Department of Computer Science and Engineering**

**INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI**
**Varanasi 221005, India**
**May 2018**

# Declaration

I declare that

1. The work contained in this report is original and has been done by the team and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date: 15/07/2018

Debdatta Chatterjee

# Certificate

*This is to certify that the work contained in this report entitled "**Face Recogntion System with Face Detection**" being submitted by Debdatta Chatterjee carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bonafide work of our supervision.*

;

**Dr. Tanima Dutta**
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005

Place: IIT (BHU) Varanasi
Date: 15/07/2018

# Abstract

The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. Identifying a person with an image has been popularized through the mass media. However, it is less robust to fingerprint or retina scanning. Face recognition is an important application of image processing owing to its use in many fields. The project presented here describes the face detection and recognition using Local Binary Pattern Histograms classifier on Yale face database. For face detection, Haar-Cascades were used and for face recognition Local Binary Pattern Histograms were used. The methodology is described including flow charts for each stage of the system. In the result section the confidence score is shown which is simply the measure of accuracy with which the system recognize the face. And the result comes up as the 4th individual is recognized with the system with maximum accuracy and the 8th individual is recognized with least accuracy. The report is concluded with the authors' opinion on the project. The area of this project face detection system with face recognition is Image processing. The software requirements for this project is Python software.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Overview

Face recognition has several advantages over other biometric modalities such as fingerprint and iris: besides being natural and nonintrusive, the most important advantage of face is that it can be captured at a distance and in a covert manner. Among the six biometric attributes considered by Hietmeyer, facial features scored the highest compatibility in a Machine Readable Travel Documents (MRTD) system based on a number of evaluation factors, such as enrollment, renewal, machine requirements, and public perception, Face recognition, as one of the major biometric technologies, has become increasingly important owing to rapid advances in image capture devices (surveillance cameras, camera in mobile phones), availability of huge amounts of face images on the Web, and increased demands for higher security.The first automated face recognition system was developed by Takeo Kanade in his Ph.D. thesis work in 1973. There was a dormant period in automatic face recognition until the work by Sirovich and Kirby on a low dimensional face representation, derived using the Karhunen–Loeve transform or Principal Component Analysis (PCA). It is the pioneering work of Turk and Pentland on Eigenface that reinvigorated face recognition research. Other major milestones in face recognition include: the Fisherface method which applied Linear Discriminant Analysis (LDA) after a PCA step to achieve higher accuracy; the use of local filters such as Gabor jets to provide more effective facial features; and the design of the AdaBoost learning based cascade classifier architecture for real time face detection .Face recognition technology is now significantly advanced since the time when the Eigenface method was proposed. In the constrained situations, for example
where lighting, pose, stand-off, facial wear, and facial expression can be controlled,automated face recognition can surpass human recognition performance, especially when the database (gallery) contains a large number of faces.1 However, automatic face recognition still faces many challenges when face images are acquired under unconstrained environments. In the following sections, we give a brief overview of the face recognition process, analyze technical challenges, propose possible solutions,and describe state-of-the-art performance.The following document is a report involved building a system for face detection and face recognition using Local Binary Pattern Histograms classifier, available in the open computer vision library (OpenCV). Face recognition is a non-invasive identification system and faster than other systems

since multiple faces can be analyzed at the same time. The difference between face detection and identification is, face detection is to identify a face from an image and locate the face. Face recognition is making the decision "whose face is it? ", using an image database. In this project both are accomplished using different techniques and are described below. The report begins with a Literature Survey and proposed work for this project. This is followed by the explanation of HAAR-cascades, Local binary pattern histogram (LBPH) algorithms. Next, the experiment methodology and the results of the project are described. Finally, a conclusion is provided.

## 1.2 Motivation of the Research Work-Literature Survey

The project on face recognition had helped the authors to have a detailed survey of LBPH classifier as Face recognition algorithm. Some of the important methods are studied will be described in this section. Face recognition systems architecture broadly consists of the three following tasks:

1. Acquisition (Detection, Tracking of face-like images)
2. Feature extraction (Segmentation, alignment & normalization of the face image)
3. Recognition

## 1.3 Proposed Work

We propose a face recognition system based on Harr- Cascade Classifier and Local Binary Pattern Histogram classifier. The main challenge for a face recognition system is of effective feature extraction. The proposed system utilizes the LBPH method for information reduction for the images. There is an incredible amount of information present even in a small face image. A method must be able to break down pictures so as to effectively represent face images rather than images in general. Base faces are generated and then image being analyzed can be represented by the system as a linear combination of these base faces. Each face that we wish to classify can be projected into face-space and then analyzed as a vector. An intermediate image is created to describe the original image in a better way by highlighting the facial characteristics.
The whole process can be divided in three major steps -

1. The first step is to find a good database of faces with multiple images for each individual.
2. The next step is to detect faces in the database images and use them to train the face recognizer.
3. The last step is to test the face recognizer to recognize faces it was trained for.

# Chapter 2

# A brief discussion on Face Recognition and Face Detection

## Face Detection Approach:

Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector.

Historically, working with only image intensities (i.e., the RGB pixel values at each and every pixel of image) made the task of feature calculation computationally expensive. A publication by Papageorgiou et al. discussed working with an alternate feature set based on Haar wavelets instead of the usual image intensities. Viola and Jones adapted the idea of using Haar wavelets and developed the so-called Haar-like features. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. For example, let us say we have an image database with human faces. It is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore a common Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object (the face in this case).

In the detection phase of the Viola–Jones object detection framework, a window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects. Because such a Haar-like feature is only a weak learner or classifier (its detection quality is slightly better than random guessing) a large number of Haar-like features are necessary to describe an object with sufficient accuracy. In the Viola–Jones object detection framework, the Haar-like features are therefore organized in something called a classifier cascade to form a strong learner or classifier.

The key advantage of a Haar-like feature over most other features is its calculation speed. Due to the use of integral images, a Haar-like feature of any size can be calculated in constant time (approximately 60 microprocessor instructions for a 2-rectangle feature).A Haar wavelet is a mathematical function that produces square-shaped waves with a beginning and an end and used to create box shaped patterns to recognize signals with sudden transformations. An example is shown in **figure 1**. By combining several wavelets, a cascade can be created that can identify edges, lines and circles with different colour intensities. These sets are used in Viola Jones face detection technique in 2001 and since then more patterns are introduced for object detection as shown in figure 1.To analyze an image using Haar-cascades, a scale is selected smaller than the target image. It is then placed on the image, and the average of the values of pixels in each section is taken. If the difference between two values, pass a given threshold, it is considered a match. Face detection on a human face is performed by matching a combination of different Haar-like-features. A single classifier is not accurate enough.

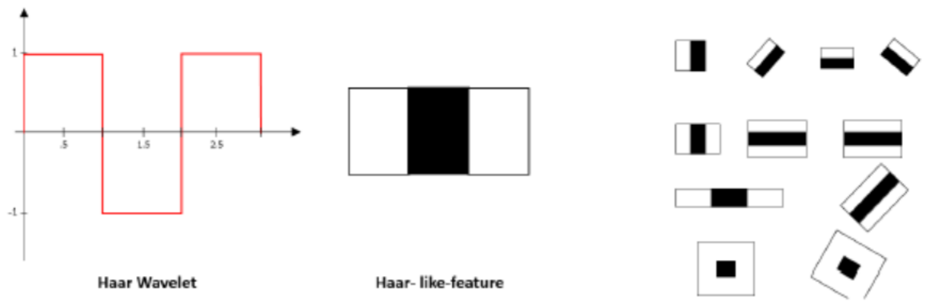


Figure 1: A Haar wavelet and resulting Haar-like features.

In this project, a similar method is used effectively to identify faces and eyes in combination resulting better face detection. Similarly, in Viola Jones method several classifiers were combined to create stronger classifiers. ADA boost is a machine learning algorithm that tests out several week classifiers on a selected location and choose the most suitable . It can also reverse the direction of the classifiers and get better results if necessary .[1]

Furthermore, Weight-update-steps can be updated only on misses to get better performance. The cascade is scaled by 1.25 and re-iterated in order to end different sized faces. Running the cascade on an image using conventional loops takes a large amount of computing power and time. Viola Jones used a summed area table (an integral image) to compute the matches fast. First developed in 1984 , it became popular after 2001 when Viola Jones implemented Haar-cascades for face detection. Using an integral image enables matching features with a single pass over the image.

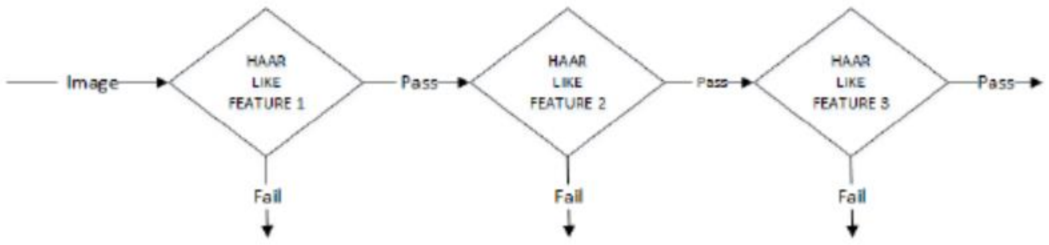


Figure 2: Haar-Cascade flow chart

## Face Recognition Approaches

The following sections describe the face recognition algorithm Local binary pattern histogram and how they are implemented in OpenCV.

### Local Binary Pattern Histogram: Introduction and Concept

Local binary patterns (LBP) is a type of visual descriptor used for classification in computer vision. LBP is the particular case of the Texture Spectrum model proposed in 1990. LBP was first described in 1994. It has since been found to be a powerful feature for texture classification; it has further been determined that when LBP is combined with the Histogram of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

A comparison of several improvements of the original LBP in the field of background subtraction was made in 2015 by Silva et al.A full survey of the different versions of LBP can be found in Bouwmans et al.

The LBP feature vector, in its simplest form, is created in the following manner:

- Divide the examined window into cells (e.g. 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
- Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
- Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.
- Optionally normalize the histogram.
- Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

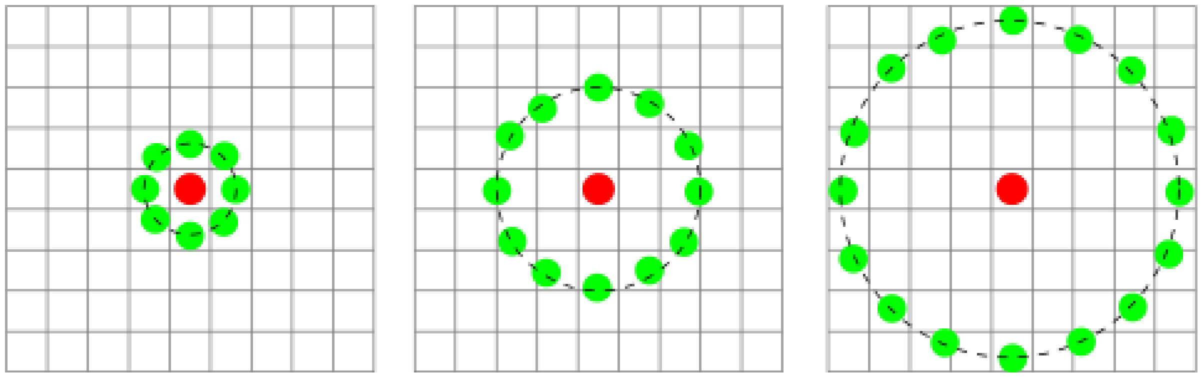


Figure 3: Three neighborhood examples to define a texture and display local binary pattern (LBP)

The feature vector can now be processed using the Support vector machine, extreme learning machines, or some other machine-learning algorithm to classify images. Such classifiers can be used for face recognition or texture analysis.

A useful extension to the original operator is the so-called uniform pattern[8], which can be used to reduce the length of the feature vector and implement a simple rotation invariant descriptor. This idea is motivated by the fact that some binary patterns occur more commonly in texture images than others. A local binary pattern is called uniform if the binary pattern contains at most two 0-1 or 1-0 transitions. For example, 00010000(2 transitions) is a uniform pattern, 01010100(6 transitions) is not. In the computation of the LBP histogram, the histogram has a separate bin for every uniform pattern, and all non-uniform patterns are assigned to a single bin. Using uniform patterns, the length of the feature vector for a single cell reduces from 256 to 59. The 58 uniform binary patterns correspond to the integers 0, 1, 2, 3, 4, 6, 7, 8, 12, 14, 15, 16, 24, 28, 30, 31, 32, 48, 56, 60, 62, 63, 64, 96, 112, 120, 124, 126, 127, 128, 129, 131, 135, 143, 159, 191, 192, 193, 195, 199, 207, 223, 224, 225, 227, 231, 239, 240, 241, 243, 247, 248, 249, 251, 252, 253, 254 and 255.

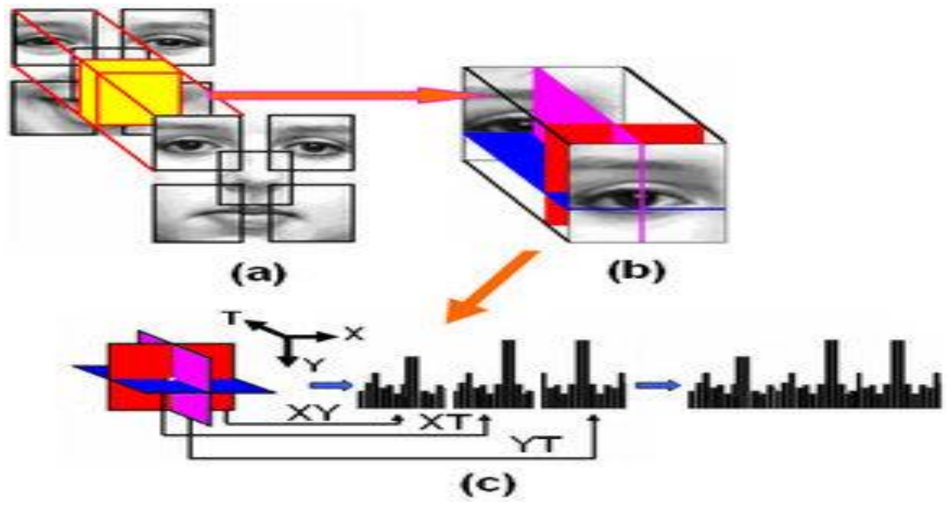


Figure 4: Description of facial expressions with local binary patterns

Using a clockwise or counter-clockwise direction surrounding pixel values are compared with the central as shown in figure 5. The value of intensity or luminosity of each neighbor is compared with the center pixel. Depending if the difference is higher or lower than 0, a 1 or a 0 is assigned to the location. The result provides an 8-bit value to the cell. The advantage of this technique is even if the luminosity of the image

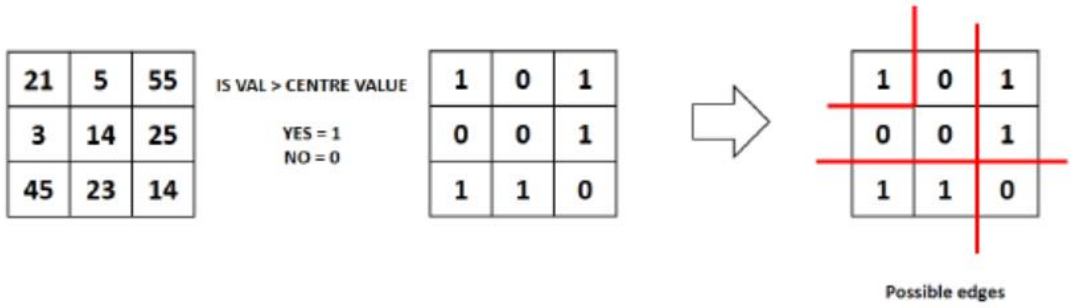


Figure 5: Local Binary Pattern Histogram Generating 8-bit Num

is changed as in figure 6, the result is the same as before. Histograms are used in larger cells to append the frequency of occurrences of values making process faster. By analyzing the results in the cell, edges can be detected as the values change. By computing the values of all cells and concatenating the histograms, feature vectors can be obtained. Images can be classified by processing with an ID attached. Input images are classified using the same process and compared with the dataset and distance is obtained. By setting up a threshold, it can be identified if it is a known or unknown face. Eigenface and Fisherface compute the dominant features of the whole training set while LBPH analyze them individually.

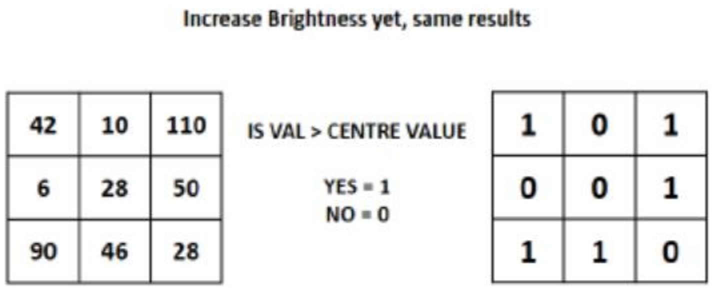


Figure 6: The results are same even if brightness is changed

We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **euclidean distance**, **chi-square**, **absolute value**, etc.

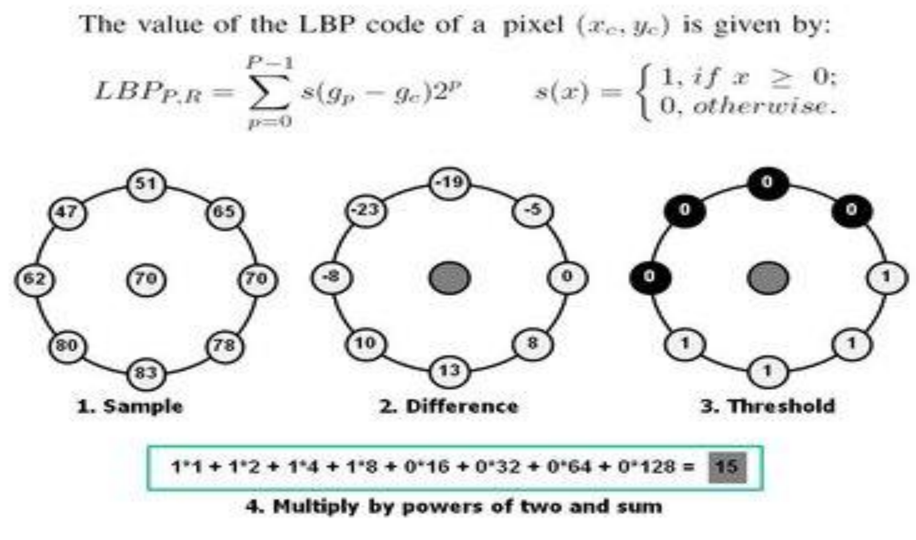
In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

LBP in Spatial Domain

The basic idea for developing the LBP operator was that two-dimensional surface textures can be described by two complementary measures: local spatial patterns and gray scale contrast. The original LBP operator (Ojala et al. 1996) forms labels for the image pixels by thresholding the 3 x 3 neighborhood of each pixel with the center value and considering the result as a binary number. The histogram of these 28 = 256 different labels can then be used as a texture descriptor.[2] This operator used jointly with a simple local contrast measure provided very good performance in unsupervised texture segmentation (Ojala and Pietikäinen 1999). After this, many related approaches have been developed for texture and color texture segmentation.

The LBP operator was extended to use neighborhoods of different sizes (Ojala et al. 2002). Using a circular neighborhood and bilinearly interpolating values at non-integer pixel coordinates allow any radius and number of pixels in the neighborhood. The gray scale variance of the local neighborhood can be used as the complementary contrast measure. In the following, the notation (P,R) will be used for pixel neighborhoods which means P sampling points on a circle of radius of R. See Fig. 2 for an example of LBP computation.


The value of the LBP code of a pixel $(x_c, y_c)$ is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \qquad s(x) = \begin{cases} 1, if\ x \geq 0; \\ 0, otherwise. \end{cases}$$

$$1{*}1 + 1{*}2 + 1{*}4 + 1{*}8 + 0{*}16 + 0{*}32 + 0{*}64 + 0{*}128 = \boxed{15}$$

Figure 7: An example of LBP computation.

Another extension to the original operator is the definition of so-called uniform patterns, which can be used to reduce the length of the feature vector and implement a simple rotation-invariant descriptor. This extension was inspired by the fact that some binary patterns occur more commonly in texture images than others. A local binary pattern is called uniform if the binary pattern contains at most two bitwise transitions from 0 to 1 or vice versa when the bit pattern is traversed circularly. For example, the patterns 00000000 (0 transitions), 01110000 (2 transitions) and 11001111 (2 transitions) are uniform whereas the patterns 11001001 (4 transitions) and 01010010 (6 transitions) are not. In the computation of the LBP labels, uniform patterns are used so that there is a separate label for each uniform pattern and all the non-uniform patterns are labeled with a single label. For example, when using (8,R) neighborhood, there are a total of 256 patterns, 58 of which are uniform, which yields in 59 different labels.

Ojala et al. (2002) noticed in their experiments with texture images that uniform patterns account for a little less than 90% of all patterns when using the (8,1) neighborhood and for around 70% in the (16,2) neighborhood. Each bin (LBP code) can be regarded as a micro-texton. Local primitives which are codified by these bins include different types of curved edges, spots, flat areas etc.

## Spatiotemporal LBP

The original LBP operator was defined to only deal with the spatial information. Later, it was extended to a spatiotemporal representation for dynamic texture analysis. For this purpose, the so-called Volume Local Binary Pattern (VLBP) operator was proposed (Zhao and Pietikäinen 2007). The idea behind VLBP consists of looking at dynamic texture as a set of volumes in the (X,Y,T) space where X and Y denote the spatial coordinates and T denotes the frame index (time). The neighborhood of each pixel is thus defined in three dimensional space.[3] Then, similarly to LBP in spatial domain, volume textons can be defined and extracted into histograms. Therefore, VLBP combines motion and appearance together to describe dynamic textures.

To make VLBP computationally simple and easy to extend, an operator based on co-occurrences of local binary patterns on three orthogonal planes (LBP-TOP) was also introduced. LBP-TOP considers three orthogonal planes: XY, XT and YT, and concatenates local binary pattern co-occurrence statistics in these three directions as shown in Figure 8. The circular neighborhoods are generalized to elliptical sampling to fit to the space-time statistics.

Fig. 8 shows example images from three planes. The XY plane represents appearance information, while the XT plane gives a visual impression of one row changing in time and YT describes the motion of one column in temporal space. The LBP codes are extracted for all pixels from the XY, XT and YT planes, denoted as XY-LBP, XT-LBP and YT-LBP, and histograms from these planes are computed and concatenated into a single histogram. In such a representation, a dynamic texture is encoded by an appearance (XY-LBP) and two spatial temporal (XT-LBP and YT-LBP) co-occurrence statistics.

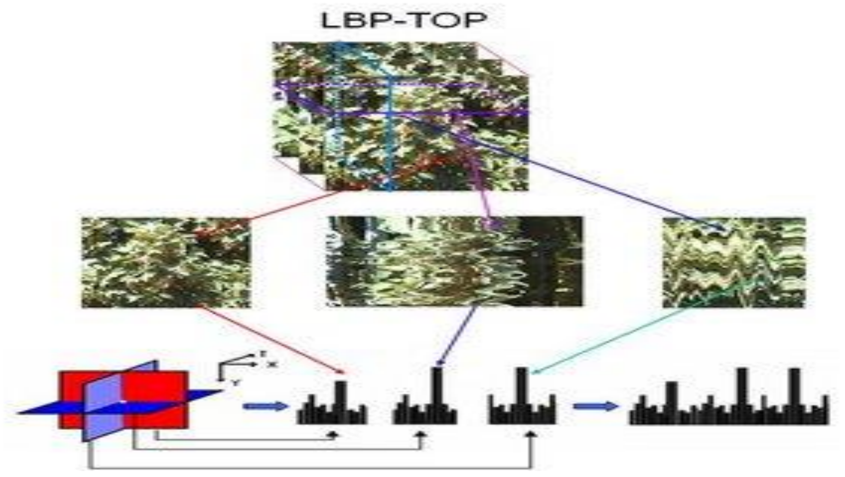


Figure 8: LBP from three orthogonal planes.

Setting the radius in the time axis to be equal to the radius in the space axis is not reasonable for dynamic textures. So we have different radius parameters in space and time to set. In the XT and YT planes, different radii can be assigned to sample neighboring points in space and time. More generally, the radii in axes X, Y and T, and the number of neighboring points in the XY, XT and YT planes can also be different.

## Face description using LBP

In the LBP approach for texture classification, the occurrences of the LBP codes in an image are collected into a histogram. The classification is then performed by computing simple histogram similarities.

However, considering a similar approach for facial image representation results in a loss of spatial information and therefore one should codify the texture information while retaining also their locations. One way to achieve this goal is to use the LBP texture descriptors to build several local descriptions of the face and combine them into a global description. Such local descriptions have been gaining interest lately which is understandable given the limitations of the holistic representations. [4]These local feature based methods are more robust against variations in pose or illumination than holistic methods.

The basic methodology for LBP based face description proposed by Ahonen et al. (2006) is as follows: The facial image is divided into local regions and LBP texture descriptors are extracted from each region independently. The descriptors are then concatenated to form a global description of the face, as shown in Fig. 4.

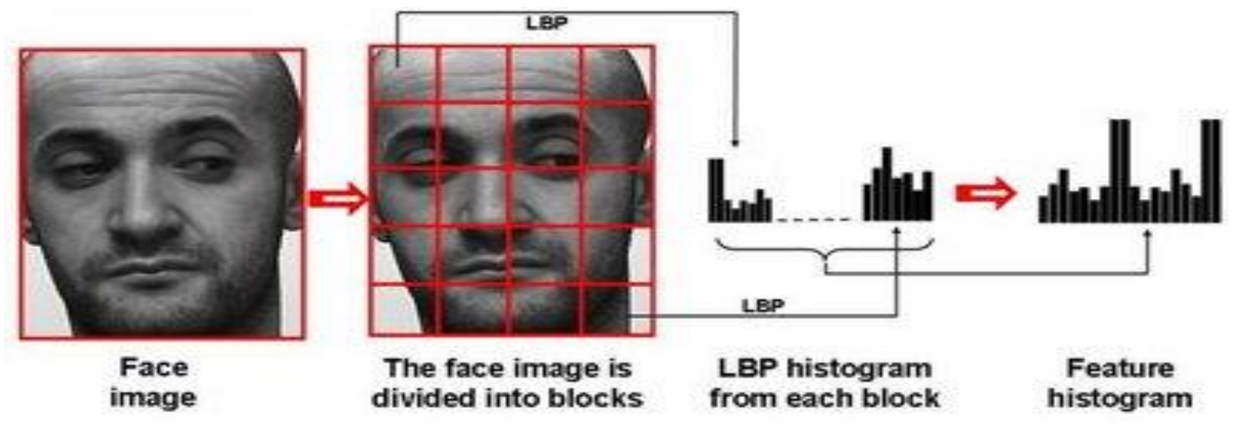

LBP
LBP
Face image
The face image is divided into blocks
LBP histogram from each block
Feature histogram

Figure 9: Face description with local binary patterns.

This histogram effectively has a description of the face on three different levels of locality: the LBP labels for the histogram contain information about the patterns on a pixel-level, the labels are summed over a small region to produce information on a regional level and the regional histograms are concatenated to build a global description of the face.[5]

It should be noted that when using the histogram based methods the regions do not need to be rectangular.Neither do they need to be of the same size or shape, and they do not necessarily have to cover the whole image. It is also possible to have partially overlapping regions.

The two-dimensional face description method has been extended into spatiotemporal domain (Zhao and Pietikäinen 2007).[6] Fig. 9 depicts facial expression description using LBP-TOP. Excellent facial expression recognition performance has been obtained with this approach.

Since the publication of the LBP based face description, the methodology has already attained an established position in face analysis research and applications.[7] A notable example is illumination-invariant face recognition system proposed by Li et al. (2007), combining NIR imaging with LBP features and Adaboost learning. Zhang et al. (2005) proposed the extraction of LBP features from images obtained by filtering a facial image with 40 Gabor filters of different scales and orientations, obtaining outstanding results. Hadid and Pietikäinen (2009) used spatiotemporal LBPs for face and gender recognition from video sequences, while Zhao et al. (2009) adopted the LBP-TOP approach to visual speech recognition achieving leading-edge performance without error-prone segmentation of moving lips.[8]

**T**he LBPH uses 4 parameters:

- **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.

- **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.

- **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

- **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

**Training the Algorithm**: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.[9]

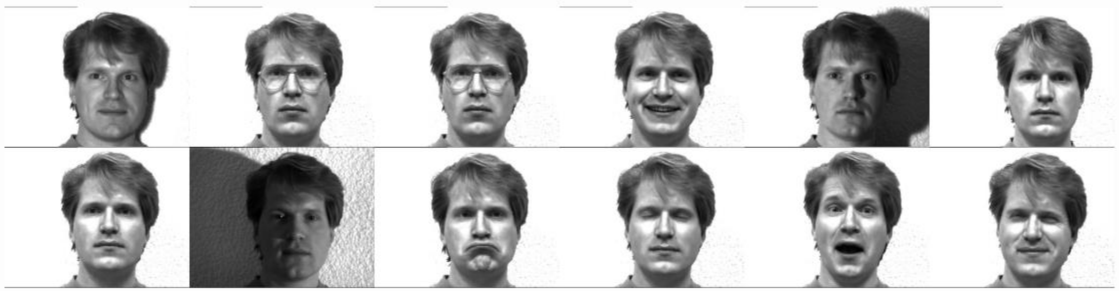
**Applying the LBP operation**: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**.

# Chapter 3

# Project Work

## Experiment

**Database:** We will use the Yale Face Database that contains 165 grayscale imagesof 15 individuals in gif format, There are 11 images for each individual. In each image, the individual has a different facial expression like happy, sad, normal, surprised, sleepy etc. Indeed, there are 166 images with 12 images for the first individual.



We will use this database by using 10 images of the total 11 images of each individual in training our face recognizer and the remaining single image of each individual to test our face recognition algorithm.The images corresponding to each individual are named like subject(number) facial expression(number) where subject numbers are 1,2,3,4,5,… and facial expressions are sad,happy etc

As I had mentioned earlier, there are 12 images for individual number 01. Out of the 11 images for each individual (12 for the first individual), we will not train the image with the **.sad** extension. We will use these images to test the face recognizer

## Implementation

We have performed the code for face recognition in an open-source cross platform integrated development environment(IDE) SPYDER.The first step of starting the coding is to import the required modules -

1. **cv2** - This is the OpenCV module and contains the functions for face detection and recognition.
2. **os -** This module will be used to maneuver with image and directory names. First, we will use this module to extract the image names in the database directory and then from these names we will extract the individual number, which will be used as a label for the face in that image.
3. **Image** - Since, the dataset images are in gif format and as of now, OpenCV does not support gif format, we will use Image module from PIL to read the image in grayscale format.
4. **numpy**- Our images will be stored in numpy arrays.

## Load the face detection Cascade

The first step is to detect the face in each image. Once, we get the region of interest containing the face in the image, we will use it for training the recognizer. For the purpose of face detection, we will use the Haar Cascade provided by OpenCV. . We will use haarcascade_frontalface_default.xml for detecting the face. So, we load the cascade using the cv2.CascadeClassifer function which takes the path to the cascade xml file. I have copied the xml file in the current working directory, so we have used the relative path.

## Create the Face Recognizer Object

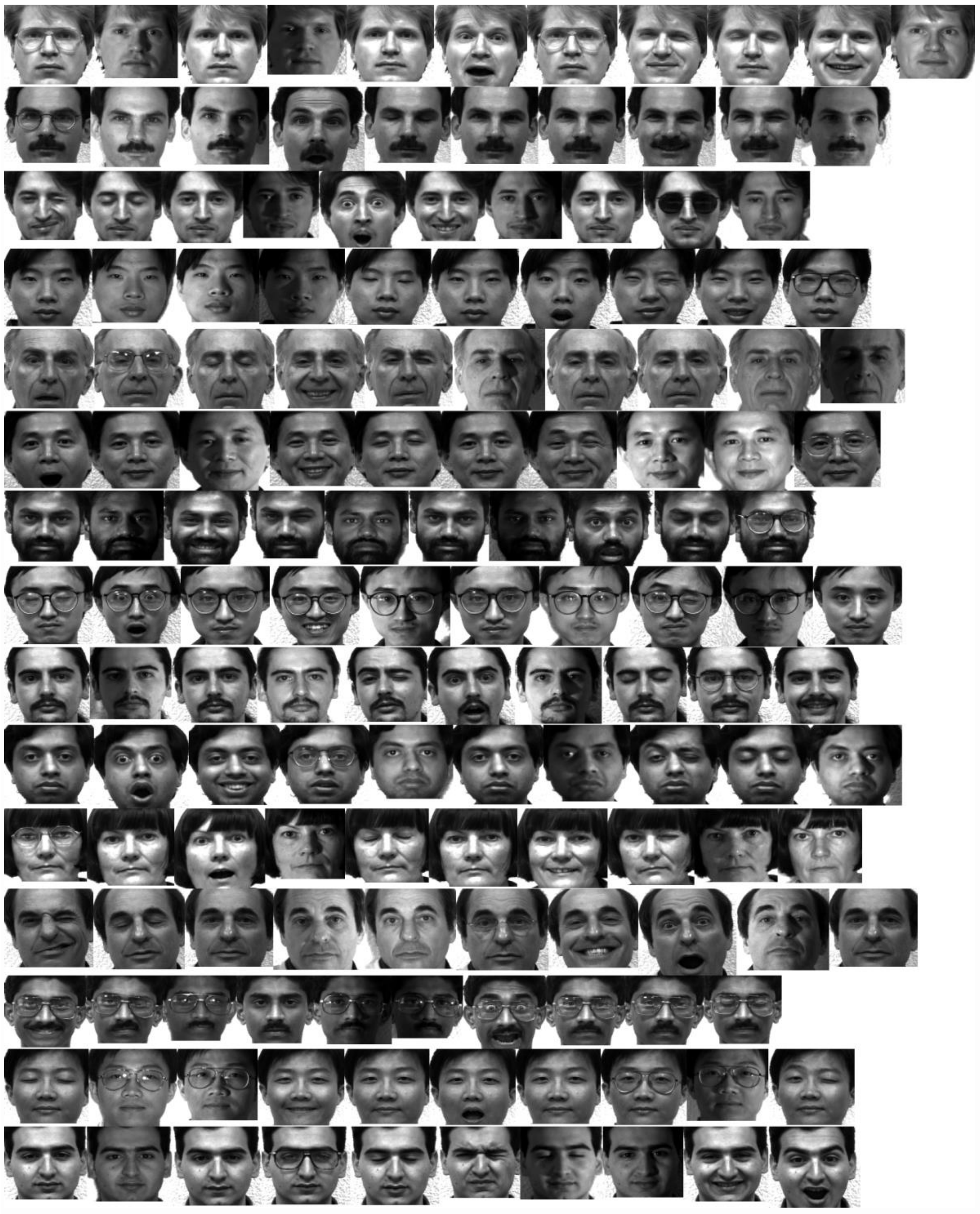
The next step is creating the face recognizer object. Local Binary Patterns Histograms Face Recognizer –createLBPHFaceRecognizer().We will use Local Binary Patterns Histograms Face Recognizer.

Now, we will define a function get_images_and_labels  that takes the absolute path to the image database as input argument and returns tuple of 2 list, one containing the detected faces and the other containing the corresponding label for that face. For example, if the ith index in the list of faces represents the 5th individual in the database, then the corresponding ith location in the list of labels has value equal to 5.

We use CascadeClassifier.detectMultiscale  to detect faces in the image. Although, in most cases, we need to tune the CascadeClassifier.detectMultiscale  function to correctly recognize faces in the image. The CascadeClassifier.detectMultiscale function returns a list of faces. For each face it returns a rectangle in the format *(Top-Left x pixel value*, *Top-Left y pixel value*, *Width of rectangle*, *Height of rectangle.)*. Once, we are done with this loop, we return the 2 lists in the form of a tuple.

## Preparing the training set

We pass the get_images_and_labels function with the path of the database directory. This path has to be the absolute path. This functions returns the features (images) and labels (labels) which will be used to train the face recognizer in the next step.The image below shows that detected faces for each individual. Row 1 contains the face images of individual 1, row 2 contains the face images of individual 2 and so on.



## Perform the training

We perform the training using the FaceRecognizer.train function. It requires 2 arguments, the features which in this case are the images of faces and the corresponding labels assigned to these faces which in this case are the individual number that we extracted from the image names.

## Testing the face recognizer

We will test the results of the recognizer by using the images with .**sad** extension which we had not used earlier. As done in the get_images_and_labels function, we append all the image names with the **.sad** extension in a image_paths list. Then for each image in the list, we read it in gray-scale format and detect faces in it.
Once, we have the ROI containing the faces, we pass the ROI to the FaceRecognizer.predict function which will assign it a label and it will also tell us how confident it is about the recognition. The label is an integer that is one of the individual numbers we had assigned to the faces earlier. This label is stored in nbr_predicted. The more the value of confidence variable is, the less the recognizer has confidence in the recognition. A confidence value of 0.0 is a perfect recognition.

## Results and Discussions

## What is confidence measure?

As in many other works (Lleida and Rose, 1996; Jiang, 2005), our first confidence measure for automatic face recognition is an estimate of the a poste-riori class probability. The output of our classifier is P(F |C), where C is the recognized face class and F represents the face features. The likelihoods P(F |C) are normalized to compute the a posteriori class prob-abilities as follows:

$$P(C|F) = \frac{P(F|C).P(C)}{\sum_{I \in \mathcal{FIM}} P(F|I).P(I)} \qquad (1)$$

F IM represents the set of all faces and P(C) denotes the prior probability of the face class C.
In the first version of our algorithm, called Absolute confidence value method, only the faces C so that

$$\hat{C} = \arg\max_{C}(P(C|F)) \qquad (2)$$

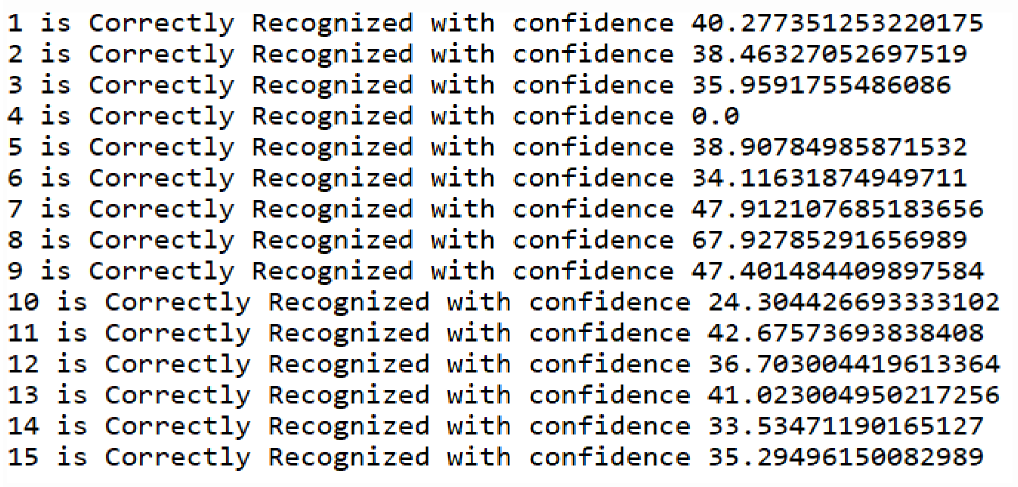
$$P(\hat{C}|F) > T \qquad (3)$$

are considered as recognized correctly.
In the second version of our approach, called *Relative confidence value* method, the difference be-tween the *best* hypothesis and the *second best* one is computed by the following equation:

$$P\Delta \quad = \quad P(\hat{C}|F) - \max_{C \neq \hat{C}}(P(C|F)) \qquad (4)$$

Only the faces with *P*□ > *T* are accepted. This second approach aims at identifying the faces that “dominate” all the other candidates, which is not al-ways well captured by the first method.
*T* is in the both cases an acceptation threshold and its optimal value is found experimentally

The training algorithm was tested by .sad extention images and the following result showed up for the 15 individuals.The fourth person is recognized by the system most accurately.8 th person is recognized least accurately.



```
1 is Correctly Recognized with confidence 40.277351253220175
2 is Correctly Recognized with confidence 38.46327052697519
3 is Correctly Recognized with confidence 35.9591755486086
4 is Correctly Recognized with confidence 0.0
5 is Correctly Recognized with confidence 38.90784985871532
6 is Correctly Recognized with confidence 34.11631874949711
7 is Correctly Recognized with confidence 47.912107685183656
8 is Correctly Recognized with confidence 67.92785291656989
9 is Correctly Recognized with confidence 47.401484409897584
10 is Correctly Recognized with confidence 24.304426693333102
11 is Correctly Recognized with confidence 42.67573693838408
12 is Correctly Recognized with confidence 36.703004419613364
13 is Correctly Recognized with confidence 41.023004950217256
14 is Correctly Recognized with confidence 33.53471190165127
15 is Correctly Recognized with confidence 35.29496150082989
```

# Chapter 4

# Conclusion and Discussion

In this project we performed the task of face recognition using OpenCV in less than 40 lines of python codes. The code assigns a label to each image that is to recognized. When, the face is not known by the face recognizer, the value of confidence score will be very high and we can use a threshold to ascertain that the face was not recognized. When the same face recognition system is applied over 100 testing images the accuracy decreased. There were more results coming up with low accuracy measure.Thirty subjects were recognized with zero confidence measure in case of 100 testing images.

# References

1. “Vallaste e-teatmik,” [Online]. Available: http://vallaste.ee/index.htm?Ty pe=UserId&otsing=5027. [Accessed October 2013]

2. Ojala, Timo, Matti Pietikainen, and Topi Maenpaa. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns.” *IEEE Transactions on pattern analysis and machine intelligence*24.7 (2002): 971–987.

3. Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. “Face recognition with local

4. binary patterns.” *Computer vision-eccv 2004* (2004): 469–481.

5. LBPH OpenCV: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms
6. Local Binary Patterns: http://www.scholarpedia.org/article/Local_Binary_Patterns

7. Lenc, Ladislav & Král, Pavel. (2011). Confidence measure for automatic face recognition. KDIR 2011 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval..

8. Ahonen, T., Hadid, A. and Pietikäinen, M. (2006), Face Description with Local Binary Patterns: Application to Face Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence 28(12):2037-2041.
9. Hadid, A. and Pietikäinen, M. (2009), Combining Appearance and Motion for Face and Gender Recognition from Videos. Pattern Recognition 42(11):2818-2827.