PROJECT REPORT

ON

Description of Interest Regions with Local Binary Patterns (LBP)

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

A Adithya

N Avinash Reddy

B Kushal Kumar

P Sagar

Under the esteemed guidance of

Prof G Mallikarjuna Rao,

Department of Computer Science and Engineering
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Affiliated to JNT University, Hyderabad)

Bachupally, Nizampet Road , Hyderabad -500072

CERTIFICATE

This is to certify that the project entitled "**Description of interest regions with Local Binary Patterns (LBP)**" has been submitted by

A Adithya	(07241A0557)
N Avinash Reddy	(07241A0564)
B Kushal Kumar	(07241A0579)
P Sagar	(07241A0597)

In partial fulfillment of the requirements for the award project in Bachelor of Technology in Computer science and Engineering from Jawaharlal Nehru Technological University, Hyderabad. The results embodied in this project have not been submitted to any other university or institution for the award of any degree or diploma.

(Signature)
Project Guide
Prof G Mallikarjuna Rao
Professor
Dept. of CSE

(Signature)
HOD
Dr. K Anuradha
Professor & HOD
Dept. of CSE

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our beloved Director Dr. P. S. Raju and Principal Dr. J. N. Murthy for their valuable guidance and for permitting us to carry out this project.

We are thankful to our HOD DR. K. Anuradha, who has shown keen interest in our project and encouraged us all along the course.

Our project guide, Prof. G Mallikarjuna Rao has supported us all along the project. His help and guidance assured that the project achieves all that we aimed for at the beginning without making any sacrifices in the features and in time. We convey our sincere thanks to Dr. Praveen Kumar who has shared the results and conveyed valuable suggestions.

We express our thanks to all those who contributed for the successful completion of the project work.

With gratitude,

A Adithya (07241A0557)
B Kushal Kumar (07241A0579)
N Avinash Reddy (07241A0564)
P Sagar (07241A0597)

Contents

- Software Development Life Cycle
  - Types of SDLC
    - Full SDLC and Partial SDLC
  - Developmental Model
    - Waterfall Model, V-Model and Spiral Model
  - Methodology used in developing a software product
  - Advantages
  - Limitations of waterfall model
- System Design
  - Introduction to UML
    - Visualization
    - Specification
    - Construction
    - Documentation
  - UML diagrams
    - Static or structural diagrams
    - Dynamic or behavioral diagrams
  - UML diagrams for LBP
    - Usecase diagram
    - Sequence diagrams
    - State-chart diagrams
    - Activity diagram

Figure Index

- .NET framework stack.
- Processing in MATLAB.
- Calculation in MATLAB.
- Comparison in MATLAB through Correlation.
- FERET Nomenclature.
- Testing strategy.
- Sample output for distorted image.
- Processing (step-by-step) using GUI.
    - ✓ Image Distortion.
    - ✓ LBP generation for distorted image.
    - ✓ Selecting the input distorted image folder.
    - ✓ Starting the process.
    - ✓ Calling MATLAB.
    - ✓ Retrieving and Displaying Results.
    - ✓ Output for distorted Image1.
    - ✓ Output for distorted Image2.
    - ✓ Output for distorted Image3.
    - ✓ Output for distorted Image4.
    - ✓ Output for distorted Image5.
    - ✓ Output for distorted Image6.

Abstract

The Local Binary Pattern (LBP) approach provides a neoteric method for Interest Region Description. LBP adopts the idea that the appearance of an interest region can be well characterized by the distribution of its local features. The most well-known descriptor built on this idea is the SIFT (Scale Invariant Feature Transform) descriptor that uses gradient as its local feature. Thus far, existing texture features are not widely utilized in the context of region description. Local Binary Pattern (LBP) is a typical algorithm, widely used in the texture analysis and classification domains. However it can be extended for other domains like Computer vision, Face Recognition, Background Subtraction, and Recognition of 3D textured surface. In this project "Description of Interest regions with Local Binary Patterns", it is aimed to develop face recognition system that is capable of recognizing the images based on local features of the facial image irrespective of changes in capturing environment. LBP has properties that favor its usage in the Interest Region Description such as tolerance against illumination changes and computational simplicity. These two properties of LBP help analyze the image textures in challenging real-time environments.

The package has been tested with FERET Database containing 8,665 images of 994 subjects. The database is well acquainted in the research community. The results are encouraging with 94% recognition rate.

Introduction

Face Recognition involves recognizing people with their intrinsic facial characteristics. Compared to other biometrics, such as fingerprint, DNA, or voice, face recognition is more natural, nonintrusive and can be used without the cooperation of the subject. Since the very first automatic system for face recognition there has been a growing attention given to face recognition. Due to powerful computers and recent advances in pattern recognition, face recognition systems can now perform in real-time and achieve satisfying performance under controlled conditions, leading to many potential applications.

A face recognition system can be used in two modes: verification (or authentication) and identification. A face verification system involves confirming or denying the identity claimed by a person (one-to-one matching). On the other hand, a face identification system attempts to establish the identity of a given person out of a pool of N people (one-to-N matching). When the identity of the person may not be in the database, this is called open set identification. While verification and identification often share the same classification algorithms, both modes target distinct applications. In verification mode, the main applications concern access control, such as computer or mobile device log-in, building gate control, digital multimedia data access. Over traditional security access systems, face verification has many advantages: the biometric signature cannot be stolen, lost or transmitted, like for ID card, token, badges or forgotten like passwords or PIN codes. In identification mode, potential applications mainly involve video surveillance (public places, restricted areas), information retrieval (police databases, multimedia data management) or human computer interaction.

- ## Need for Face Recognition

Face recognition has always been a serious challenging task under research. On one hand, its applications are very useful for personal verification and recognition and on the other hand, it has always been very difficult to implement due to all the different situations in which the human face can be found and other similar limitations and difficulties. Nevertheless, the approaches of the last few decades have been focusing on developing a system for face recognition. Due to the difficulty of the face recognition task, the number of techniques is large and diverse.

There are different ways to identify a person which would be classified into two categories.

- ❖ Biometric Methods.
- ❖ Non-biometric Methods.

Non-Biometric identification system uses serial numbers of person id and a password or an ID card or a system password. However these ID's could be forged or taken

by other people, which cannot be identified by the system and hence cannot verify the validity of the card holder.

Biometric Recognition systems such as IRIS, Finger print, Voice recognition, Palm geometry and Facial features to identify a person have high accuracy in Recognition. However IRIS and finger print scanning are intrusive identification systems that need the cooperation of users and the equipments are really quite expensive.

On a contrary The Face Recognition system has cameras installed at a surveillance place which can capture all the objects in real time without even being noticed at times. The pause and declare interaction is one of the best method for bank transactions and security areas; people feel conscious of it, as well as comfortable and safe with it.

Humans mostly use faces to recognize individuals, advances in computing technology now enables similar recognition automatically. Early face Recognition System used simple geometric models but the recognition process has now matured into a science of sophisticated mathematical representation and matching process.

- ## Problem Statement

Although face detection receives considerable attention, it still remains a difficult pattern recognition task, because of the high variability of the face appearance. Faces are non-rigid, dynamic objects with a large diversity in shape, color and texture, due to multiple factors such as head pose, lighting conditions (contrast, shadows), facial expressions, occlusions (glasses) and other facial features (make-up, beard). Large variability in face appearance also affects face verification. Furthermore the variations between the images of the same face due to illumination and viewing direction are almost always larger than the image variation due to change in face identity. Another difficulty comes from the lack of reference images to trained face templates. In real-life applications, the enrolment procedure has to be fast and is generally done once. The few available training data are usually not enough to cover the intra-personal variability of the face. Moreover a significant mismatch between training and testing conditions may happen (especially lighting). Finally, the verification performance is highly related to the quality of the face localization step.

- ## Face Recognition Techniques

The applications for face recognition are very varied. We can divide them into two big groups, the applications that required face identification and the ones that need face verification. The difference is that the first one uses a face to match with other one on a database; on the other hand, the verification technique tries to verify a human face from a given sample of that face. Face recognition could be also divided into two different groups, according to their field of application. The main reason for promoting this technique is law enforcement application; however, it can also be used for commercial application. Among the law enforcement applications, some representative examples are mug shot albums, video surveillance and shoplifting. Concerning commercial applications we can

differentiate between entertainment (video games, virtual reality and training programs), smart cards (driver's license, passport and voter registration) and information security (TV parental control, cell phone and database security).

The various algorithmic approaches used in the system include

- PCA- Principal Component Analysis.
- LBP- Local Binary Patterns
- LFA- Local Feature Analysis.
- LDA- Linear Discriminant Analysis.
- ICA- Independent Component Analysis.
- EBGM- Elastic Graph Matching.
- NN- Neural Networks.
- AAM-Active Appearance Model.
- SVM- Support Vector Machine.
- HMM- Hidden Markov Models etc.

The common or most general face recognition approaches include

- Acquiring a sample
- Extracting features
- Comparison Templates
- Declaring a match

• Overview of popular approaches commonly used

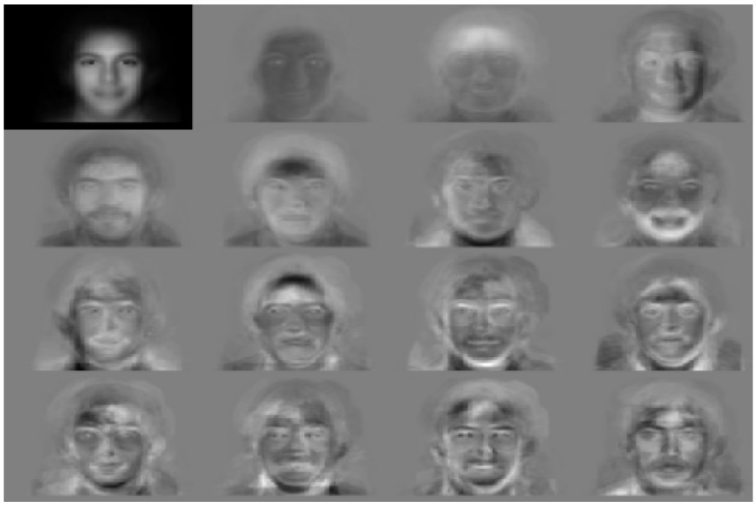
This section provides a brief overview of popular approaches most widely used and the next section provides the importance of Local Binary Patterns in face recognition.

PCA-Principal Component Analysis

This is most commonly referred to as the use of Eigen-faces, is the technique pioneered by Kirby and Sirivich in 1988. With PCA, the probe and gallery images must be the same size and must first be normalized to line up the eyes and mouth of the subjects within the images.

The PCA approach is then used to reduce the dimension of the data by means of data compression basics and reveals the most effective low dimensional structure of facial patterns. This reduction in dimensions removes information that is not useful and precisely decomposes the face structure into orthogonal (uncorrelated) components known as Eigen faces. Each face image may be represented as a weighted sum (feature vector) of the Eigen faces, which are stored in a 1D array. A probe image is compared against a gallery image by measuring the distance

between their respective feature vectors. The PCA approach typically requires the full frontal face to be presented each time; otherwise the image results in poor performance.



Standard Eigenfaces: Feature vectors are derived using eigenfaces.

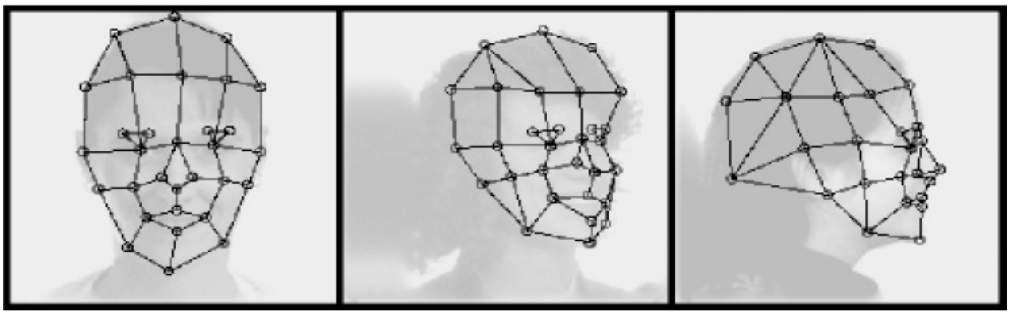
LDA: Linear Discriminant Analysis

LDA is a statistical approach for classifying samples of unknown classes based on training samples with known classes. This technique aims to maximize between-class (i.e., across users) variance and minimize within-class (i.e., within user) variance. In Figure where each block represents a class, there are large variances between classes, but little variance within classes. When dealing with high dimensional face data, this technique faces the small sample size problem that arises where there are a small number of available training samples compared to the dimensionality of the sample space.

Example of Six Classes Using LDA

EBGM: Elastic Bunch Graph Matching

EBGM relies on the concept that real face images have many non-linear characteristics that are not addressed by the linear analysis methods discussed earlier, such as variations in illumination (outdoor lighting vs. indoor fluorescents), pose (standing straight vs. leaning over) and expression (smile vs. frown). A Gabor wavelet transform creates a dynamic link architecture that projects the face onto an elastic grid. The Gabor jet is a node on the elastic grid, notated by circles on the image below, which describes the image behavior around a given pixel. It is the result of a convolution of the image with a Gabor filter, which is used to detect shapes and to extract features using image processing. [A convolution expresses the amount of overlap from functions, blending the functions together.] Recognition is based on the similarity of the Gabor filter response at each Gabor node. This biologically-based method using Gabor filters is a process executed in the visual cortex of higher mammals. The difficulty with this method is the requirement of accurate landmark localization, which can sometimes be achieved by combining PCA and LDA methods.



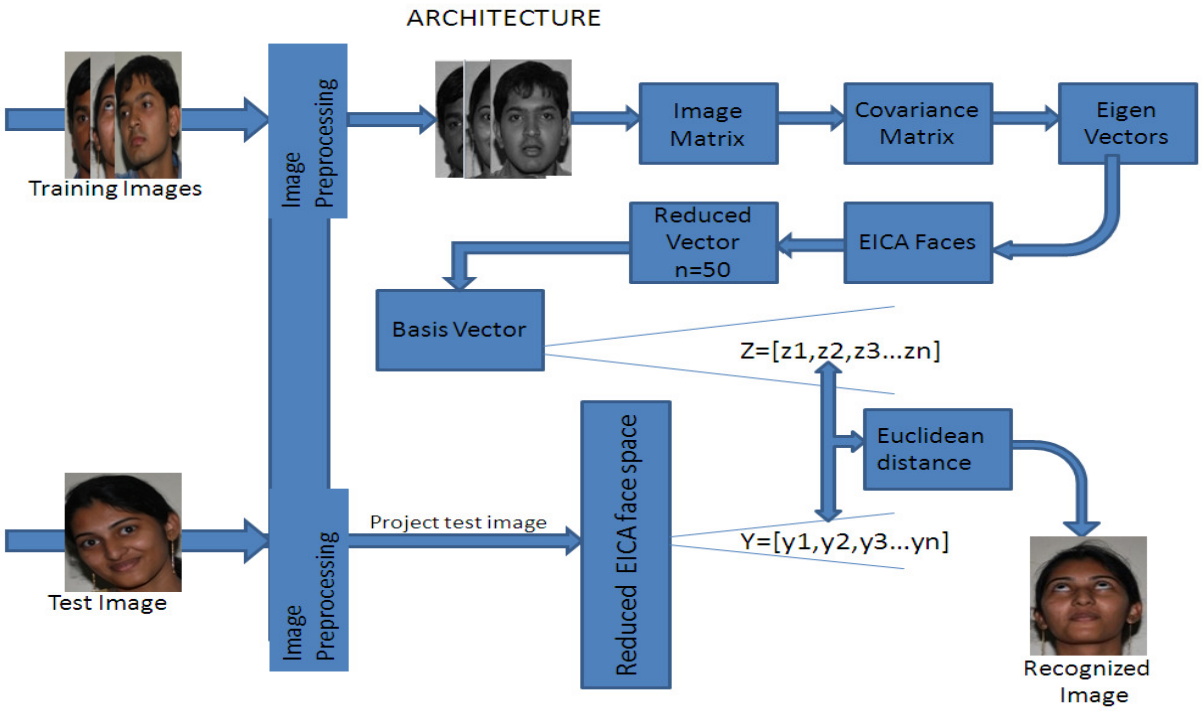
Elastic Bunch Map Graphing

ICA: Independent Component Analysis

ICA searches for a linear transformation to express a set of random variables as linear combinations of statistically independent source variables. The search criterion involves the minimization of the mutual information expressed as a function of high order cumulants. While PCA considers the 2nd order moments and it de-correlates data, ICA would further reduce statistical dependencies and produce a sparse and independent code useful for subsequent pattern discrimination and associative recall. The role ICA plays is to seek non-accidental and sparse feature codes analogous to the goal of sensory systems and to detect redundant features and to form a representation in which these redundancies are reduced and the

independent features and objects are represented explicitly. ICA thus provides a more powerful data representation method than PCA does.

Recent research on ICA-based face recognition leads to conflicting results. Bartlett et al implemented the info-max algorithm proposed by Bell and Sejnowski and applied it to face recognition. The independent components are derived using a neural network method from the point of view of optimal information transfer in neural networks with sigmoidal transfer functions. Their results based on a FERET data set show that ICA representations are superior to PCA representations for face recognition. Moghaddam implemented ICA for face recognition using a FERET data set. His results, however, suggest that ICA does not provide a systematic advantage over PCA for face recognition.

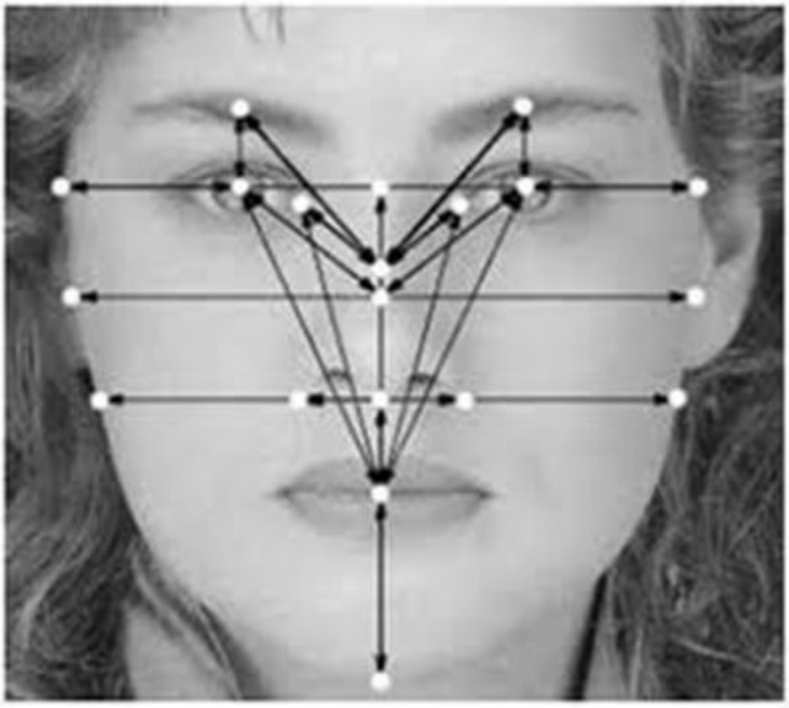


Independent Component Analysis.

LFA: Local Feature Analysis

Low-dimensional representations of sensory signals are keys to solving many of the computational problems encountered in high-level vision. Principal component analysis (PCA) has been used in the past to derive practically useful compact representations for different classes of objects. One major objection to the applicability of PCA is that it invariably leads to global, non-topographic representations that are not amenable to further processing and are not biologically plausible. A new mathematical construction, local feature analysis (LFA), for deriving local topographic representations for any class of objects is introduced. The

LFA representations are sparse-distributed and, hence, are effectively low-dimensional and retain all the advantages of the compact representations of the PCA. But, unlike the global Eigen modes, they give a description of objects in terms of statistically derived local features and their positions. The theory is illustrated by using it to extract local features for three ensembles: 2D images of faces without background, 3D surfaces of human heads, and finally 2D faces on a background. The resulting local representations have powerful applications in head segmentation and face recognition.



Local Feature Analysis.

NN: Neural Network based Face Recognition

The network will receive the 960 real values as a 960-pixel input image (Image size ~ 32 x 30). It will then be required to identify the face by responding with a 94-element output vector.

The 94 elements of the output vector each represent a face. To operate correctly the network should respond with a 1 in the position of the face being presented to the network, all other values in the output vector should be 0.

In addition, the network should be able to handle noise. In practice the network will not receive a perfect image of face which represented by vector as input. Specifically, the network should make as few mistakes as possible when classifying images with noise of mean 0 and standard deviation of 0.2 or less.

The neural network needs 960 inputs and 94 neurons in its output layer to identify the faces. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values.

The hidden layer has 200 neurons. This number was picked by guesswork and experience. If the network has trouble learning, then neurons can be added to this layer.

The network is trained to output a 1 in the correct position of the output vector and to fill the rest of the output vector with 0's. However, noisy input images may result in the network not creating perfect 1's and 0's. After the network has been trained the output will be passed through the competitive transfer function. This function makes sure that the output corresponding to the face most like the noisy input image takes on a value of 1 and all others have a value of 0. The result of this post-processing is the output that is actually used.

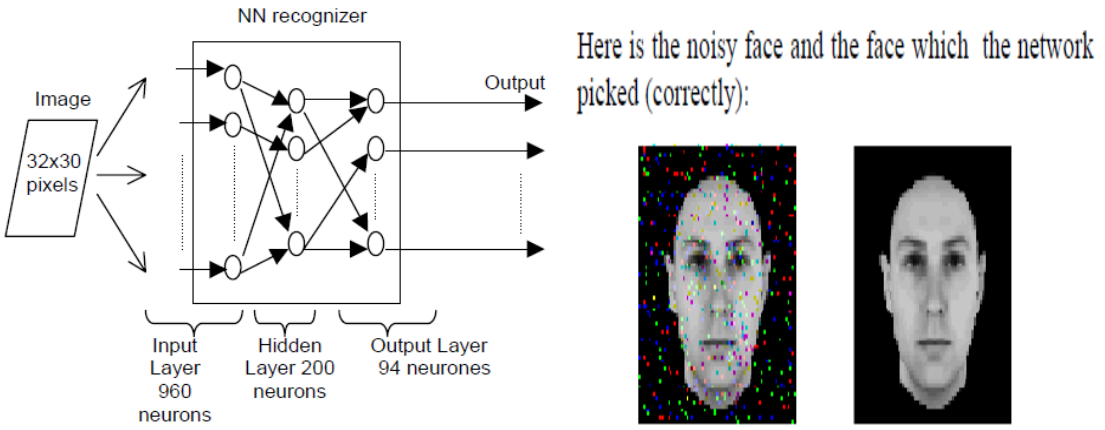
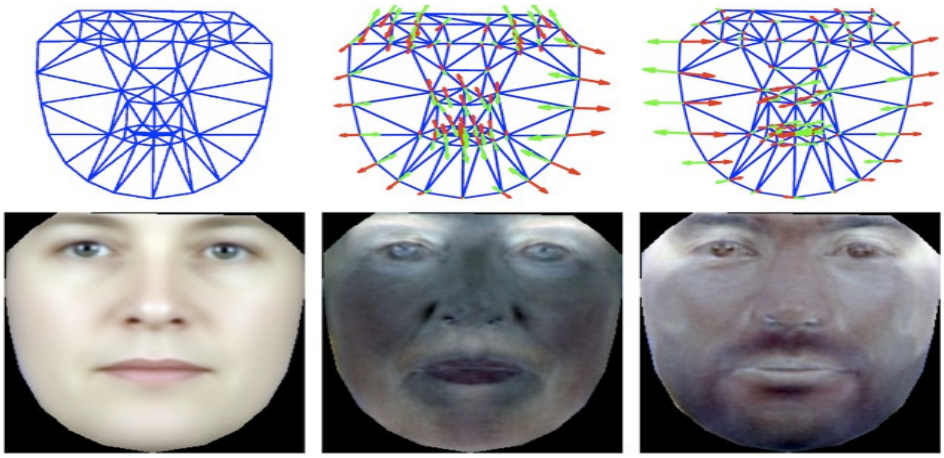


Figure 1. Architecture of neural network

AAM: Active Appearance Model

In many practical applications an AAM alone is insufficient. A suitable initialization is required for the matching process and when the AAM unconstrained may not always converge to the correct solution. The appearance

model provides shape and texture information which are combined to generate a model instance. A natural approach to initialization and constraint is to provide prior estimates of the position of some of the shape points, either manually or using automatic feature detectors. For instance, when matching a face model it is useful to have an estimate of the positions of the eyes, which could either be provided by a user or located using a suitable eye detector.



Active Appearance Model.

The original least squares matching of the AAM search algorithm reformulated into the statistical framework. This allows the introduction of prior probabilities on the model parameters and the inclusion of prior constraints on point positions. The latter allows one or more points to be pinned down to particular positions with a given variance. This framework enables the AAM to be integrated with other feature location tools in a principled manner, as long as those tools can provide an estimate of the error on their output.

TT: Trace Transform method of Face recognition

The Trace transform is a generalization of the Radon transform, is a new tool for image processing which can be used for recognizing objects under transformations, e.g. rotation, translation and scaling. To produce the Trace transform one computes a functional along tracing lines of an image. Each line is characterized by two parameters, namely its distance from the centre of the axes and the orientation the normal to the line has with respect to the reference direction. In addition, we define parameter along the line with its origin at the foot of the normal. The image is transformed to another image with the Trace transform which is a 2-D function depending on parameters. Different Trace Transforms can

be produced from an image using different trace functionals. One of the key properties of the Trace transform is that it can be used to construct features invariant to rotation, translation and scaling. We should point out that invariance to rotation and scaling is harder to achieve than invariance to translation. Let us assume that an object is subjected to linear distortions, i.e. rotation, translation and scaling. It is equivalent to saying that the image remains the same but viewed from a linearly distorted coordinate system.

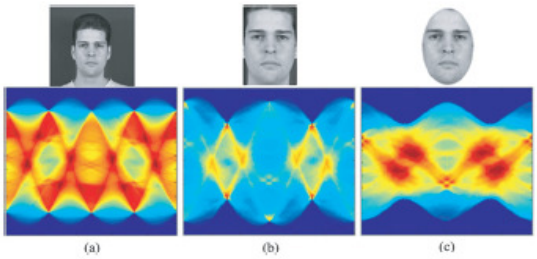


Fig. 4. Examples of the Trace transform. (a) full image (b) rectangular shape and (c) elliptical shape.

- ## Role of Local Binary Patterns in face recognition

Human faces are not an invariant characteristic; in fact, a person's face can change very much during short periods of time (from one day to another) and because of long periods of time (a difference of months or years). One problem of face recognition is the fact that different faces could seem very similar; therefore, a discrimination task is needed. On the other hand, when we analyze the same face, many characteristics may have changed. Ones of the most important problems are changes in illumination, variability in facial expressions, the presence of accessories (glasses, beards, etc); finally, the rotation of a face may change many facial characteristics.

The Local Binary Pattern (LBP) approach provides a neoteric method for Interest Region Description. This adopts the idea that the appearance of an interest region can be well characterized by the distribution of its local features. Thus far, existing texture features are not widely utilized in the context of region description. Local Binary Pattern (LBP) is a typical texture analysis algorithm, widely used in the texture analysis and classification domains. LBP has been highly successful for various computer vision problems such as Face Recognition, Background Subtraction, and Recognition of 3D textured surfaces, but it has not been used for the description of interest regions so far. Thus this project implements a well-known local binary pattern (LBP) feature. LBP has properties that favor its usage in the Interest Region Description such as tolerance against illumination changes and computational simplicity. These two properties of LBP help analyze the image textures in challenging real-time environments.

Local Binary Patterns

The local binary pattern descriptor was originally designed for texture description. The operator assigns a label to every pixel of the image by considering the neighborhood of each pixel and thresholding the values based on the central pixel value and thus obtaining the result as a binary number. Thus the values obtained are used as texture descriptors. The ability of Local Binary Patterns can be extended by using neighborhoods of different sizes to make LBP scalable.

Defining the local neighborhood as a set of sampling points evenly spaced on a circle centered at a pixel to be labeled allows any radius and number of sampling points. Bilinear interpolation is used when a sampling point does not fall into the center of the pixel.

- ## Interest Region Description

Our interest region descriptor is based on the SIFT descriptor which has shown to give excellent results. The basic idea is that the appearance of an interest region can be well characterized by the distribution of its local features. In order to incorporate spatial information into the representation, the region is divided into cells and for each cell a feature histogram is accumulated. The final representation is achieved by concatenating the histograms over the cells and normalizing the resulting descriptor vector. The major difference between the proposed descriptor and the SIFT descriptor is that they rely on different local features. Instead of the gradient magnitude and orientation used by the SIFT, we use novel local binary pattern (LBP) features. We give a brief review of the SIFT descriptor and the LBP operator.

SIFT and LBP

SIFT Descriptor

The SIFT descriptor is a 3D histogram of gradient locations and orientations. Location is quantized into a 4×4 location grid and the gradient angle is quantized into 8 orientations, resulting in a 128-dimensional descriptor. First, the gradient magnitudes and orientations are computed within the interest region. The gradient magnitudes are then weighted with a Gaussian window overlaid over the region. To avoid boundary effects in the presence of small shifts of the interest region, a tri-linear interpolation is used to distribute the value of each gradient sample into adjacent histogram bins. The final descriptor is obtained by concatenating the orientation histograms over all locations. To reduce the effects of illumination change the descriptor is first normalized to unit length. Then, the influence of large gradient magnitudes is reduced by thresholding the descriptor entries, such that each one is no larger than 0.2, and renormalizing to unit length.

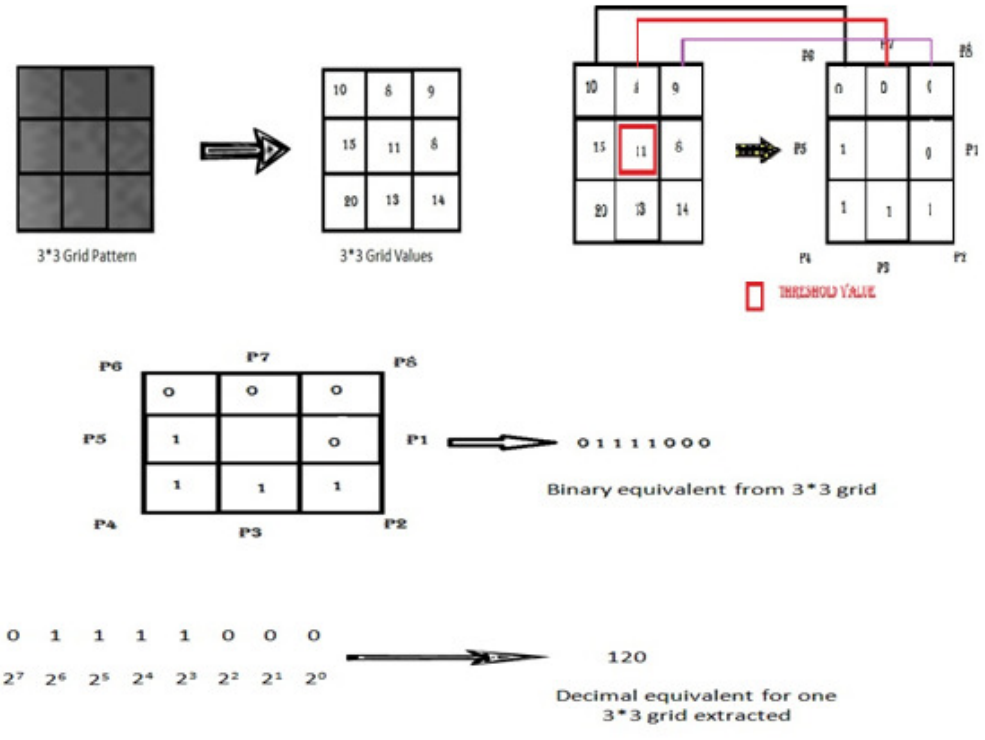
LBP Operator

The local binary pattern is a powerful gray level invariant texture primitive. The histogram of the binary patterns computed over a region is used for texture description. The operator describes each pixel by the relative gray levels of its neighboring pixels, given below is an illustration with 8 neighbors. If the gray level of the neighboring pixel is higher or equal, the value is set to one, otherwise to zero. The descriptor describes the result over the neighborhood as a binary number.

$$LBP_{R,N}(x, y) = \sum_{i=0}^{N-1} s(ni - nc) * 2^i \ , \qquad s(x) = 1 \text{ if } x \geq 0$$

$$= 0 \text{ otherwise}$$

Where n_c corresponds to the gray level of the center pixel of a local neighborhood and n_i to the gray levels of N equally spaced pixels on a circle of radius R. The values of neighbors that do not fall exactly on pixels are estimated by bilinear interpolation. Since correlation between pixels decreases with distance, a lot of the texture information can be obtained from local neighborhoods. Thus, the radius R is usually kept small. In practice the above given equation means that the signs of the differences in a neighborhood are interpreted as an N-bit binary number, resulting in 2N distinct values for the binary pattern. The LBP has several properties that favor its usage in interest region description. The features are robust against illumination changes, they are very fast to compute, do not require many parameters to be set, and have high discriminative power.



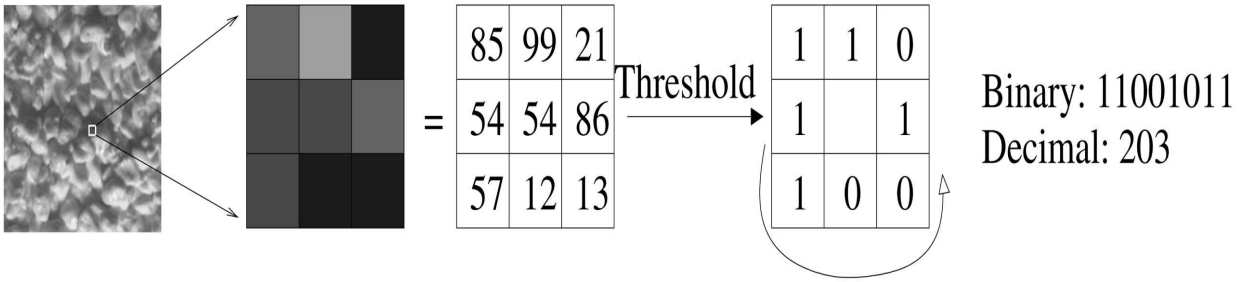
As automatic face analysis which includes, e.g., face detection, face recognition and facial expression recognition has become a very active topic in computer vision research, thus a key issue in face analysis is finding efficient descriptors for face appearance. Different holistic methods such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and the more recent 2-D PCA have been studied widely but lately also local descriptors have gained attention due to their robustness to challenges such as pose and illumination changes.

One of the first face descriptors based on information extracted from local regions is the Eigen-features method proposed by Pentland et al. This is a hybrid approach in which the features are obtained by performing PCA to local face regions independently. In Local Feature Analysis, kernels of local spatial support are used to extract information about local facial components. Elastic Bunch Graph Matching (EBGM) describes faces using Gabor filter responses in certain facial landmarks and a graph describing the spatial relations of these landmarks. The validity of the component based approach is also attested by the study conducted by Heisele et al. in which a component-based face recognition system clearly outperformed global approaches on a test database containing faces rotated in depth.

Using local photometric features for object recognition in the more general context has become a widely accepted approach. In that setting the typical approach is to detect interest points or interest regions in images, perform normalization with respect to affine transformations and describe the normalized interest regions using local descriptors. This bag-of-key points approach is not suited for face description as such since it does not retain information on the spatial setting of the detected local regions but it does bear certain similarities to local feature based face description.

Finding good descriptors for the appearance of local facial regions is an open issue. Ideally, these descriptors should be easy to compute and have high extra-class variance (i.e., between different persons in the case of face recognition) and low intra-class variance, which means that the descriptor should be robust with respect to aging of the subjects, alternating illumination and other factors.

The texture analysis community has developed a variety of different descriptors for the appearance of image patches. However, face recognition problem has not been associated to that progress in texture analysis field as it has not been investigated from such point of view.



- ## LBP Based Face Description

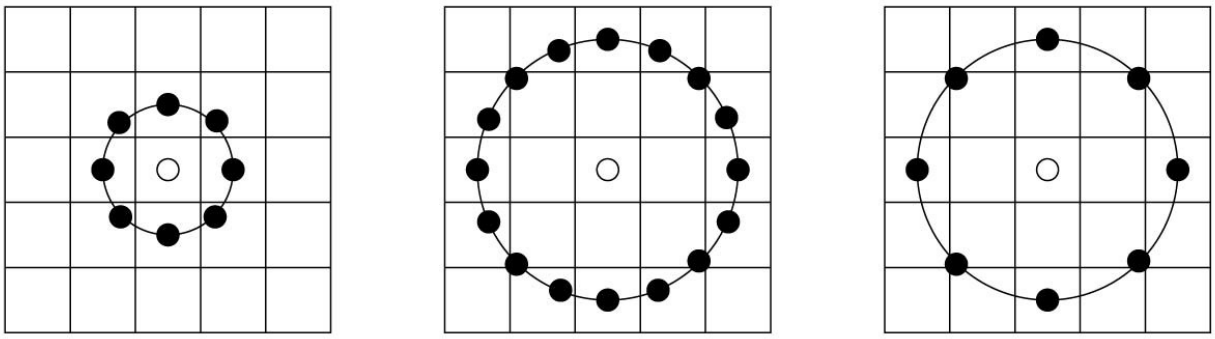
The LBP operator is one of the best performing texture descriptors and it has been widely used in various applications. It has proven to be highly discriminative and its key advantages, namely its invariance to monotonic gray level changes and computational efficiency, make it suitable for demanding image analysis tasks. The idea of using LBP for face description is motivated by the fact that faces can be seen as a composition of micro-patterns which are well described by such operator.

Local binary patterns

The LBP operator was originally designed for texture description. The operator assigns a label to every pixel of an image by thresholding the 3x3-neighborhood of each pixel with the center pixel value and considering the result as a binary number. Then the histogram of the labels can be used as a texture descriptor.

To be able to deal with textures at different scales, the LBP operator was later extended to use neighborhoods of different sizes. Defining the local neighborhood as a set of sampling points evenly spaced on a circle centered at the pixel to be labeled allows any radius and number of sampling points. Bilinear interpolation is used when a sampling point does not fall in the center of a pixel. In the following, the notation (P, R) will be used for pixel neighborhoods which mean P-sampling points on a circle of radius of R. Another extension to the original operator is the definition of so called uniform patterns.

A local binary pattern is called uniform if the binary pattern contains at most two bitwise



Circular (8, 1), (16, 2) and (8, 2) neighborhoods

The pixel values are bi-linearly interpolated whenever the sampling point is not in the center of a pixel. Transition from 0 to 1 or vice versa in the bit pattern is considered circular. For example, the patterns 00000000 (0 transitions), 01110000

(2 transitions) and 11001111 (2 transitions) are uniform whereas the patterns 11001001 (4 transitions) and 01010011 (6 transitions) are not.

In this work, the LBP method presented in the previous subsection is used for face description. The procedure consists of using the texture descriptor to build several local descriptions of the face and combining them into a global description. Instead of striving for a holistic description this approach was motivated by two reasons: the local feature based or hybrid approaches to face recognition have been gaining interest lately which is understandable given the limitations of the holistic representations. The local feature based and hybrid-methods seem to be more robust against variations in pose or illumination than holistic methods.

Another reason for selecting the local feature based approach is that trying to build a holistic description of a face using texture methods is not reasonable since texture descriptors tend to average over the image area. This is a desirable property for ordinary textures, because texture description should usually be invariant to translation or even rotation of the texture and, especially for small repetitive textures, the small-scale relationships determine the appearance of the texture and thus the large-scale relations do not contain useful information. For faces however, the situation is different: retaining the information about spatial relations is important.

A facial image divided into 7*7, 5*5 and 3*3 rectangular regions.

This reasoning leads to the basic methodology of this work. The facial image is divided into local regions and texture descriptors are extracted from each region independently. The descriptors are then concatenated to form a global description of the face.

- **Rotational Invariance**

As the first step toward gray-scale invariance, we subtract without losing information, the gray value of the center pixel from the gray values of the circularly symmetric neighborhood $g_p(p=0,\ldots., p-1)$. Next, we assume that differences g_p-g_c is independent of g_c, which allows us to factorize. The factorized distribution is only an approximation of the joint distribution. We neglect small loss in information as it allows us to achieve invariance with respect to shifts in gray scale. The distribution $t(g_c)$ describes the over-luminance of the image, which is unrelated to local image texture and, consequently, does not provide useful information for texture analysis. Hence, much of the information in the original joint gray level distribution about the textural characteristics is conveyed by the joint difference distribution .This is a highly discriminative texture operator. It records the occurrences of various patterns in the neighborhood of each pixel in a P-dimensional histogram. For constant regions, the differences are zero in all directions. On a slowly sloped edge, the operator records the highest difference in the gradient direction and zero values along the edge and, for a spot, the differences are high in all directions. Signed differences g_p-g_c is not affected by changes in mean luminance; hence, the joint difference distribution is invariant against gray-scale shifts. We achieve invariance with respect to the scaling of the gray scale by considering just the signs of the differences instead of their exact values:

$$T = t\left(s(g_0 - g_c), s(g_0 - g_c), \ldots\ldots\ldots, s(g_0 - g_c)\right)$$

Where s(x) =1, if x>=0 and s(x) =0, if x<0.

By assigning a binomial factor 2^p for each sign s (g_p-g_c), we transform the above expression into a unique $LBP_{P,R}$ number that characterizes the spatial structure of the local image texture:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)\,2^p$$

The name "Local Binary Pattern" reflects the functionality of the operator, i.e., a local neighborhood is threshold at the gray value of the center pixel into binary pattern. $LBP_{P,R}$ operator is by definition invariant against any monotonic transformation of the gray scale i.e., as long as the order of the gray values in the image stays the same, the output of the $LBP_{P,R}$ operator remains constant. If we set (P =8; R =1), we obtain $LBP_{8,1}$. The two differences between $LBP_{8,1}$ and LBP are: 1) The pixels in the neighbor set are indexed so that they form a circular chain and 2) the gray values of the diagonal pixels are determined by interpolation. Both modifications are necessary to obtain the circularly symmetric neighbor set, which allows for deriving a rotation invariant version of $LBP_{P,R}$. B. Achieving Rotation Invariance The $LBP_{P,R}$ operator produces 2^p different output values, corresponding to the 2^p different binary patterns that can be formed by the P pixels in the neighbor set. When the image is rotated, the gray values g_p will correspondingly move along the perimeter of the circle around g_0. Since g_0 is always assigned to be the gray value of element (0, R) to the right of gc rotating a particular binary pattern naturally results in a different $LBP_{P,R}$ value. This does not apply to patterns comprising of only 0s (or 1s) which remain constant at all rotation angles. To remove the effect of rotation, i.e., to assign a unique identifier to each rotation invariant local binary pattern we define:

$$LBP^{ri}_{P,R} = min\{ROR(LBP_{P,R}, i) \mid i = 0,1,\ldots,P-1\}$$

Where ROR (x, i) performs a circular bit-wise right shift on the P-bit number x i times. In terms of image pixels, simply corresponds to rotating the neighbor set clockwise so many times that a maximal number of the most significant bits, starting from g_{p-1}, is 0. $LBP^{ri}_{P,R}$ quantifies the occurrence statistics of individual rotation invariant patterns corresponding to certain micro-features in the image; hence, the patterns can be considered as feature detectors.

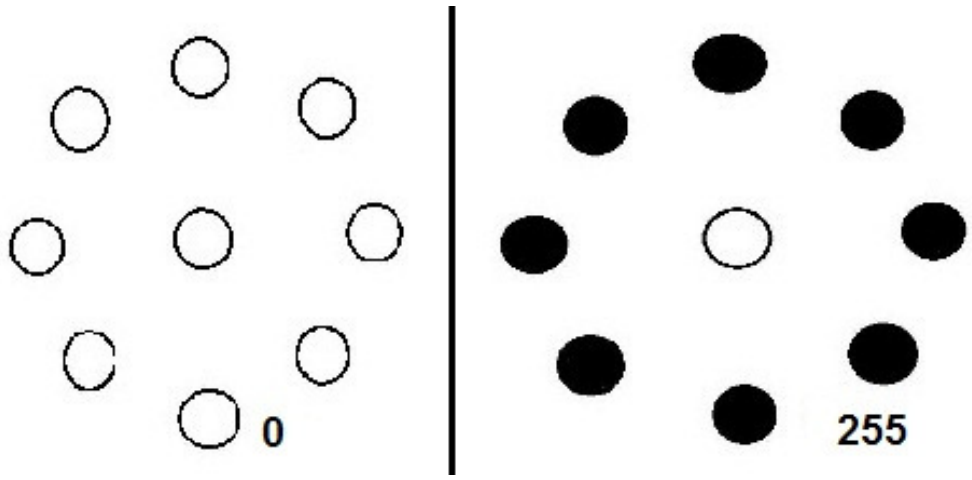
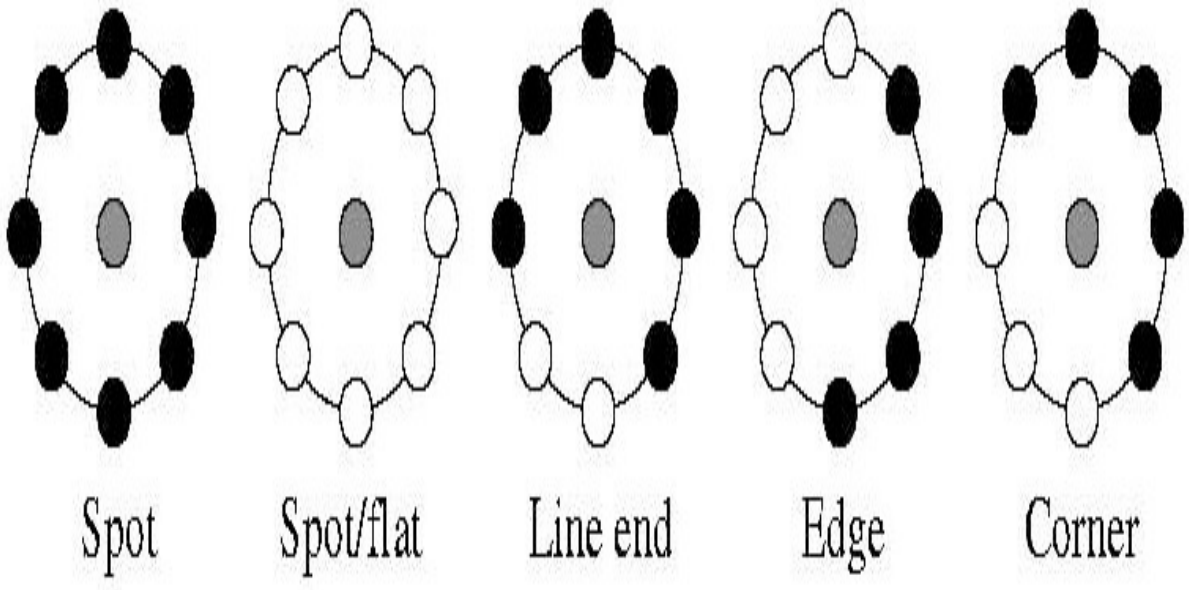


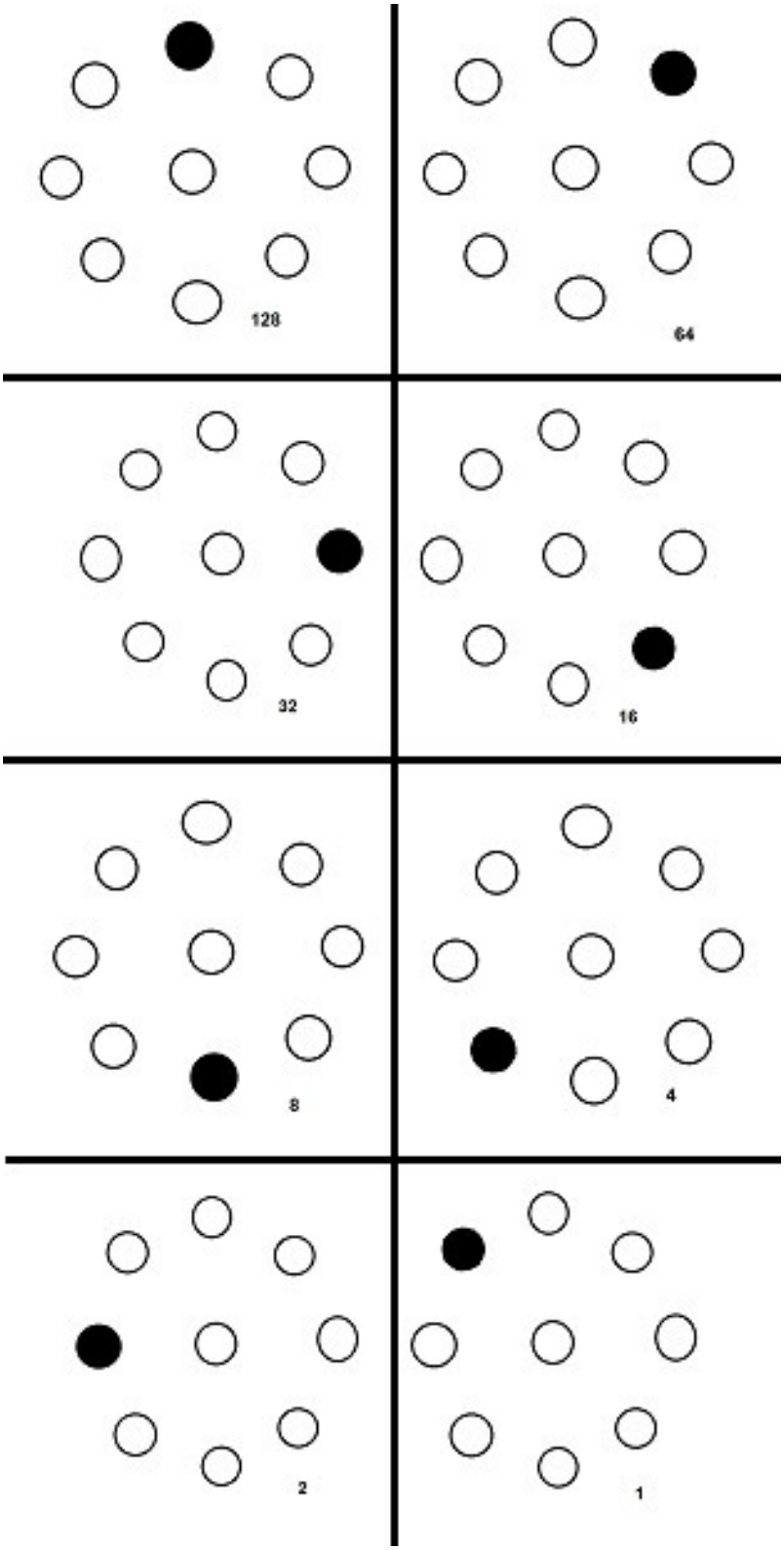
Figure: Neighborhood of radius=1 unit.

With all zeroes surrounding the central pixel also known as “SPOT”

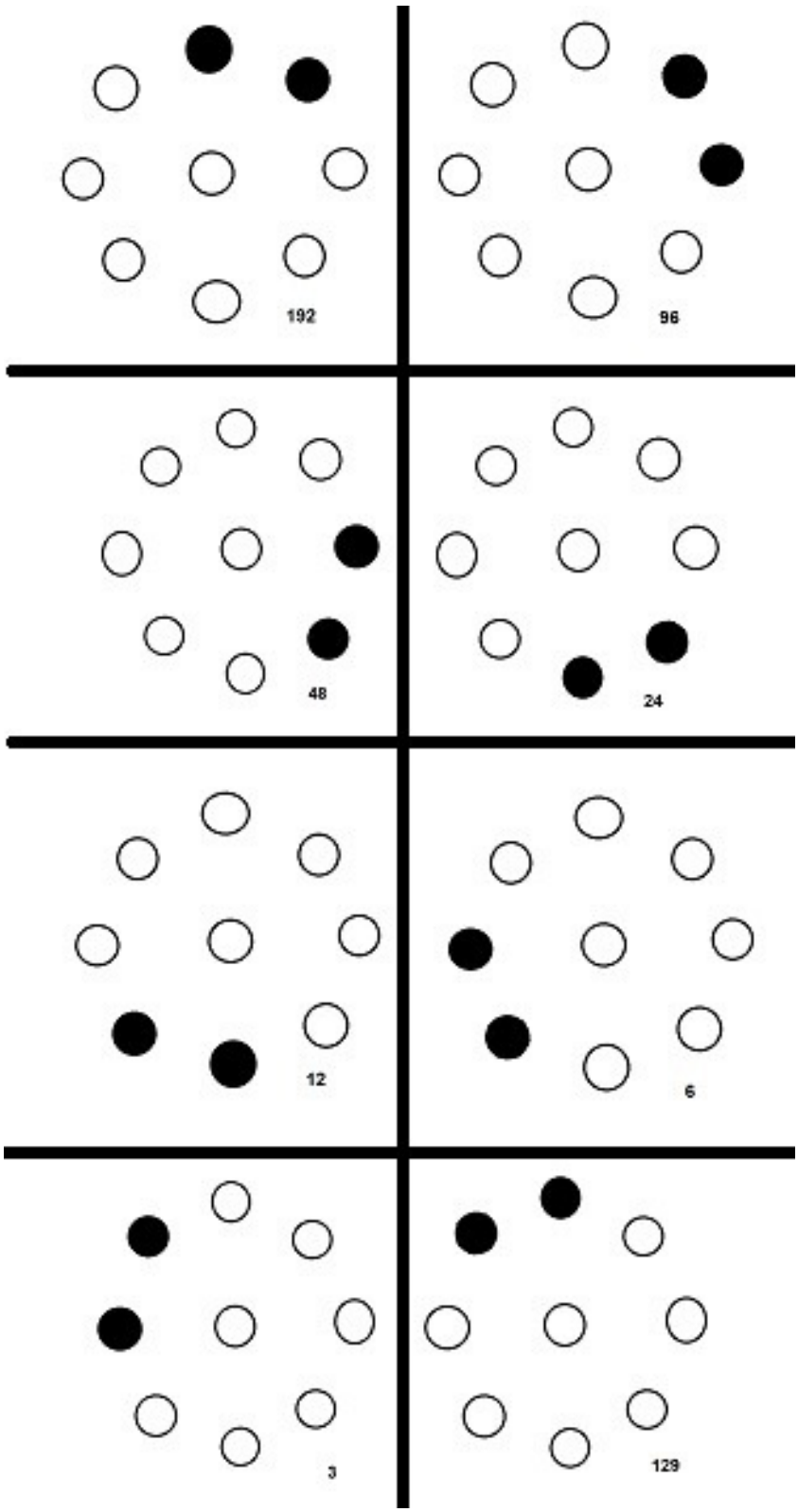
With all ones surrounding the central pixel also known as “FLAT”



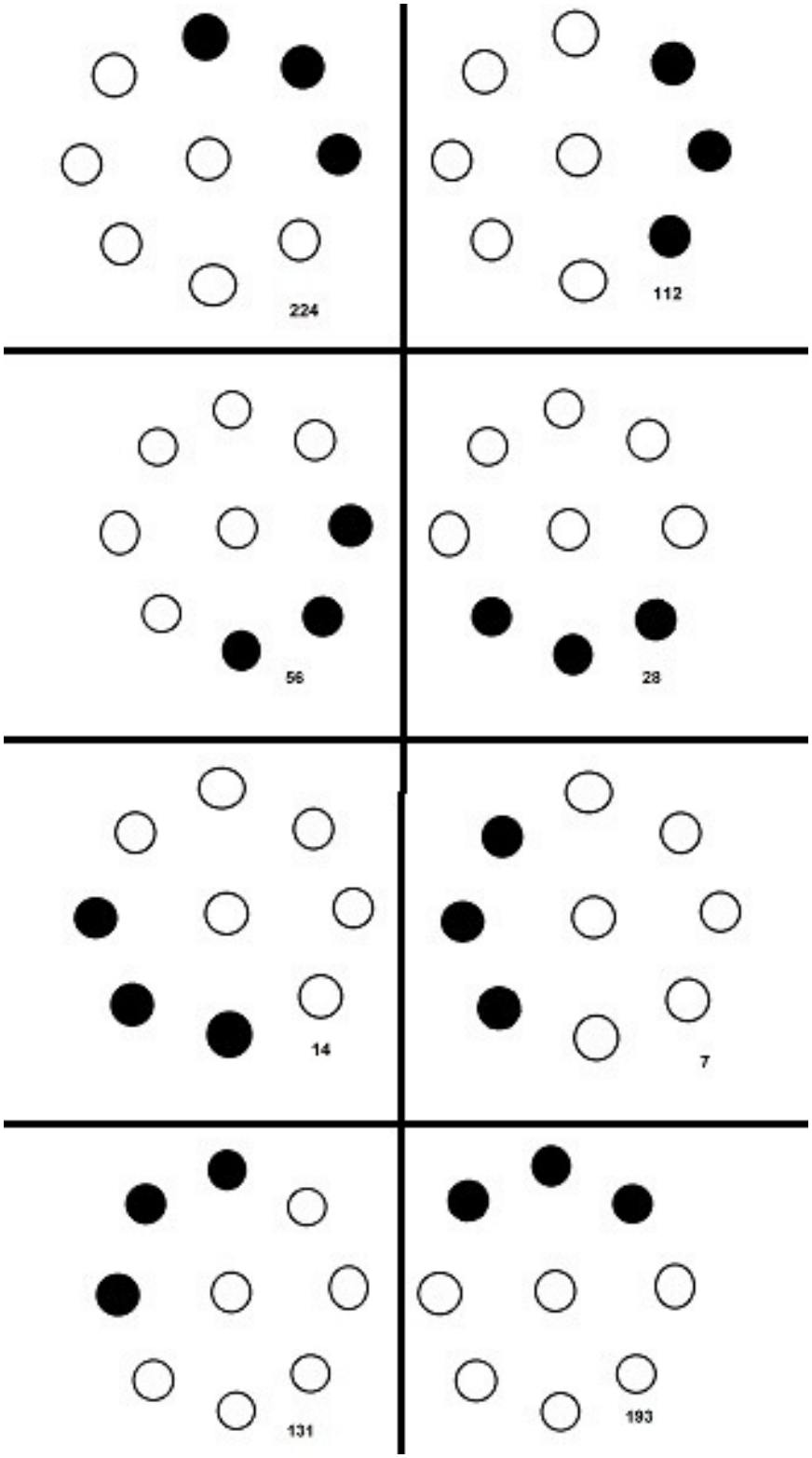
Common Patterns Detected by LBP approach.



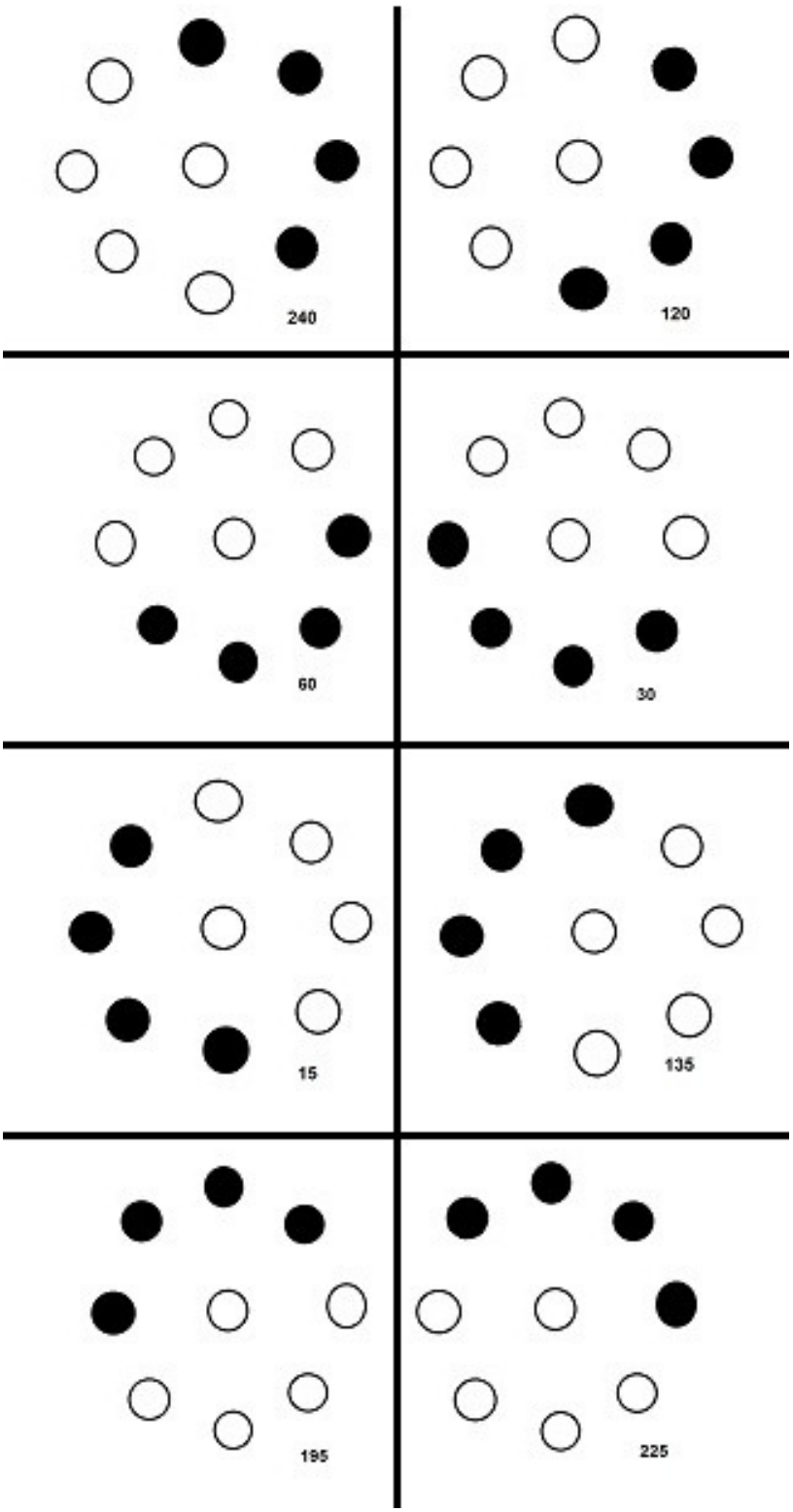
LBP for one pixel highlighted (with their corresponding decimal equivalent values).



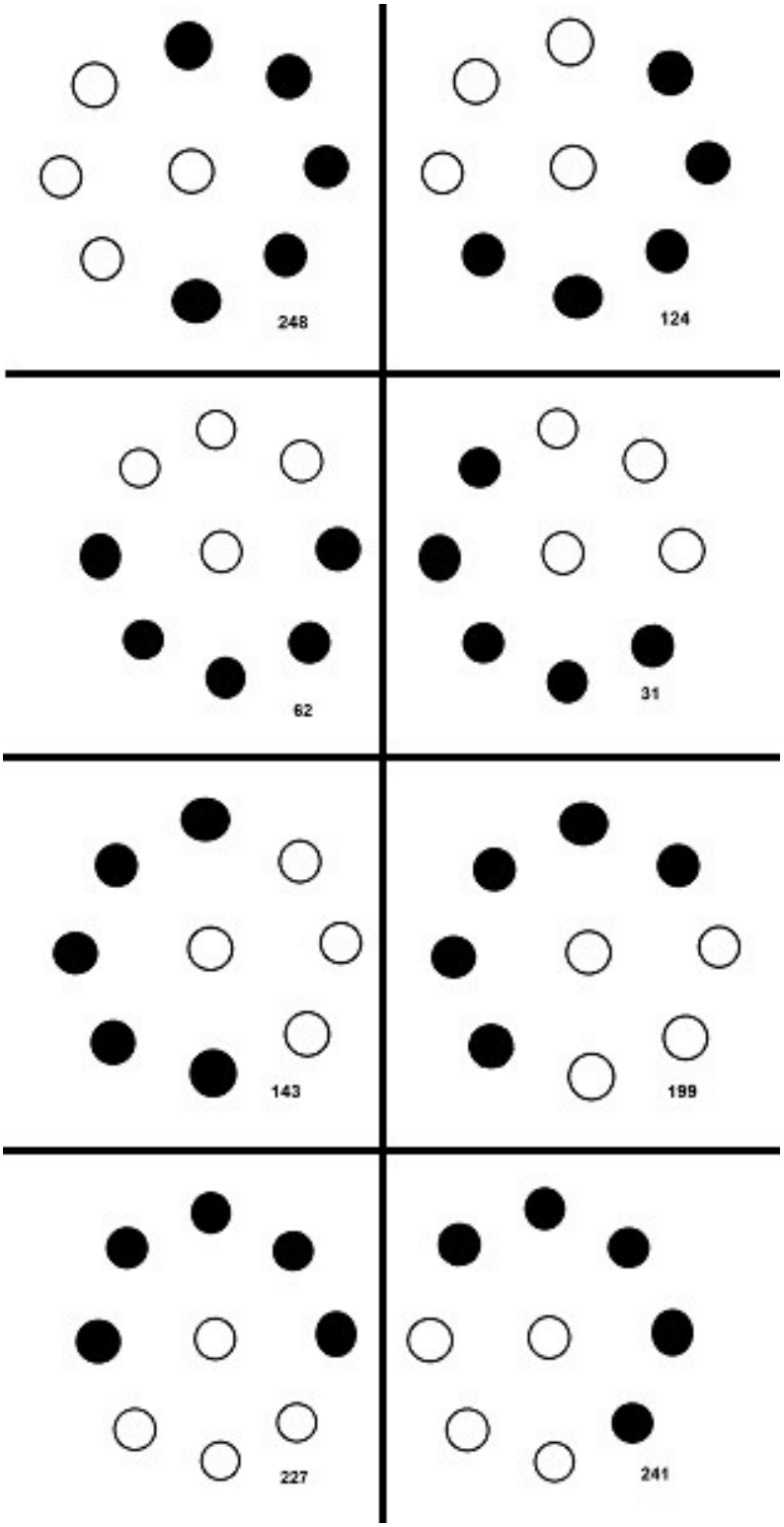
LBP for two pixels highlighted (with their corresponding decimal equivalent values).



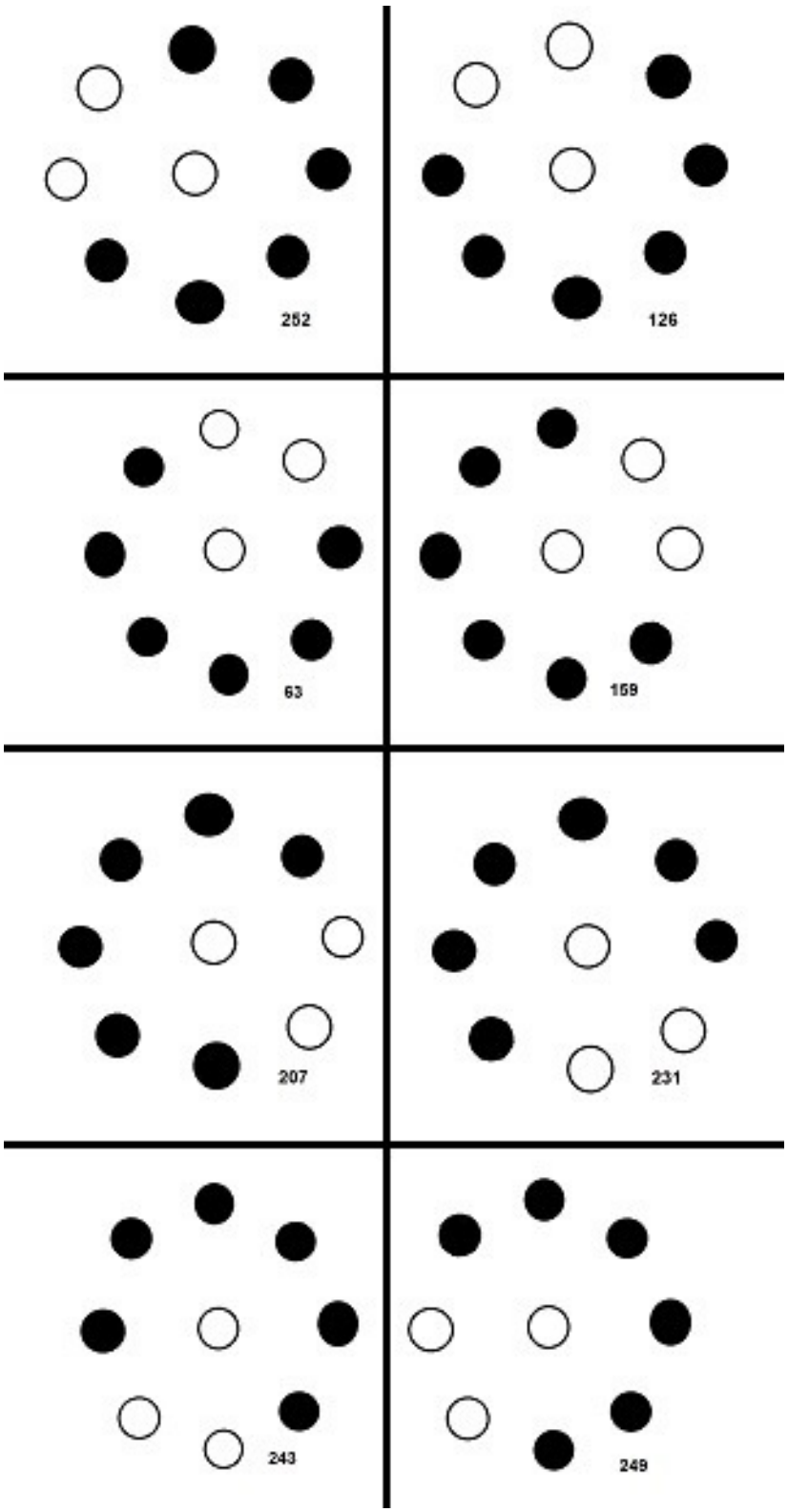
LBP for three pixels highlighted (with their corresponding decimal equivalent values).



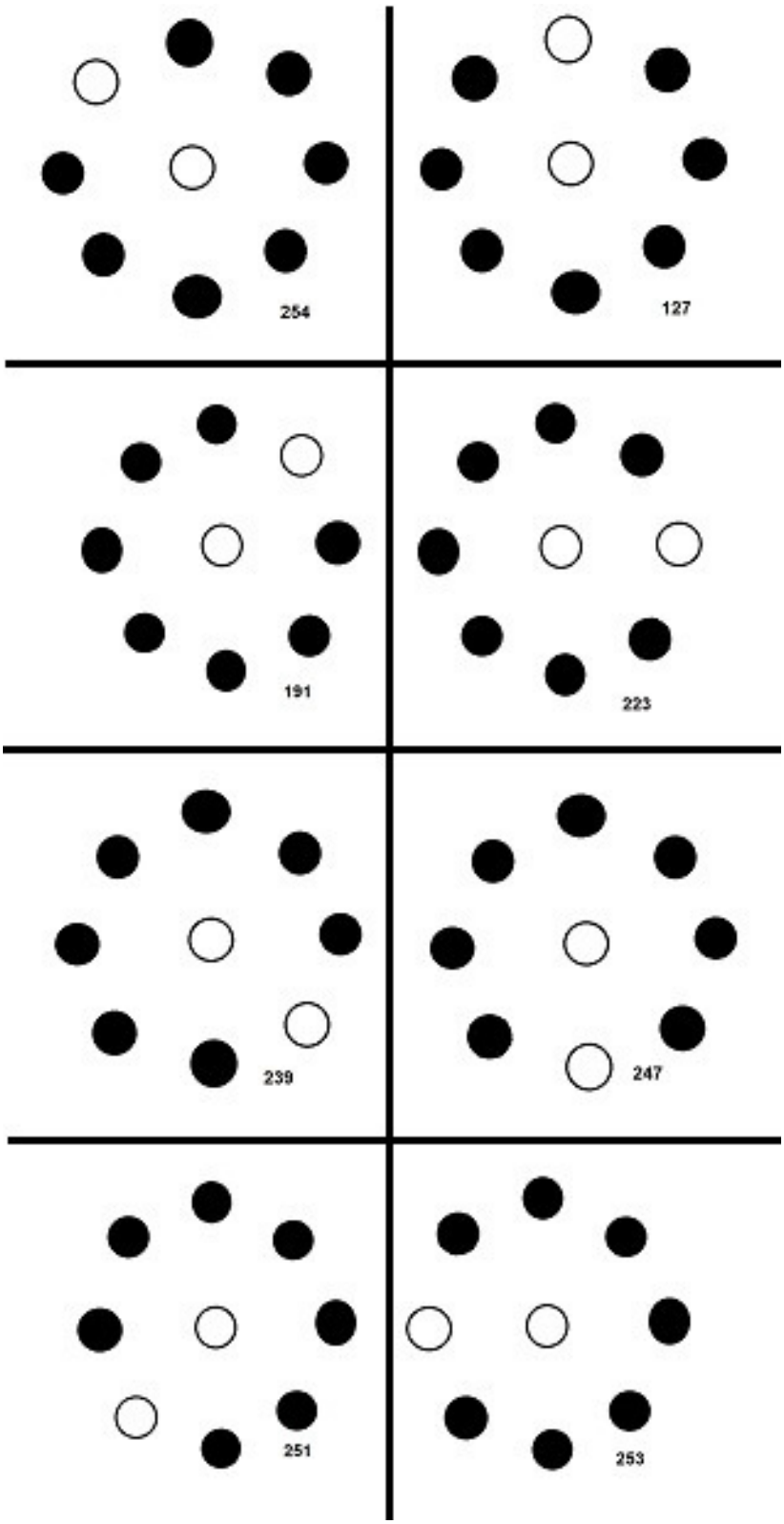
LBP for four pixels highlighted (with their corresponding decimal equivalent values).



LBP for five pixels highlighted (with their corresponding decimal equivalent values).



LBP for six pixels highlighted (with their corresponding decimal equivalent values).



LBP for two pixels highlighted (with their corresponding decimal equivalent values).

SDLC-Software Development Life Cycle

SDLC is a life cycle that consists of certain steps that are maintained to develop a project. By this we can easily rectify the errors that occur during execution of a project. While integrating the modules, as there should not be any problem we follow SDLC.

- ## Types of SDLC

Full SDLC

Full SDLC means giving the whole project to the only one developer i.e. all the phases are completed by one developer.

Partial SDLC

Partial SDLC means giving the project to the different developers in different modules. This may be given up to detailed design stage to one developer or it may be given from coding & testing phase to another developer.

- ## Development Model

Development model is the set of steps that encompasses various methods, tools and procedures to accomplish the phases in SDLC.A model is chosen based on the nature of the project, application & deliverables that are required.

Waterfall Model

This life cycle model demands a systematic sequential approach to software development that begins with the customer's software requirements & progresses through analysis, design, coding, testing & maintenance.

V-Model

In this model if there are any changes to be done in future there is no need to start from user requirements stage. We can decide the stage from which we have to start in-case of any modifications.

Spiral Model

The Spiral Model for software engineering has been developed to encompass the best features of both the Classic life cycle & Prototyping, while at the same time adding a new element-risk analysis-that is missing in these paradigms.

The Spiral Model defines 4 major activities.

- Planning: Determination of objectives, alternatives & constraints.
- Risk Analysis: Analysis of alternatives & identification resolution of risks.
- Engineering: Development of the "next level" product.
- Customer Evaluation: Assessment of the results of Engineering.

- ## Methodology in developing software product

The project titled “Description of Interest Regions with Local Binary Patterns (IN MATLAB)” is developed based on the Waterfall Model, which consists of 5 phases.

- Analysis
- Design
- Coding
- Testing
- Maintenance

1. Analysis

Software is always a part of a larger system. The work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software. This system view is essential when software must interface with other elements such as hardware, people and databases.

System engineering and analysis encompasses requirements gathering at the system level with small amount of top-level design and analysis.

2. Design

Design is concerned with identifying software components, specifying relationship among components, specifying software structure, maintaining a record of design decisions and providing blue print for the implementation phase. Design consists of Architectural Design and Detailed Design.

Architectural Design involves identifying software components, decoupling and decomposing them into processing modules and conceptual data structures, specifying the inter connections among components.

Detailed Design is concerned with the details of “how to”: how to package the processing modules and how to implement the processing algorithms, data structures and inter connections among modules and data structures.

3. Coding

The design must be translated into a machine-readable form. If design is performed in a detailed manner, coding can be accomplished mechanistically.

4. Testing

Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional external that is conducting tests to uncover errors and ensure that define input will produce actual results that agree with required results.

5. Maintenance

Software will undoubtedly undergo change after it is delivered to the customer. Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment or because the customer requires functional or performance enhancements.

- ## Advantages

The linear sequential model is the oldest and the most widely used paradigm for software engineering.

It provides a template for each method (i.e. analysis, design, coding, testing & maintenance).

- ## Limitations of Waterfall Model

Life cycle model is a valid model of the development process in situations where it is possible to write a reasonably complete set of specifications for the software product at the beginning of the life cycle.

System Design

- ## Introduction to UML

The unified modeling language is a standard language for specifying, Visualizing, Constructing and documenting the software system and its components. It is a graphical language that provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain and control information about the systems. The UML notation is useful for graphically depicting Object Oriented Analysis and Object Oriented Design (OOA and OOD) modules.

Visualizing

Through UML we see or visualize an existing system and ultimately we visualize how the system is going to be after implementation. Unless we think we cannot implement.UML helps to visualize how the components of the system communicate and interact with each other.

Specifying

Specifying means building models that are precise, unambiguous and complete UML addresses the specification of all the important Analysis Design, Implementation decisions that must be made in developing and deploying a software system.

Constructing

UML s models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like Java or C++ or VB. Forward Engineering and Reverse Engineering is possible through UML.

Documenting

The deliverables of a project apart from coding are some artifacts which are critical in controlling, measuring and communicating about a system during its development viz. Requirements, Architecture, Design, Source code, Project plans, Tests, Prototypes, Releases etc.

- ## UML Diagrams

Diagrams are graphical presentation of set of elements. Diagrams project a system, or visualize a system from different angles and perspectives. These diagrams help in quick and fair understanding of the system.

The UML has nine diagrams these diagrams can be classified into the following two groups.

- Static Diagrams
- Dynamic Diagrams

Static or Structural Diagrams

Class diagram

This shows a set of classes, interfaces, collaborations and their relationships. There are the most common diagrams in modeling the object oriented systems and are used to give the static view of a system.

Object diagram

Shows a set of objects and their relationships and are used to show the data structures, the static snapshots of instances of the elements in a class diagram. Like class diagram, the object diagrams also address the static design view or process view of a system.

Component diagram

Shows a set of components and their relationships and are used to illustrate the static implementation view of a system. They are related to class diagrams where in components map to one or more classes, interfaces of collaborations.

Deployment diagram

Shows a set of nodes and their relationships. They are used to show the static deployment view of the architecture of a system. They are related to the component diagrams where a node encloses one or more components.

Dynamic or behavioral diagrams

Use Case diagram

Use-case Diagrams show a set of use cases and actors and their relationships. These diagrams illustrate the static use case view of a system and are important in organizing and modeling the behaviors of a system.

Sequence diagram & collaboration diagram

These two diagrams are semantically same i.e. the dynamics of a system can be modeled using one diagram and transform it to the other kind of diagram without loss of information. Both form the, Interaction diagram.

Sequence diagram

Sequence diagram is an interaction diagram which focuses on the time ordering of messages it shows a set of objects and messages exchange between these objects. This diagram illustrates the dynamic view of a system.

Collaboration diagram

This diagram is an interaction diagram that stresses or emphasizes the structural organization of the objects that send and receive messages. It shows a set of objects, links between objects and messages send and received by those objects. There are used to illustrate the dynamic views of the system.

Activity Diagrams

Activity diagram shows the flow from one activity to another within a system. The activities may be sequential or branching objects that act and are acted upon. These also show the dynamic view of the system.

- ## UML diagrams for LBP

❖ Usecase Diagrams

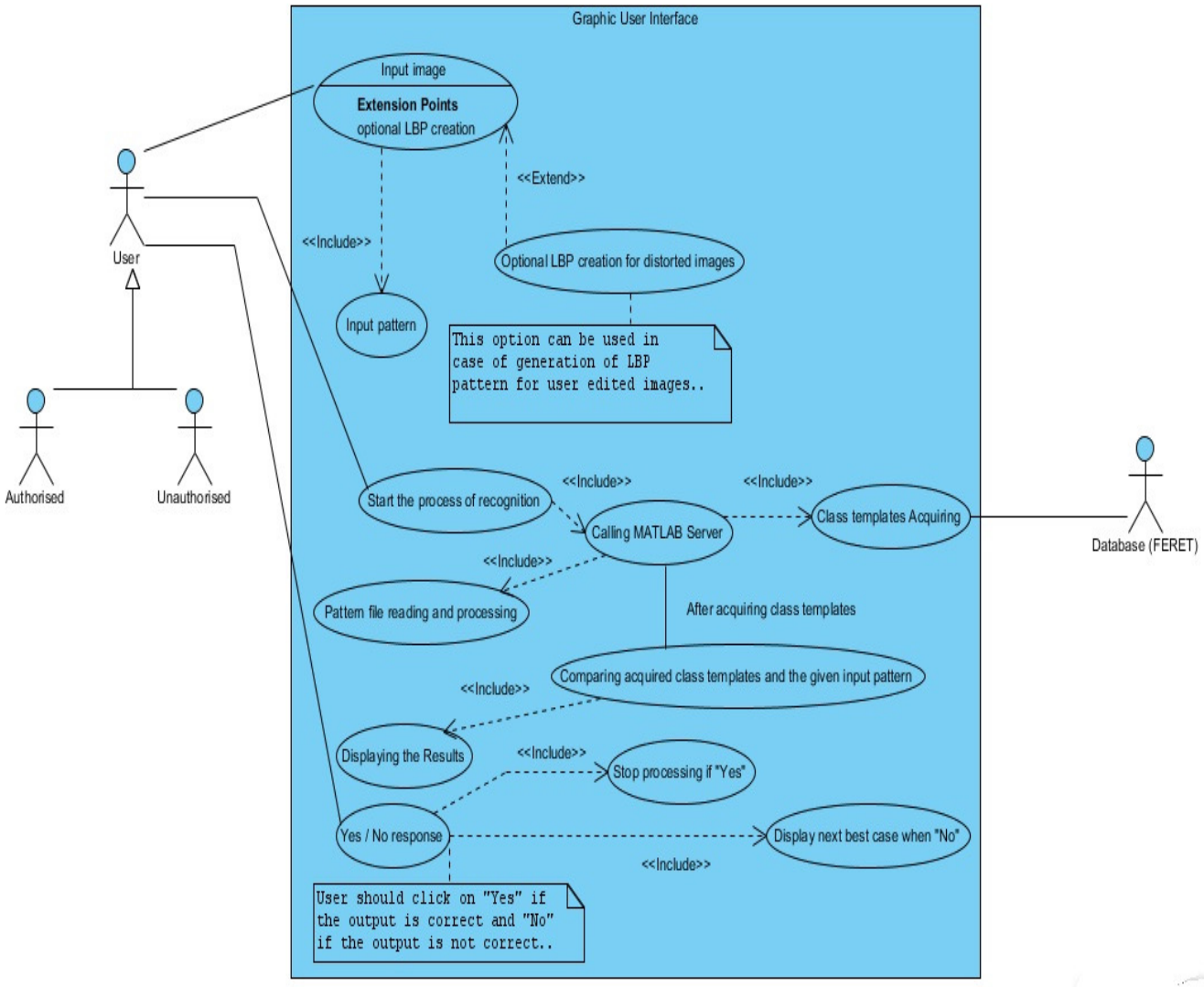
A Use Case Diagram is "a diagram that shows the relationships among actors and use cases within a system." Use case diagrams are often used to:

- ✓ Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model.
- ✓ Communicate the scope of a development project.
- ✓ Model your analysis of your usage requirements in the form of a system use case model.

A use case model is comprised of one or more use case diagrams and any supporting documentation such as use case specifications and actor definitions. Within most use case models the use case specifications tend to be the primary artifact with use case diagrams filling a supporting role as the "glue" that keeps your requirements model together. Use case models should be developed from the point of view of your project stakeholders and not from the (often technical) point of view of developers.

There are guidelines for:

- use cases
- Actors
- Relationships
- System Boundary Boxes



Usecase Diagram for LBP

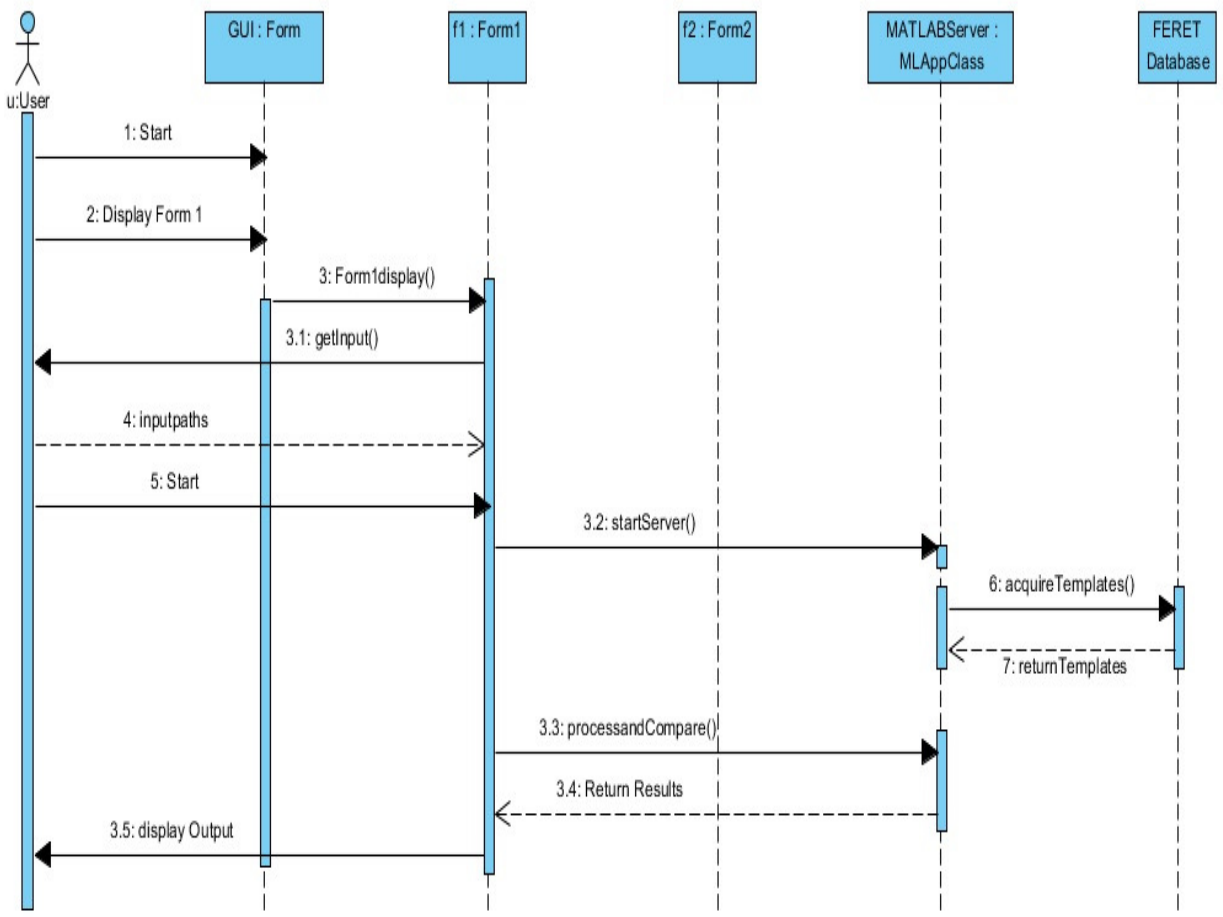
❖ Sequence Diagrams

Sequence Diagrams are a dynamic modeling technique, as are collaboration diagrams and activity diagrams. Models the sequential logic, it effect the time ordering of messages between classifiers. UML sequence diagrams are typically used to:

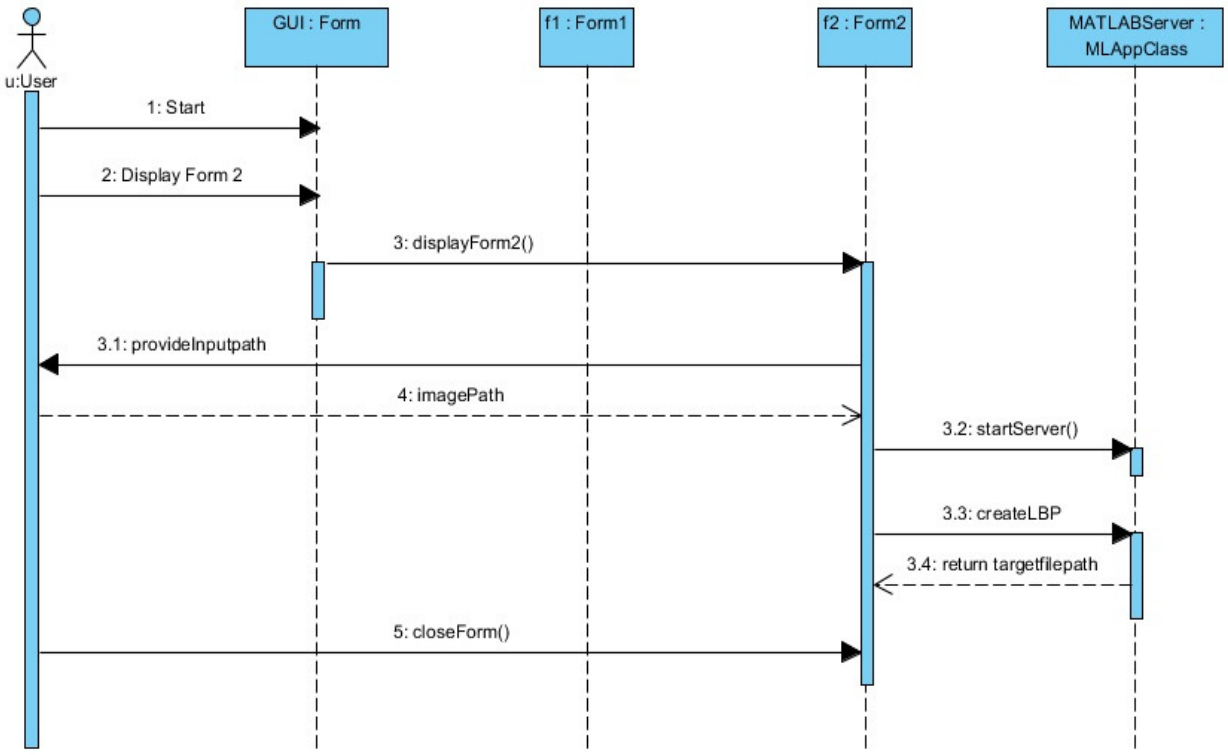
Validate and flesh out the logic of a usage scenario. A usage scenario is exactly what its name indicates – the description of a potential way that your system is used. The logic of a usage scenario may be part of a use case, perhaps an alternate

course; one entire pass through a use case, such as the logic described by the basic course of action or a portion of the basic course of action plus one or more alternate scenarios; or a pass through the logic contained in several use cases, for example a student enrolls in the university then immediately enrolls in three seminars.

- ✓ Explore your design because they provide a way for you to visually step through invocation of the operations defined by your classes.
- ✓ To detect bottlenecks within an object-oriented design. By looking at what messages are being sent to an object, and by looking at roughly how long it takes to run the invoked method, you quickly get an understanding of where you need to change your design to distribute the load within your system. In fact some CASE tools even enable you to simulate this aspect of your software.
- ✓ Give you a feel for which classes in your application are going to be complex, which in turn is an indication that you may need to draw state chart diagrams for those classes.



Sequence diagram if testing image provided.



User-distorted testing image provided.

❖ Statechart Diagrams

Statechart diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So Statechart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of Statechart diagram is to model life time of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purposes of using Statechart diagrams:

- ✓ To model dynamic aspect of a system.
- ✓ To model life time of a reactive system.
- ✓ To describe different states of an object during its life time.
- ✓ Define a state machine to model states of an object.

How to draw Statechart Diagram?

Statechart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.

Statechart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

Before drawing a Statechart diagram we must have clarified the following points:

- ✓ Identify important objects to be analyzed.
- ✓ Identify the states.
- ✓ Identify the events.

During the life cycle of an object (here order object) it goes through some states and there may be some abnormal exists also. This abnormal exit may occur due to some problem in the system. When the entire life cycle is complete it is considered as the complete transaction.

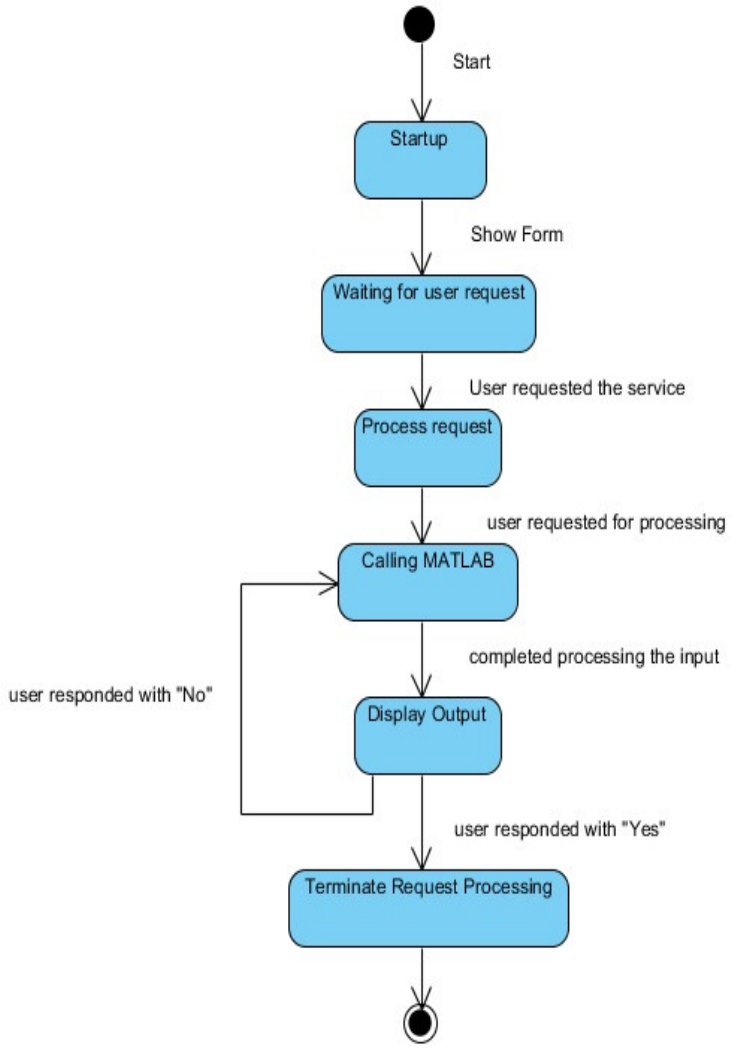
Where to use Statechart Diagrams?

Statechart diagrams are used to model dynamic aspect of a system like other four diagrams disused in this tutorial. But it has some distinguishing characteristics for modeling dynamic nature. Statechart diagram defines the states of a component and these state changes are dynamic in nature. So its specific purpose is to define state changes triggered by events. Events are internal or external factors influencing the system.

Statechart diagrams are used to model states and also events operating on the system. When implementing a system it is very important to clarify different states of an object during its life time and statechart diagrams are used for this purpose. When these states and events are identified they are used to model it and these models are used during implementation of the system.

So the main usages can be described as:

- ✓ To model object states of a system.
- ✓ To model reactive system. Reactive system consists of reactive objects.
- ✓ To identify events responsible for state changes.
- ✓ Forward and reverse engineering.



State chart diagram for LBP

❖ **Activity Diagram**

Activity diagram shows the flow from one activity to another within a system. The activities may be sequential or branching objects that act and are acted upon. These also show the dynamic view of the system.

The following are the steps to be followed for an activity diagram.

- Identify the scope of the activity diagram

Begin by identifying what it is you are modeling.

- ✓ Is it a single use case?
- ✓ Is it a portion of a use case?
- ✓ Is it a business process that includes several use cases?
- ✓ Is it a single method of a class?

Once you identify the scope of your diagram, you should add a label at the top, using a note, indicating an appropriate title for the diagram and a unique identifier for it. You may also want to include the date and even the names of the authors of the diagram.

- Add start and end points

Every activity diagram has a starting point and an ending point, so you might as well add them right away. In UML Distilled (see Resources), Fowler and Scott make ending points optional. Sometimes an activity is simply a dead end but, if this is the case, then there is no harm in indicating the only transition is to an ending point. This way, when someone else reads your diagram, he or she knows you have considered how to exit these activities.

- Add activities

If you are modeling a use case, introduce an activity for each major step initiated by an actor (this activity would include the initial step, plus any steps describing the response of the system to the initial step). If you are modeling a high-level business process, introduce an activity for each major process, often a use case or a package of use cases. Finally, if you are modeling a method, then it is common to have an activity for this step in the code.

- Add transitions from the activities

My style is always to exit an activity, even if it is simply to an ending point. Whenever there is more than one transition out of an activity, you must label each transition appropriately.

- Add decision points

Sometimes the logic of what you are modeling calls for a decision to be made. Perhaps something needs to be inspected or compared to something else. Important to note is that the use of decision points is optional.

- Identify opportunities for parallel activities

Two activities can occur in parallel when no direct relationship exists between them and they must both finish before a third activity can.

❖ **Class Diagram**

The purpose of the class diagram is to show the types being modeled within the system. In most UML models these types include:

- ✓ a class
- ✓ an interface
- ✓ a data type
- ✓ a component.

UML uses a special name for these types: "classifiers." Generally, you can think of a classifier as a class, but technically a classifier is a more general term that refers to the other three types above as well.

Class name

The UML representation of a class is a rectangle containing three compartments stacked vertically. The top compartment shows the class's name. The middle compartment lists the class's attributes. The bottom compartment lists the class's operations. When drawing a class element on a class diagram, you must use the top compartment, and the bottom two compartments are optional. (The bottom two would be unnecessary on a diagram depicting a higher level of detail in which the purpose is to show only the relationship between the classifiers.)

Class attribute list

The attribute section of a class (the middle compartment) lists each of the class's attributes on a separate line. The attribute section is optional, but when used it contains each attribute of the class displayed in a list format. The line uses the following format:

name : attribute type

In business class diagrams, the attribute types usually correspond to units that make sense to the likely readers of the diagram (i.e., minutes, dollars, etc.). However, a class diagram that will be used to generate code needs classes whose attribute types are limited to the types provided by the programming language, or types included in the model that will also be implemented in the system.

Sometimes it is useful to show on a class diagram that a particular attribute has a default value. (For example, in a banking account application a new bank account would start off with a zero balance.) The UML specification allows for the identification of default values in the attribute list section by using the following notation:

name : attribute type = default value

Class operations list

The class's operations are documented in the third (lowest) compartment of the class diagram's rectangle, which again is optional. Like the attributes, the operations of a class are displayed in a list format, with each operation on its own line. Operations are documented using the following notation:

name(parameter list) : type of value returned

When documenting an operation's parameters, you may use an optional indicator to show whether or not the parameter is input to, or output from, the operation. Typically, these indicators are unnecessary unless an older programming language such as Fortran will be used, in which case this information can be helpful.

Inheritance

A very important concept in object-oriented design, inheritance, refers to the ability of one class (child class) to inherit the identical functionality of another class (super class), and then add new functionality of its own. To model inheritance on a class diagram, a solid line is drawn from the child class (the class inheriting the behavior) with a closed, unfilled arrowhead (or triangle) pointing to the super class.

Abstract classes and operations

The class which provides a facility for base classes to implement the parent class methods as they need is called an abstract class.

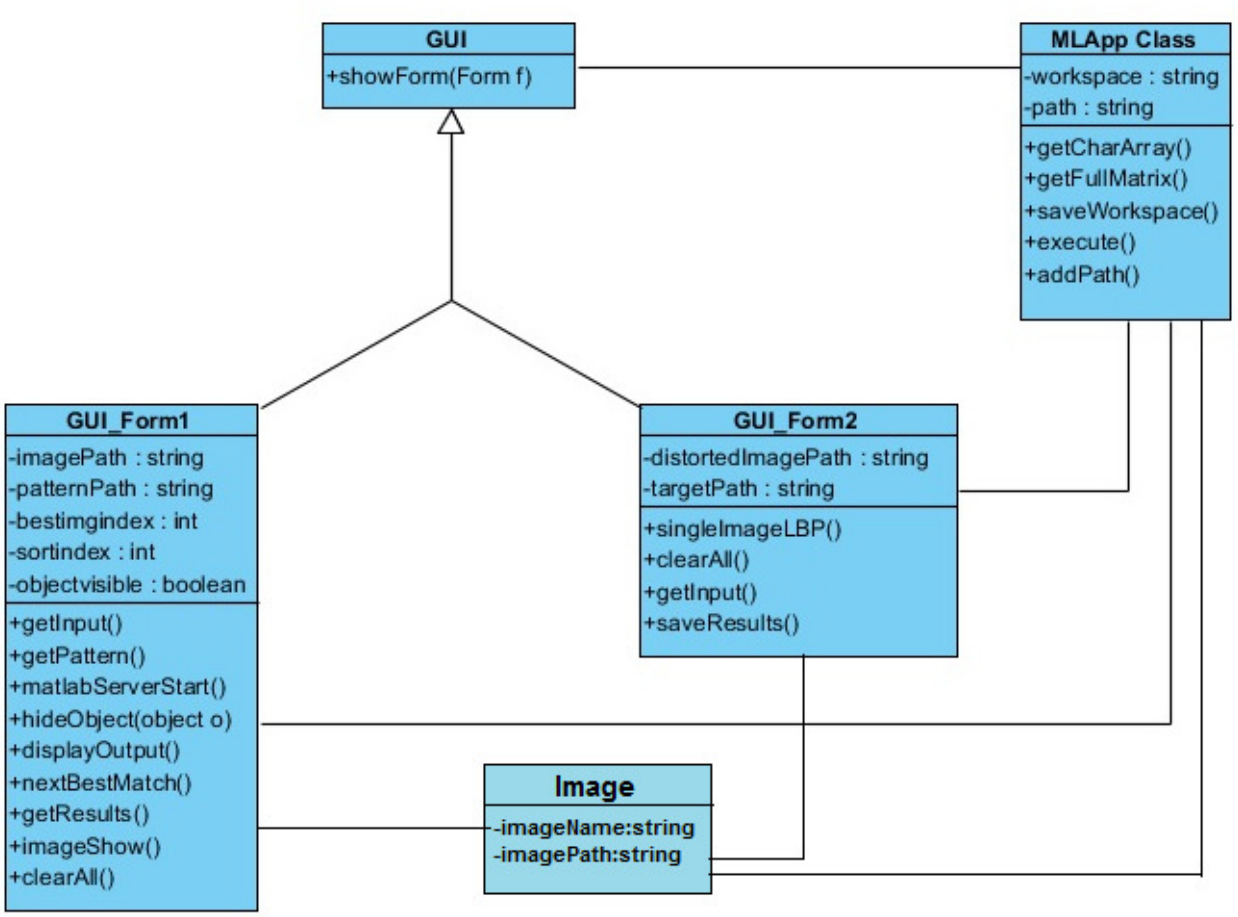
Associations

When you model a system, certain objects will be related to each other, and these relationships themselves need to be modeled for clarity. An association is a linkage between two classes. Associations are always assumed to be bi-directional; this means that both classes are aware of each other and their relationship, unless you qualify the association as some other type. A bi-directional association is indicated by a solid line between the two classes. At either end of the line, you place a role name and a multiplicity value for those wondering what the potential multiplicity values are for the ends of associations.

Potential Multiplicity Values

Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more

In a uni-directional association, two classes are related, but only one class knows that the relationship exists.



Class Diagram for LBP

Technologies Utilized

- **MATLAB 2010Ra**

MATLAB® is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. Using the MATLAB product, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN. You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas. MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

Key Features

- High-level language for technical computing
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics functions for visualizing data
- Tools for building custom graphical user interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel

MATLAB® supports the entire data analysis process, from acquiring data from external devices and databases, through preprocessing, visualization, and numerical analysis, to producing presentation-quality output.

The MATLAB® Language

The MATLAB® language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution. With the MATLAB language, you can program and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks, such as declaring variables, specifying data

types, and allocating memory. In many cases, MATLAB eliminates the need for 'for' loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.

At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, object-oriented programming (OOP), and debugging features.

MATLAB lets you execute commands or groups of commands one at a time, without compiling and linking, enabling you to quickly iterate to the optimal solution.

For fast execution of heavy matrix and vector computations, MATLAB uses processor-optimized libraries. For general-purpose scalar computations, MATLAB generates machine-code instructions using its JIT (Just-In-Time) compilation technology. This technology, which is available on most platforms, provides execution speeds that rival those of traditional programming languages.

Development Tools

MATLAB includes development tools that help you implement your algorithm efficiently. These include the following.

MATLAB Editor - Provides standard editing and debugging features, such as setting breakpoints and single stepping.

M-Lint Code Checker - Analyzes your code and recommends changes to improve its performance and maintainability.

MATLAB Profiler - Records the time spent executing each line of code

Directory Reports - Scan all the files in a directory and report on code efficiency, file differences, file dependencies, and code coverage.

Data Analysis

The MATLAB product provides interactive tools and command-line functions for data analysis operations, including:

- Interpolating and decimating
- Extracting sections of data, scaling, and averaging
- Thresholding and smoothing
- Correlation, Fourier analysis, and filtering
- 1-D peak, valley, and zero finding

- Basic statistics and curve fitting
- Matrix analysis

Data Access

MATLAB is an efficient platform for accessing data from files, other applications, databases, and external devices. You can read data from popular file formats, such as Microsoft Excel; ASCII text or binary files; image, sound, and video files; and scientific files, such as HDF and HDF5. Low-level binary file I/O functions let you work with data files in any format. Additional functions let you read data from Web pages and XML.

You can call other applications and languages, such as C, C++, COM objects, DLLs, Java, FORTRAN, and Microsoft Excel, and access FTP sites and Web services. Using the Database Toolbox™, you can also access data from ODBC/JDBC-compliant databases.

Importing and Exporting Graphic Files

MATLAB lets you read and write common graphical and data file formats, such as GIF, JPEG, BMP, EPS, TIFF, PNG, HDF, AVI, and PCX. As a result, you can export MATLAB plots to other applications, such as Microsoft Word and Microsoft PowerPoint, or to desktop publishing software. Before exporting, you can create and apply style templates, covering characteristics such as layout, font, and line thickness, to meet publication specifications.

Performing Numeric Computation

MATLAB® contains mathematical, statistical, and engineering functions to support all common engineering and science operations. These functions, developed by experts in mathematics, are the foundation of the MATLAB language. The core math functions use the LAPACK and BLAS linear algebra subroutine libraries and the FFTW Discrete Fourier Transform library. Because these processor-dependent libraries are optimized to the different platforms that MATLAB supports, they execute faster than the equivalent C or C++ code.

MATLAB provides the following types of functions for performing mathematical operations and analyzing data:

- Matrix manipulation and linear algebra
- Polynomials and interpolation
- Fourier analysis and filtering
- Data analysis and statistics

- Optimization and numerical integration
- Ordinary differential equations (ODEs)
- Partial differential equations (PDEs)
- Sparse matrix operations

MATLAB can perform arithmetic on a wide range of data types, including doubles, singles, and integers.

Add-on toolboxes (available separately) provide specialized mathematical computing functions for areas including signal processing, optimization, statistics, symbolic math, partial differential equation solving, and curve fitting.

Integrating MATLAB Code with Other Languages and Applications

MATLAB provides functions for integrating C and C++ code, FORTRAN code, COM objects, and Java code with your applications. You can call DLLs, Java classes, and ActiveX controls. Using the MATLAB engine library, you can also call MATLAB from C, C++, or FORTRAN code.

Deploying Applications

You can create your algorithm in MATLAB and distribute it to other MATLAB users directly as MATLAB code. Using the MATLAB Compiler (available separately), you can deploy your algorithm, as a stand-alone application or as a software module that you include in your project, to users who do not have MATLAB.

Additional products let you convert your algorithm into a software module that is callable from COM or Microsoft Excel.

.NET builder Toolbox (dotnetbuilder Toolbox)

Helps us integrate the applications developed in MATLAB to run through a GUI developed in .NET framework.

- ## Introduction to .NET 3.5 Framework

The Microsoft .NET Framework is a software technology that is available with several Microsoft Windows operating systems. It includes a large library of pre-coded solutions to common programming problems and a virtual machine that manages the execution of programs written specifically for the framework. The .NET Framework is a key Microsoft offering and is intended to be used by most new applications created for the Windows platform.

The pre-coded solutions that form the framework's Base Class Library cover a large range of programming needs in a number of areas, including user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library is used by programmers, who combine it with their own code to produce applications.

Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. Also part of the .NET Framework, this runtime environment is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security, memory management, and exception handling. The class library and the CLR together compose the .NET Framework.

Principal design features

- ✓ **Interoperability**

Because interaction between new and older applications is commonly required, the .NET Framework provides means to access functionality that is implemented in programs that execute outside the .NET environment. Access to COM components is provided in the System. Run time. Interlope Services and System. Enterprise Services namespaces of the framework; access to other functionality is provided using the P/Invoke feature.

- ✓ **Common Runtime Engine**

The Common Language Runtime (CLR) is the virtual machine component of the .NET framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviors in the areas of memory management, security, and exception handling.

- ✓ **Base Class Library**

The Base Class Library (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides classes which encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction and XML document manipulation.

- ✓ **Simplified Deployment**

Installation of computer software must be carefully managed to ensure that it does not interfere with previously installed software, and that it conforms to security requirements. The .NET framework includes design features and tools that help address these requirements.

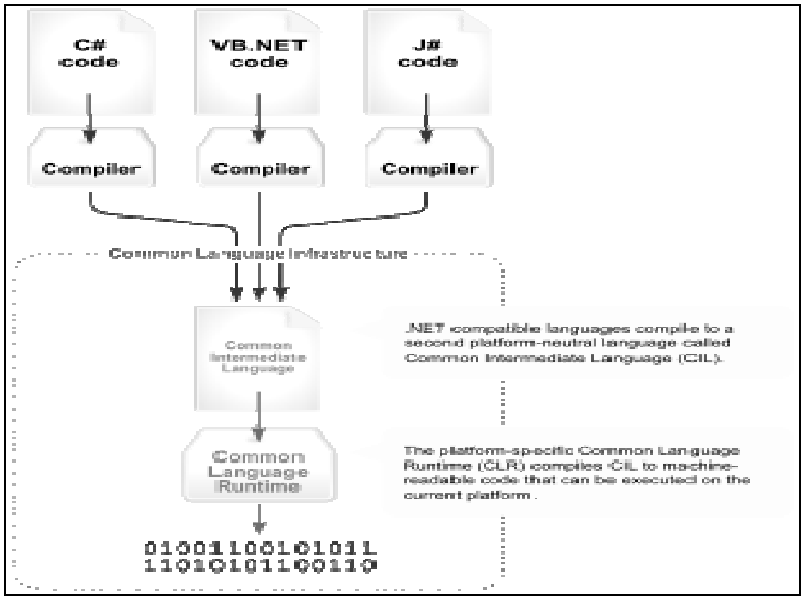
- ✓ **Security**

The design is meant to address some of the vulnerabilities, such as buffer overflows, that have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

- ✓ **Portability**

The design of the .NET Framework allows it to theoretically be platform agnostic, and thus cross-platform compatible. That is, a program written to use the framework should run without change on any type of system for which the framework is implemented. Microsoft's commercial implementations of the framework cover Windows, Windows CE, and the Xbox 360. In addition, Microsoft submits the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as open standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

✓ **Architecture**



Visual overview of the Common Language Infrastructure (CLI)

✓ **Common Language Infrastructure**

The core aspects of the .NET framework lie within the Common Language Infrastructure, or CLI. The purpose of the CLI is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability. Microsoft's implementation of the CLI is called the Common Language Runtime or CLR.

✓ **Assemblies**

The intermediate CIL code is housed in .NET assemblies. As mandated by specification, assemblies are stored in the Portable Executable (PE) format, common on the Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly (not to be confused with the filename on disk) contains its simple text name, version number, culture, and public key token. The public key token is a unique hash generated when the assembly is compiled, thus two assemblies with the same public key token are guaranteed to be identical from the point of view of the framework. A private key can also be specified known only to the creator of the assembly and can be used for strong naming and to guarantee that the assembly is from the same author when a new version of the assembly is compiled (required to add an assembly to the Global Assembly Cache).

- ✓ **Metadata**

All CLI is self-describing through .NET metadata. The CLR checks the metadata to ensure that the correct method is called. Metadata is usually generated by language compilers but developers can create their own metadata through custom attributes. Metadata contains information about the assembly, and is also used to implement the reflective programming capabilities of .NET Framework.

- ✓ **Security**

.NET has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. Code Access Security is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). Code Access Security uses evidence to determine the permissions granted to the code. Other code can demand that calling code is granted a specified permission. The demand causes the CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

When an assembly is loaded the CLR performs various tests. Two such tests are validation and verification. During validation the CLR checks that the assembly contains valid metadata and CIL, and whether the internal tables are correct. Verification is not so exact. The verification mechanism checks to see if the code does anything that is 'unsafe'. The algorithm used is quite conservative; hence occasionally code that is 'safe' does not pass. Unsafe code will only be executed if the assembly has the 'skip verification' permission, which generally means code that is installed on the local machine.

.NET Framework uses appdomains as a mechanism for isolating code running in a process. Appdomains can be created and code loaded into or unloaded from them independent of other appdomains. This helps increase the fault tolerance of the application, as faults or crashes in one appdomain do not affect rest of the application. Appdomains can also be configured independently with different security privileges. This can help increase the security of the application by isolating potentially unsafe code. The developer, however, has to split the application into sub domains; it is not done by the CLR.

✓ **Class library**

Namespaces in the BCL
System
System. Code Dom
System. Collections
System. Diagnostics
System. Globalization
System. IO
System. Resources
System. Text
System. Text. Regular Expressions

Microsoft **.NET Framework** includes a set of standard class libraries. The class library is organized in a hierarchy of namespaces. Most of the built in APIs are part of either System.* or Microsoft.* namespaces. It encapsulates a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. The .NET class libraries are available to all .NET languages. The .NET Framework class library is divided into two parts: the Base Class Library and the Framework Class Library.

The Base Class Library (BCL) includes a small subset of the entire class library and is the core set of classes that serve as the basic API of the Common Language Runtime. The classes in mscorlib.dll and some of the classes in System.dll and System.core.dll are considered to be a part of the BCL. The BCL classes are available in both .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silver light and Mono.

The Framework Class Library (FCL) is a superset of the BCL classes and refers to the entire class library that ships with .NET Framework. It includes an expanded set of libraries, including Win Forms, ADO.NET, ASP.NET, Language

Integrated Query, Windows Presentation Foundation, Windows Communication Foundation among others. The FCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to the standard libraries of Java.

- ✓ **Memory management**

The .NET Framework CLR frees the developer from the burden of managing memory (allocating and freeing up when done); instead it does the memory management itself. To this end, the memory allocated to instantiations of .NET types (objects) is done contiguously from the managed heap, a pool of memory managed by the CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use by the CLR. When there is no reference to an object, and it cannot be reached or used, it becomes garbage. However, it still holds on to the memory allocated to it. .NET Framework includes a garbage collector which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them.

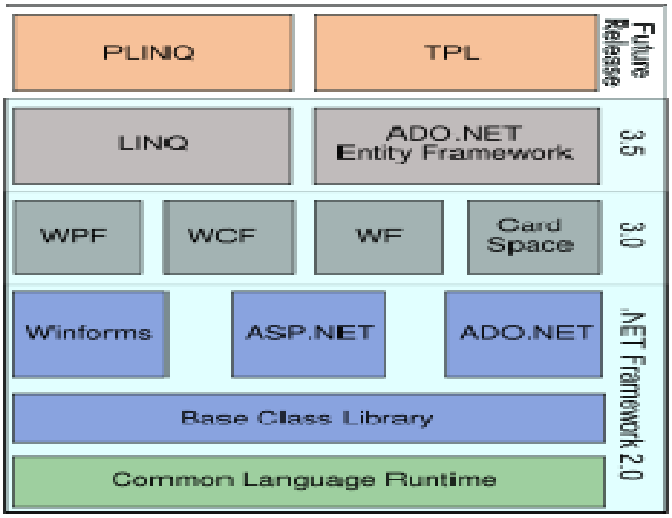
The .NET Garbage Collector (GC) is a non-deterministic, compacting, mark-and-sweep garbage collector. The GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, the GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (managed objects). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers. When the GC runs, it pauses the application, and for each object referred to in the root, it recursively enumerates all the objects reachable from the root objects and marks them as reachable. It uses .NET metadata and reflection to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage. This is the mark phase. Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then compacted together, by using memory to copy them over to the free space to make them contiguous again. Any reference to an object invalidated by moving the object is updated to reflect the new location by the GC. The application is resumed after the garbage collection is over.

The GC used by .NET Framework is actually generational. Objects are assigned a generation; newly created objects belong to Generation 0. The objects

that survive a garbage collection are tagged as Generation 1, and the Generation 1 objects that survive another collection are Generation 2 objects. The .NET Framework uses up to Generation 2 objects. Higher generation objects are garbage collected less frequently than lower generation objects. This helps increase the efficiency of garbage collection, as older objects tend to have a larger lifetime than newer objects. Thus, by removing older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need to be checked and compacted.

✓ **Versions**

Microsoft started development on the .NET Framework in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.

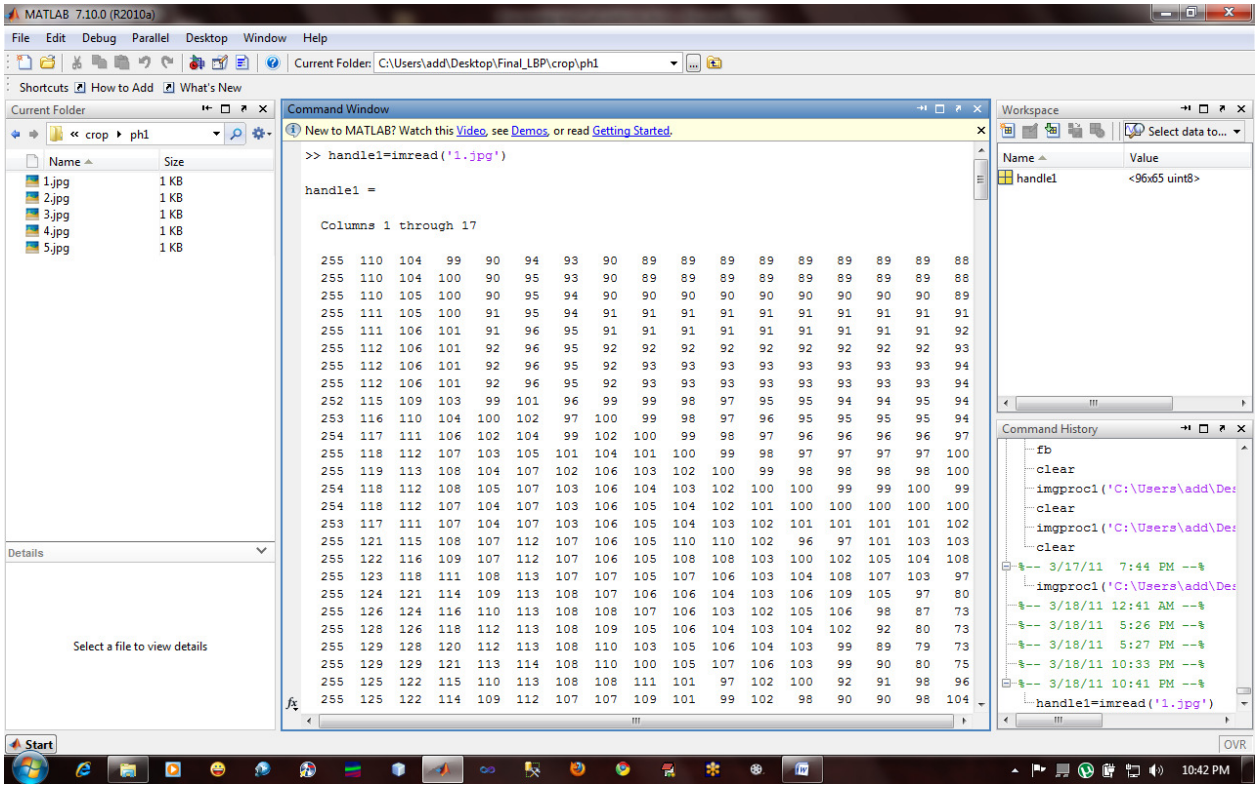


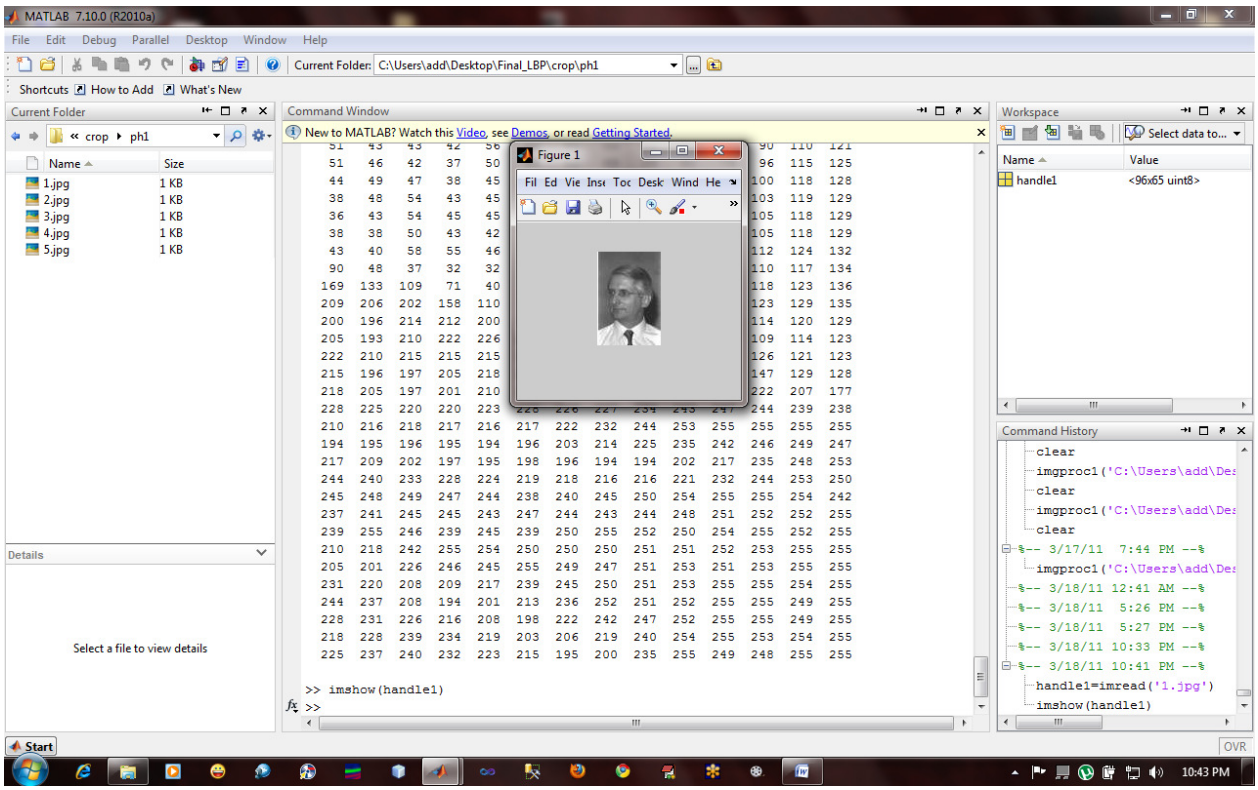
The .NET Framework Stack

✓ **The .NET Framework stack.**

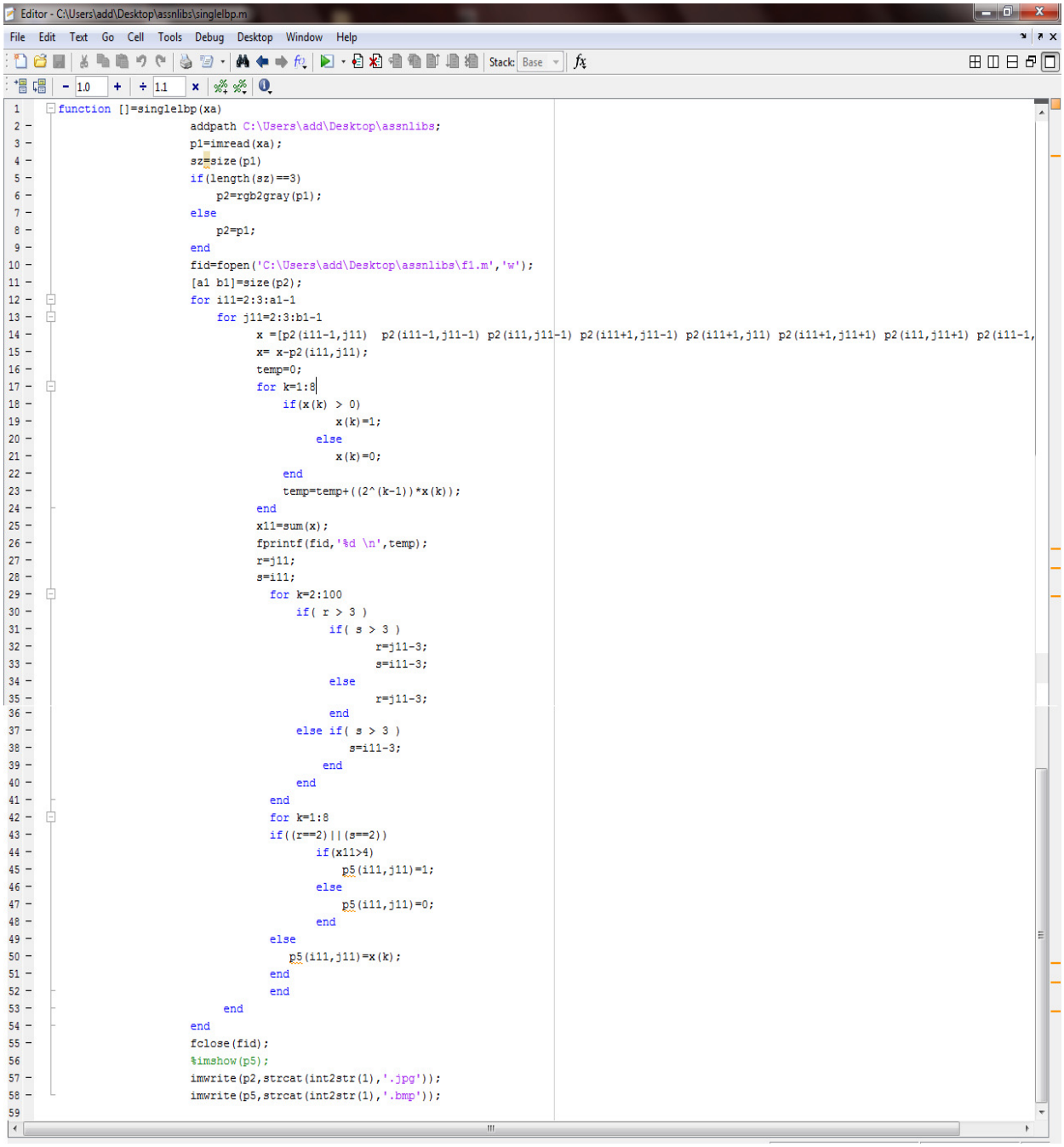
Version	Version Number	Release Date
1.0	1.0.3705.0	2002-01-05
1.1	1.1.4322.573	2003-04-01
2.0	2.0.50727.42	2005-11-07
3.0	3.0.4506.30	2006-11-06
3.5	3.5.21022.8	2007-11-09

- **Processing**


MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\add\Desktop\Final_LBP\crop\ph1
Shortcuts How to Add What's New
Current Folder
Name Size
1.jpg 1 KB
2.jpg 1 KB
3.jpg 1 KB
4.jpg 1 KB
5.jpg 1 KB
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> handle1=imread('1.jpg')
handle1 =
Columns 1 through 17
255 110 104 99 90 94 93 90 89 89 89 89 89 89 89 89 88
255 110 104 100 90 95 93 90 89 89 89 89 89 89 89 89 88
255 110 105 100 90 95 94 90 90 90 90 90 90 90 90 90 89
255 111 105 100 91 95 94 91 91 91 91 91 91 91 91 91 91
255 111 106 101 91 96 95 91 91 91 91 91 91 91 91 91 92
255 112 106 101 92 96 95 92 92 92 92 92 92 92 92 92 93
255 112 106 101 92 96 95 92 93 93 93 93 93 93 93 93 94
255 112 106 101 92 96 95 92 93 93 93 93 93 93 93 93 94
252 115 109 103 99 101 96 99 99 98 97 95 95 94 94 95 94
253 116 110 104 100 102 97 100 99 98 97 96 95 95 95 95 94
254 117 111 106 102 104 99 102 100 99 98 97 96 96 96 96 97
255 118 112 107 103 105 101 104 101 100 99 98 97 97 97 97 100
255 119 113 108 104 107 102 106 103 102 100 99 98 98 98 98 100
254 118 112 108 105 107 103 106 104 103 102 100 100 99 99 100 99
254 118 112 107 104 107 103 106 105 104 102 101 100 100 100 100 100
253 117 111 107 104 107 103 106 105 104 103 102 101 101 101 101 102
255 121 115 108 107 112 107 106 105 110 110 102 96 97 101 103 103
255 122 116 109 107 112 107 106 105 108 108 103 100 102 105 104 108
255 123 118 111 108 113 107 107 105 107 106 103 104 108 107 103 97
255 124 121 114 109 113 108 107 106 106 104 103 106 109 105 97 80
255 126 124 116 110 113 108 108 107 106 103 102 105 106 98 87 73
255 128 126 118 112 113 108 109 105 106 104 103 104 102 92 80 73
255 129 128 120 112 113 108 110 103 105 106 104 103 99 89 79 73
255 129 129 121 113 114 108 110 100 105 107 106 103 99 90 80 75
255 125 122 115 110 113 108 108 111 101 97 102 100 92 91 98 96
255 125 122 114 109 112 107 107 109 101 99 102 98 90 90 98 104
Workspace
Select data to...
Name Value
handle1 <96x65 uint8>
Command History
fb
clear
imgproc1('C:\Users\add\Des
clear
imgproc1('C:\Users\add\Des
clear
%-- 3/17/11 7:44 PM --%
imgproc1('C:\Users\add\Des
%-- 3/18/11 12:41 AM --%
%-- 3/18/11 5:26 PM --%
%-- 3/18/11 5:27 PM --%
%-- 3/18/11 10:33 PM --%
%-- 3/18/11 10:41 PM --%
handle1=imread('1.jpg')
Start
OVR
10:42 PM

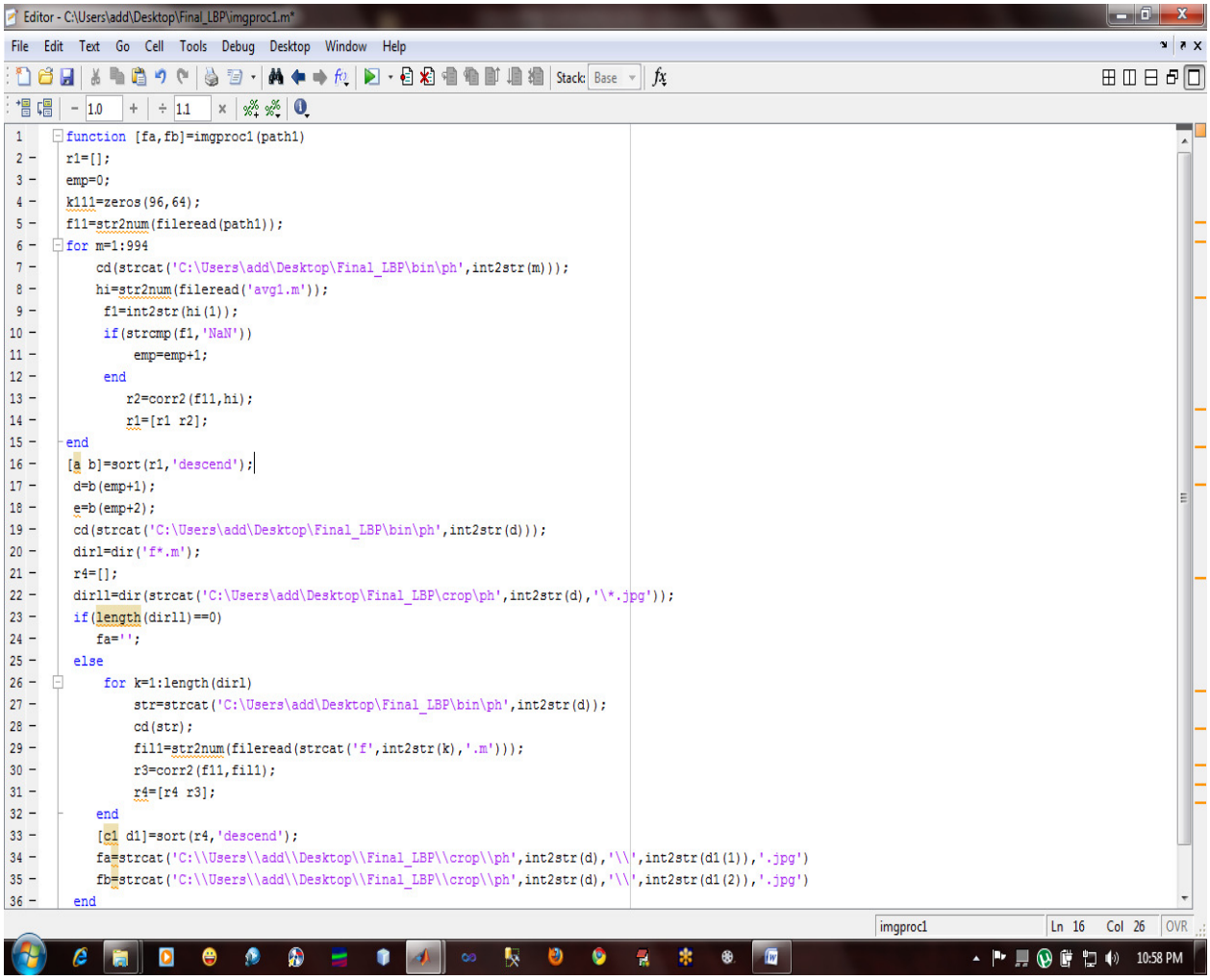

MATLAB 7.10.0 (R2010a)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\add\Desktop\Final_LBP\crop\ph1
Shortcuts How to Add What's New
Current Folder
Name Size
1.jpg 1 KB
2.jpg 1 KB
3.jpg 1 KB
4.jpg 1 KB
5.jpg 1 KB
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Figure 1
>> imshow(handle1)
>>
Workspace
Select data to...
Name Value
handle1 <96x65 uint8>
Command History
clear
imgproc1('C:\Users\add\Des
clear
imgproc1('C:\Users\add\Des
clear
%-- 3/17/11 7:44 PM --%
imgproc1('C:\Users\add\Des
%-- 3/18/11 12:41 AM --%
%-- 3/18/11 5:26 PM --%
%-- 3/18/11 5:27 PM --%
%-- 3/18/11 10:33 PM --%
%-- 3/18/11 10:41 PM --%
handle1=imread('1.jpg')
imshow(handle1)
Start
OVR
10:43 PM

- **Calculations**



The whole image is divided into blocks (3*3 grid) and the central pixel intensity value is taken as threshold and the surrounding pixel values are replaced by '1' if its intensity value is greater than the central pixel intensity value and '0' if its intensity value is less than the central pixel intensity value.Thus from each 3*3 grid a binary pattern is obtained which is converted to its decimal equivalent and stored in a file. These files which belong to a particular class are then used to create a class template which is then used for correlation with the given input image.

- **Comparison with Class templates**



The given input pattern is compared with the class templates using correlation and then the indexes are sorted based on the correlation value. The output is displayed through the GUI which calls the above function.

Experimentation with FERET database

Our approach is assessed on the face recognition problem using the images from the FERET database.

- **Experimental setup**

The system uses the FERET face images and follows the procedure of the FERET test for semiautomatic face recognition algorithms with slight modifications.

The FERET database consists of a total of 14051 gray-scale images representing 1199 individuals.

The images contain variations in lighting, facial expressions, pose angle etc. In this work, only frontal faces are considered. These facial images can be divided into five sets as the following.

F_a set, used as a gallery set, contains frontal images of 1196 people.

F_b set, (1195 images) in which the subjects were asked for an alternative facial expression than in the F_a photograph.

F_c set, (194 images). The photos were taken under different lighting conditions.

Dup I set (722 images). The photos were taken later in time.

Dup II set (234 images). This is a subset of the Dup I set containing those images that were taken at least a year after the corresponding gallery image.

Here we use the recognition rate measure as the ranking index (Rank 1 and 2).

- **Results for the FERET database**

The final recognition results for the proposed method are shown in Table II and the rank curves are plotted LBP yields clearly higher recognition rates than the control algorithms in all the FERET test sets and in the statistical test. The results on the F_c and dup II sets show that especially with weighting, the LBP based description is robust to challenges caused by lighting changes or aging of the subjects but further research is still needed to achieve even better performance.

It should be noted that the CSU implementations of the algorithms whose results are included here do not achieve the same figures as in the original FERET test due to some modifications in the experimental setup as mentioned. The results of the original FERET test can be found.

- **Robustness of the method to face localization error**

Real-world face recognition systems need to perform face detection prior to face recognition. Automatic face localization may not be completely accurate so it is desirable that face recognition works under small localization errors.

The proposed face recognition method calculates histograms over the local regions so a small change in the position of the face relative to the grid causes changes in the labels only on the borders of the local regions. Therefore it can be expected that the proposed method is not sensitive to small changes in the face localization and that using larger local regions increases the robustness to errors.

The effect of localization errors to recognition rate of the proposed method compared to PCA MachCosine was systematically tested as follows. The training images for PCA and gallery (F_a) images were normalized to size 128*128 using provided eye coordinates. The F_b set was used as probes. The probes were also normalized to size 128_128 but a random vector (_X;_Y) was added to the face location, where _X and _Yare uncorrelated and normally distributed with mean 0 and standard deviation. Ten experiments were conducted with each probe totaling 11950 queries for each tested value.

The recognition rates of the LBP based method using window sizes 21*21 and 32*32 and PCA MachCosine as a function of the standard deviation of the simulated localization offset are plotted. It can be seen that when no error or only a small error is present, LBP with small local regions works well but as the localization error increases, using larger local regions produces better recognition rate. Most interestingly, the recognition rate of the local region based methods drops significantly slower than that of PCA.

- ## Nomenclature

The naming convention for the FERET imagery in this distribution is of the form nnnnnxxfffq_yymmdd.ext where

1. nnnnn is a five digit integer that uniquely identifies the subject.

2. xx is a two lowercase character string that indicates the kind of imagery.

3. fff is a set of three binary (zero or one) single character flags. In order these denote:

   - Indicates whether the image is releasable for publication. The flag has fallen into disuse: All images are available via this CDROM distribution, but still none may be published without the explicit written permission of the government.
   - Image is histogram adjusted if this flag is 1
   - Indicates whether the image was captured using ASA 200 or 400 films, 0 implies 200.

4. q is a modifier that is not always present. When it is, the meanings are as follows:

   - Glasses worn. Note that this flag is a sufficient condition only; images of subjects wearing glasses do not necessarily carry this flag. Some retroactive re-truthing of such images to fix this problem is warranted. See also "c" below.
   - Duplicate with different hair length.
   - Glasses worn and different hair length
   - Electronically scaled (resized) and histogram adjusted.
   - Clothing has been electronically retouched.
   - Image brightness has been reduced by 40%
   - Image brightness has been reduced by 80%
   - Image size has been reduced by 10%, with white border replacement
   - Image size has been reduced by 20%, with white border replacement
   - Image size has been reduced by 30%, with white border replacement

5. The three fields are the date that the picture was taken in year, month, and day format.

Two letter code	Pose Angle (degrees)	Description	Number in Database	Number of Subjects
F_a	0 = frontal	Regular facial expression	1762	1010
F_b	0	Alternative facial expression	1518	1009
B_a	0	Frontal "b" series	200	200
B_b	0	Alternative expression to B_a	200	200
B_k	0	Different illumination to B_a	200	200
B_b	+60	Subject faces to his left which is the photographer's right	200	200
B_c	+40		200	200
B_d	+25		200	200
B_e	+15		200	200
B_f	-15	Subject faces to his right which is the photographer's left	200	200
B_g	-25		200	200
B_h	-40		200	200
B_i	-60		200	200
Q_l	-22.5	Quarter left and right	763	508
Q_r	+22.5		763	508
H_l	-67.5	Half left and right	1246	904
H_r	+67.5		1298	939
P_l	-90	Profile left and right	1318	974
P_r	+90		1342	980
R_a	+45	Random images. See note below. Positive angles indicate subject faces to photographer's right	322	264
R_b	+10		322	264
R_c	-10		613	429
R_d	-45		292	238
R_e	-80		292	238

Notes:

- ✓ F_a indicates a regular frontal image
- ✓ F_b indicates an alternative frontal image, taken seconds after the corresponding F_a
- ✓ B_a is a frontal images which is entirely analogous to the F_a series
- ✓ B_j is an alternative frontal image, corresponding to a B_a image, and analogous to the F_b image
- ✓ B_k is also a frontal image corresponding to B_a, but taken under different lighting
- ✓ B_b through Bi is a series of images taken with the express intention of investigating pose angle effects (see below). Specifically, B_f – B_i are symmetric analogues of B_b-B_e.
- ✓ R_a through R_e are "random" orientations. Their precise angle is unknown. It appears that the pose angles are random but consistent. The pose angles in the table were derived by manual measurement of inter-eye distances in the image, and in their corresponding frontal image.

Note that the modifications 'd' through 'j' are the result of applying various off-line operations to real images in the database; the "parent" image is that image without the "q" modifier present at all.

Testing

- ## Introduction to Software Product testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

Strategic approach to software testing

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progresses by moving outward along the spiral towards integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

White Box Testing

This type of testing ensures that

- ✓ All independent paths have been exercised at least once
- ✓ All logical decisions have been exercised on their true and false sides

- ✓ All loops are executed at their boundaries and within their operational bounds
- ✓ All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form. We have created independently to verify that Data flow is correct, all conditions are exercised to check their validity, and all loops are executed on their boundaries.

Basic Path Testing

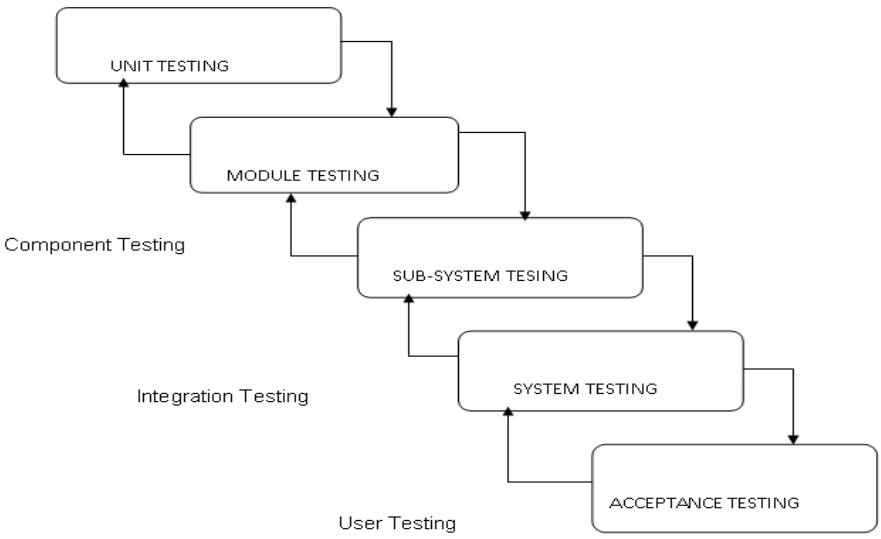
Established technique of flow graph with cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:
Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or
V(G)=P+1 or
V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity, E is the number of edges, N is the number of flow graph nodes, P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.



Conditional Testing

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

Data Flow Testing

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The definition-use chain method was used in this type of testing. These were particularly useful in nested statements.

Loop Testing

In this type of testing all the loops are tested to all the limits possible.

The following exercise was adopted for all loops:

- ✓ All the loops were tested at their limits, just above them and just below them.
- ✓ All the loops were skipped at least once.
- ✓ For nested loops test the inner most loop first and then work outwards.
- ✓ For concatenated loops the values of dependent loops were set with the help of connected loop.
- ✓ Unstructured loops were resolved into nested loops or concatenated loops and tested as above.
- ✓ Each unit has been separately tested by the development team itself and all the input have been validated.

System Testing

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps

- ## Testing with FERET cases

The system has been tested with FERET test samples given i.e... F_a, F_b, R_a, R_b and R_c. The Pattern matching is done using correlation.

The efficiency shown by the system is given below.

The system works even for distorted images. This has also been tested by user edited images and generating an LBP for the distorted image and then providing as input.

With all the image samples in the database the system gives an accuracy of 77.56 percent (6721 images matched correctly out of 8665 images).

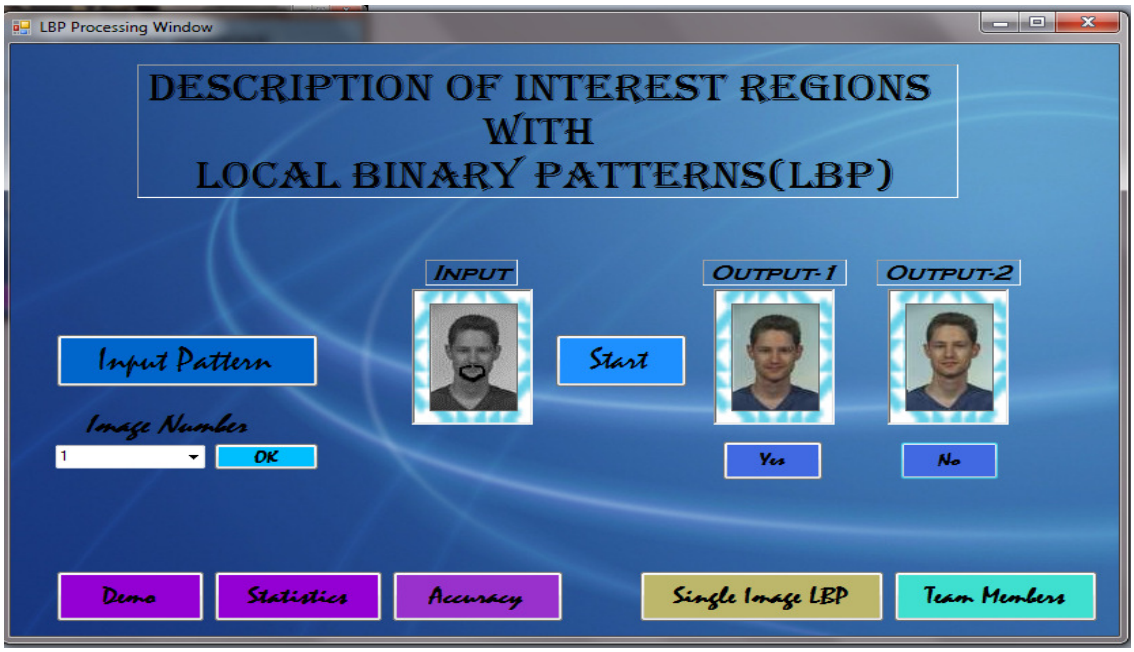
With classes whose sample image number ranges from zero to eleven the accuracy is 94.65 percent (4552 images out of 4809 images matched correctly).

With the classes whose sample image number ranges from zero to sixteen the accuracy given is 91.27 percent (5482 images out of 6006 images matched correctly).

The reason for choosing classes with eleven and sixteen image samples is the presence of those classes in good proportion in the database.

When F_a has been tested with F_b samples then the accuracy obtained

- ✓ With Class template is 32.73%.
- ✓ With Class template is 29.82%.



Output for the user distorted-input image.

5	7621	7625	7624	7623	7622	7625	7621						
8	7626	7633	7632	7628	7629	7633	7630	7631	7626	5627			
4	7634	7637	7635	7636	7637	7634							
7	7638	7644	7639	7640	7641	7643	7644	7638	6561				
4	7645	7648	7648	7646	7645	7647							
7	7649	7655	7650	7654	7651	7652	7655	7653	7656				
4	7656	7659	7658	7656	7659	7657							
7	7660	7666	7666	7665	7662	7663	7664	7661	7646				
10	7667	7676	7667	7668	7674	7671	7670	7669	7676	7673	7675	7672	
2	7677	7678	7678	7677									
0	7679	7678											
3	7679	7681	7679	7680	7681								
4	7682	7685	7684	7685	7683	7682							
10	7686	7695	7689	7688	7691	7690	7687	7693	7695	7692	7694	6422	
11	7696	7706	7699	7700	7705	7704	7701	7698	7702	7696	7706	508	7697
11	7707	7717	7708	7709	7716	7715	7711	7717	7710	6178	7707	4003	8142
11	7718	7728	7722	7726	7727	7724	7725	7723	7720	7721	7718	7719	213
11	7729	7739	7733	7731	7732	7739	7734	7737	7735	7729	7730	7736	8084
11	7740	7750	7748	7741	7742	7749	7743	7745	7740	7750	7744	1826	7716
11	7751	7761	7757	7753	7752	7755	7760	7759	7751	7754	7758	7756	7761
0	7762	7761											
11	7762	7772	7770	7764	7771	7763	7765	7766	7762	7767	7772	7768	6178
11	7773	7783	7782	7775	7783	7781	7776	7777	7774	7773	2281	4238	7778
6	7784	7789	7789	7787	7786	7788	7785	7784					
11	7790	7800	7798	7794	7799	7792	7800	7795	7796	7793	3978	7797	7791
4	7801	7804	7804	7801	7803	7802							
11	7805	7815	7807	7806	7813	7810	7814	7815	7809	7792	7805	7722	7808
11	7816	7826	7826	7822	7821	7825	7817	7818	7824	7820	7819	7823	4238
11	7827	7837	7835	7830	7836	7831	7837	7828	7827	7832	7829	1854	1511
11	7838	7848	7842	7846	7839	7838	7847	7840	7841	7848	7843	7844	7993
10	7849	7858	7851	7853	7856	7850	7857	7852	7849	7858	7854	8142	
11	7859	7869	7863	7861	7869	7862	7868	7864	7859	7860	7865	5625	3702
11	7870	7880	7879	7880	7872	7873	7878	7875	7871	7870	7874	7971	6191
11	7881	7891	7890	7886	7891	7885	7884	7883	7882	7889	2722	6178	7846
11	7892	7902	7900	7896	7895	7894	7901	7902	7898	7893	7897	7892	7899
11	7903	7913	7904	7912	7905	7911	7906	7907	7913	7908	7909	7910	2722
11	7914	7924	7922	7918	7919	7924	7923	7916	7917	7921	7915	7914	7832
11	7925	7935	7930	7931	7928	7934	7927	7935	7929	7926	7925	7933	1196
11	7936	7946	7944	7939	7940	7945	7941	7946	7938	7936	7942	8084	7937
10	7947	7956	7949	7956	7951	7954	7952	7955	7948	7792	4238	7947	
5	7957	7961	7958	7960	7961	7959	7957						
11	7962	7972	7971	7967	7964	7965	7970	7963	7966	7972	7722	7905	8046
11	7973	7983	7975	7978	7974	7976	7977	7982	7973	7981	7983	7879	5360
11	7984	7994	7986	7993	7992	7985	7987	7989	7988	7984	7990	7994	7842
11	7995	8005	8000	7999	8001	8003	7998	8004	7997	8005	7996	8002	8191
11	8006	8016	8010	8008	8016	8009	8015	8006	8007	8014	8011	8013	8012
11	8017	8027	8021	8025	8018	8019	8027	8017	8026	8022	6178	8020	4003
10	8028	8037	8032	8036	8028	8035	8029	8031	8030	8037	8033	8046	
10	8038	8047	8045	8042	8046	8040	8038	8047	8039	8041	8043	5362	
11	8048	8058	8056	8053	8057	8054	8058	8051	8055	8048	8050	8049	8052
6	8059	8064	8061	8060	8063	8059	8064	8062					
11	8065	8075	8066	8074	8068	8070	8065	8075	8067	8069	8071	8073	7722
11	8076	8086	8082	8084	8080	8081	8085	8086	8077	8079	8078	7832	8083
11	8087	8097	8091	8095	8088	8089	8090	8096	8092	6178	6191	7722	8097
11	8098	8108	8106	8103	8099	8100	8107	8104	8102	8105	8108	8101	8098
11	8109	8119	8118	8117	8113	8111	8112	8110	8119	8114	3878	8109	2108
11	8120	8130	8126	8123	8124	8128	8127	8129	8121	8125	8122	8130	8120
10	8131	8140	8133	8139	8132	8134	8135	8138	8131	8140	129	95	
10	8141	8150	8149	8145	8142	8147	8150	8144	8143	8146	8141	8148	
10	8151	8160	8159	8160	8152	8154	8155	8153	8151	8156	8158	3918	
11	8161	8171	8164	8166	8169	8165	8163	8171	8167	8161	8168	8162	8170
11	8172	8182	8180	8176	8178	8181	8172	8175	8173	8174	8177	1754	8182
11	8183	8193	8185	8191	8192	8187	8186	8183	8184	8193	8188	7792	3878
11	8194	8204	8195	8199	8196	8202	8198	8200	8197	8203	8204	7873	8192
11	8205	8215	8208	8206	8207	8215	8209	8214	8213	8205	8212	8211	2724
11	8216	8226	8218	8219	8217	8220	8224	8221	8222	8216	3816	2019	3993
11	8227	8237	8237	8236	8233	8234	8232	8231	8230	8228	8235	8229	8227
11	8238	8248	8242	8239	8240	8241	8246	8247	8243	8248	8238	8244	8245
7	8249	8255	8253	8255	8249	8252	8250	8254	8251				
8	8256	8263	8262	8258	8260	8259	8263	8261	8257	8256			
11	8264	8274	8274	8270	8272	8273	8267	8269	8266	8268	8265	8271	8264
11	8275	8285	8275	8284	8279	8283	8276	8278	8277	3665	8280	3816	8285
11	8286	8296	8289	8290	8291	8294	8287	8286	8295	8296	8288	7803	7691
11	8297	8307	8305	8298	8300	8302	8303	8297	8299	8301	8306	8307	8304
11	8308	8318	8310	8317	8309	8316	8312	8308	8318	8313	8311	1045	3755
11	8319	8329	8320	8321	8327	8322	8319	8326	8328	8329	8323	8324	8246
11	8330	8340	8339	8340	8335	8336	8332	8337	8331	8333	8330	8334	8338
11	8341	8351	8346	8345	8347	8348	8344	8349	8343	8342	8351	8341	8350
11	8352	8362	8358	8361	8362	8355	8359	8357	8360	8353	8356	8352	8337
11	8363	8373	8367	8373	8369	8368	8370	8371	8366	8363	8364	8372	8599
11	8374	8384	8384	8380	8378	8383	8381	8379	8374	8375	8382	8376	8377

Statistics sample for the database (Correlation).

Scope

This system can be used for various applications like User Authentication systems, User Verification systems. This application may be further extended for automatic login to the computers in the labs and issue of library books.

This application can be used software application that can be used anywhere which needs automatic log of all the persons attendance entering the organization. A wide range of applications can be derived from this project such as maintaining log of Restricted persons, Rule violating persons, recognizing the person without proper immigration and alerting the security etc.,

Growing numbers of applications are starting to use face-recognition as the initial step towards interpreting human actions, intention, and behavior, as a central part of next-generation smart environments. Many of the actions and behaviors humans display can only be interpreted if you also know the person's identity, and the identity of the people around them. Examples are a valued repeat customer entering a store, or behavior monitoring in an eldercare or childcare facility, and command-and-control interfaces in a military or industrial setting. In each of these applications identity information is crucial in order to provide machines with the background knowledge needed to interpret measurements and observations of human actions.

Limitations

The samples for a subject in the database should be less than or equal to eleven or sixteen beyond which the accuracy of the system falls down (as shown in the testing section and according to the collected statistics).

The subjects whose samples are greater than the limit proposed may not be classified correctly in the first 5-10 attempts in some cases.

Requirements

- ## Application

MATLAB 2010 Ra

This version provides many bug fixes as well as the toolbox required for integrating the applications written in MATLAB with .NET framework.

Microsoft Visual Studio 2010

VS2010 is used for building the GUI which interacts with the MATLAB server.

.NET Version

.NET 3.5/4.0

- ## Toolboxes

Image Processing Toolbox

This provides functions which deal with processing the images that are acquired using Image Acquisition Toolbox.

Image Acquisition Toolbox

This Toolbox provides functions for acquiring images from the database or from a physical location on the disk.

- ## Database

FERET database

Standard database used to assess the system's performance and efficiency.

- ## Platform

Operating System : Windows 7.0/X/2003/2008

- **Hardware Requirements (minimum)**

  - ✓ Pentium Processor : 233 MHZ
  - ✓ RAM Capacity : 1GB
  - ✓ Hard Disk : 30GB

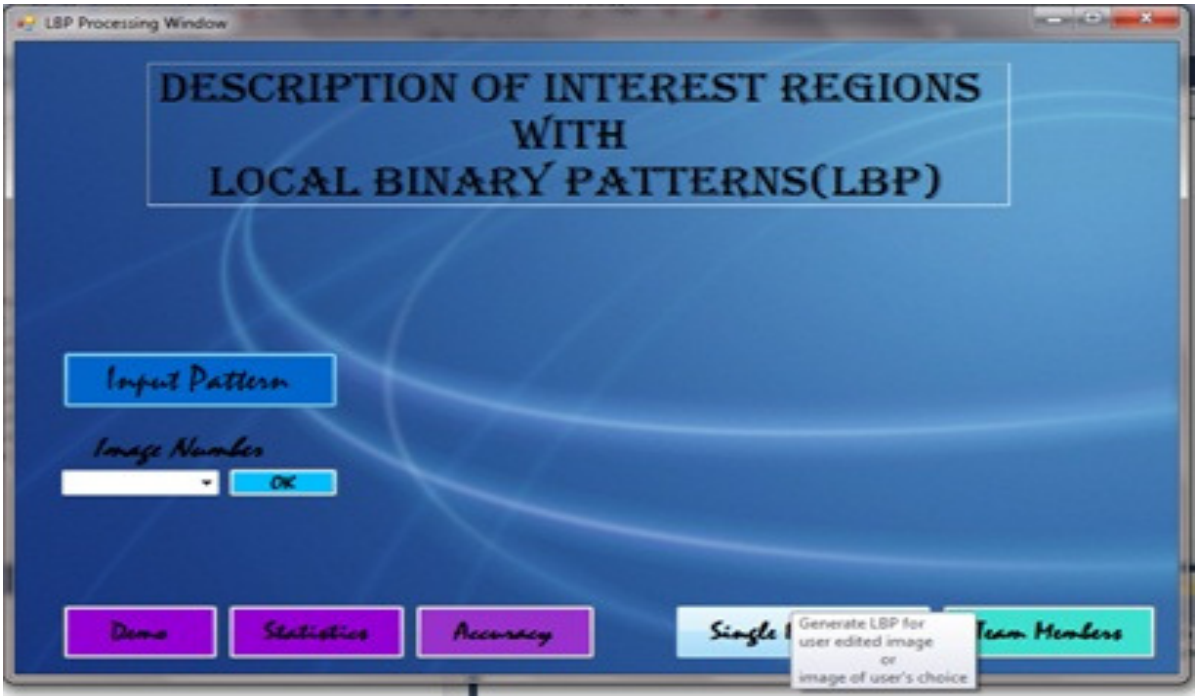
Conclusions

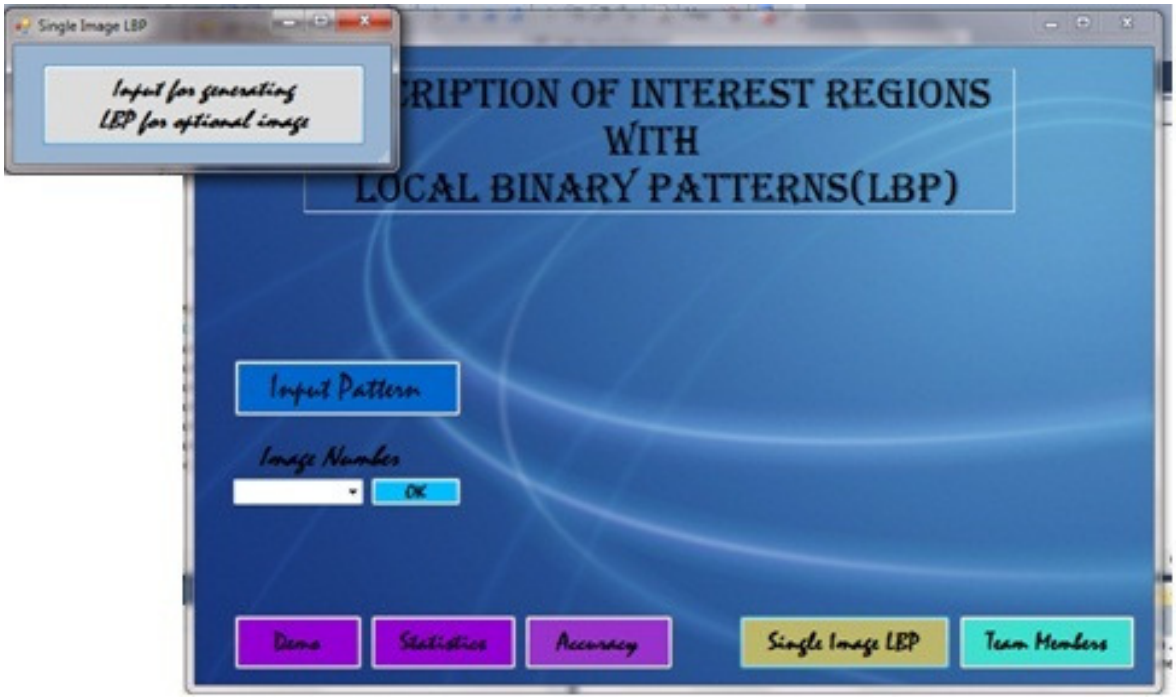
Face recognition can be done using Local Binary Patterns in an efficient manner which reduces the number of comparisons for instance the image (64*96) which is read as a matrix in MATLAB has a total of 64*96=6144 values, for a normal comparison the process should compare all the values for a match, using LBP these values reduce to 6144/9=672 (approx) for each image thus reducing the complexity in calculating the proximity of images.

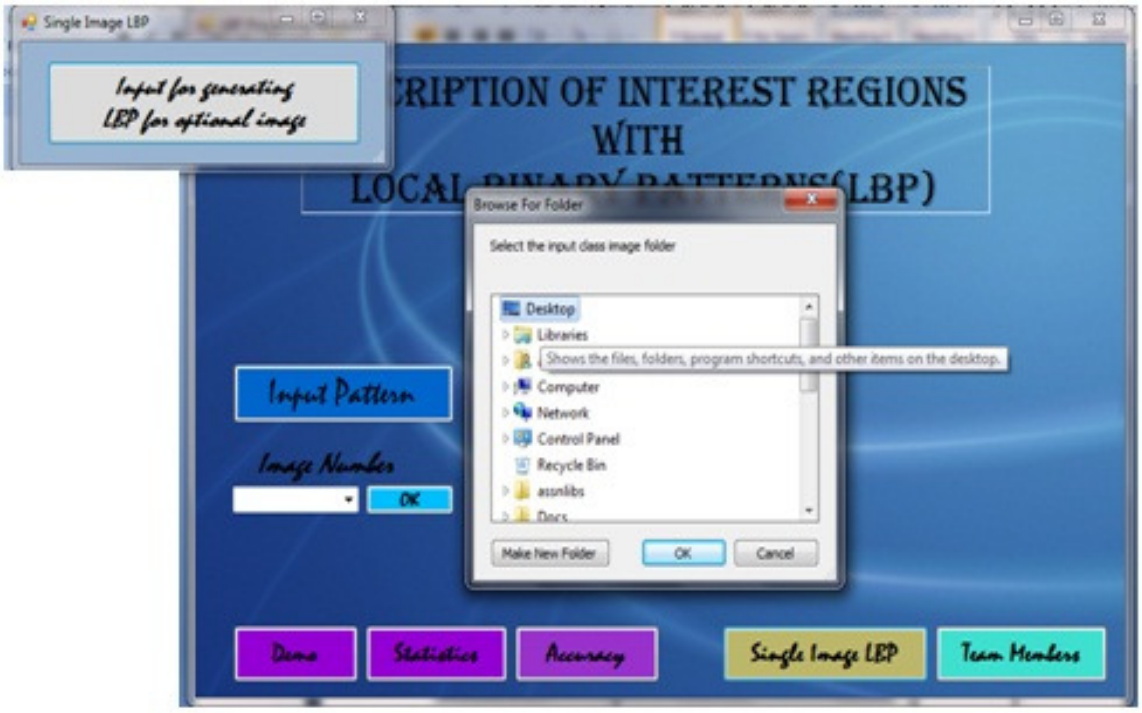
Face recognition by this approach can be even extended to be done without the notice of the subject by simply installing cameras in the surveillance.

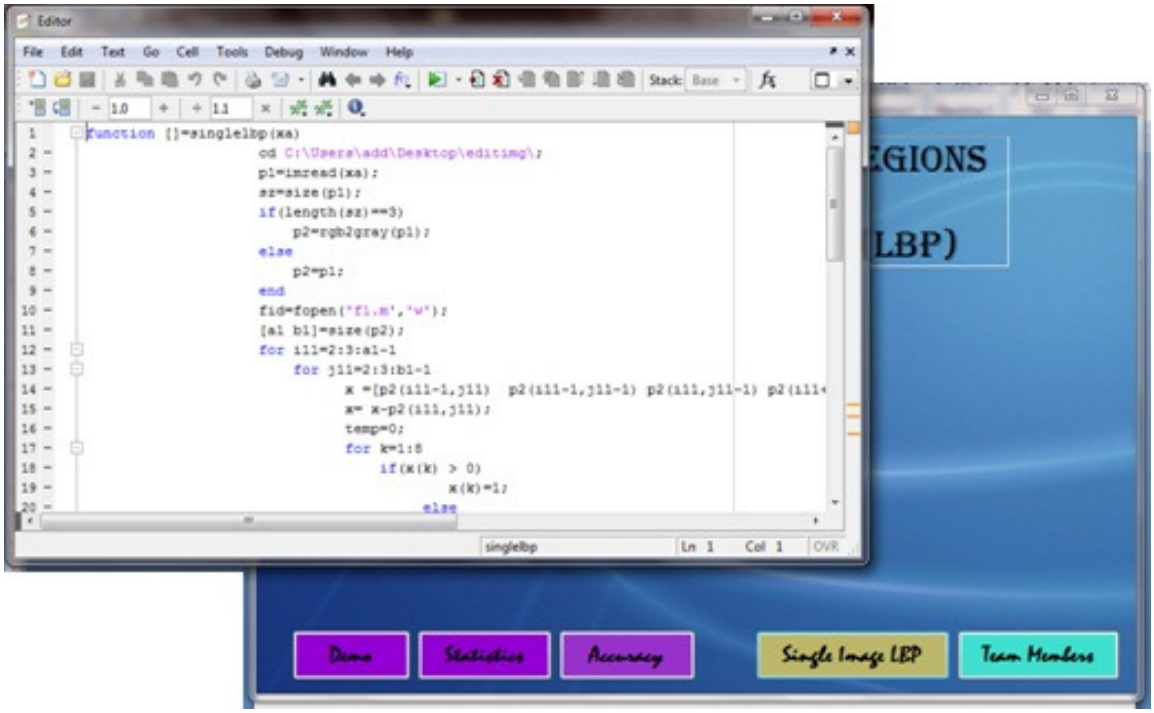
Appendix

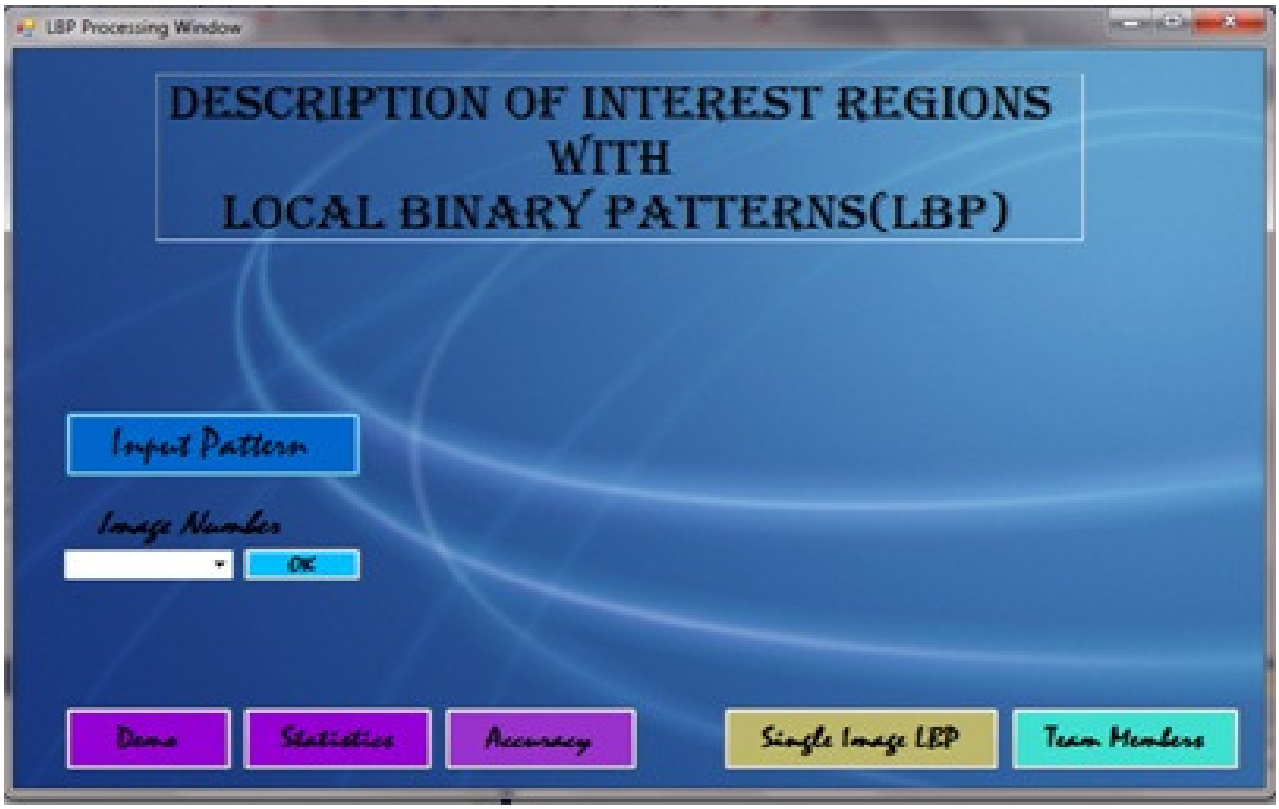
- **Snapshots**

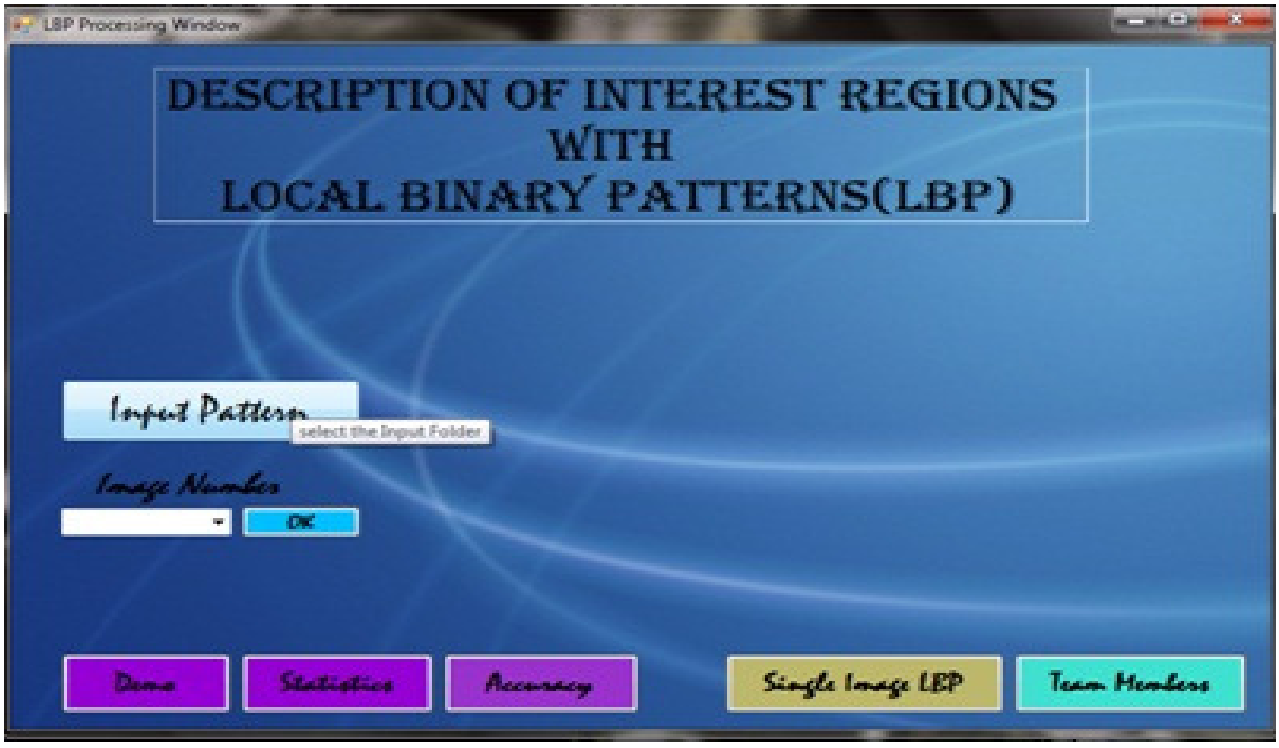




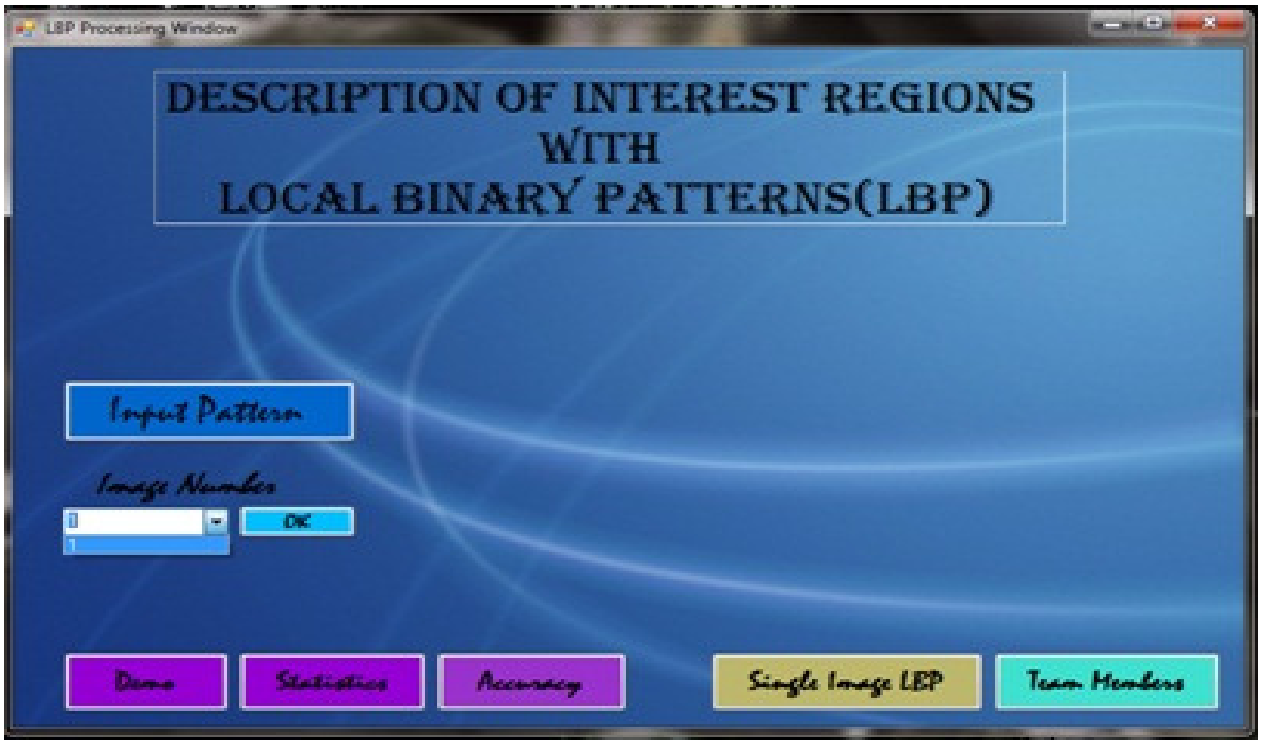

Single Image LBP
Input for generating LBP for optional image
RIPTION OF INTEREST REGIONS
WITH
LOCAL
(LBP)
Browse For Folder
Select the input class image folder
Desktop
Libraries
Shows the files, folders, program shortcuts, and other items on the desktop.
Computer
Network
Control Panel
Recycle Bin
asunlibs
Docs
Make New Folder
OK
Cancel
Input Pattern
Image Number
OK
Demo
Statistics
Accuracy
Single Image LBP
Team Members

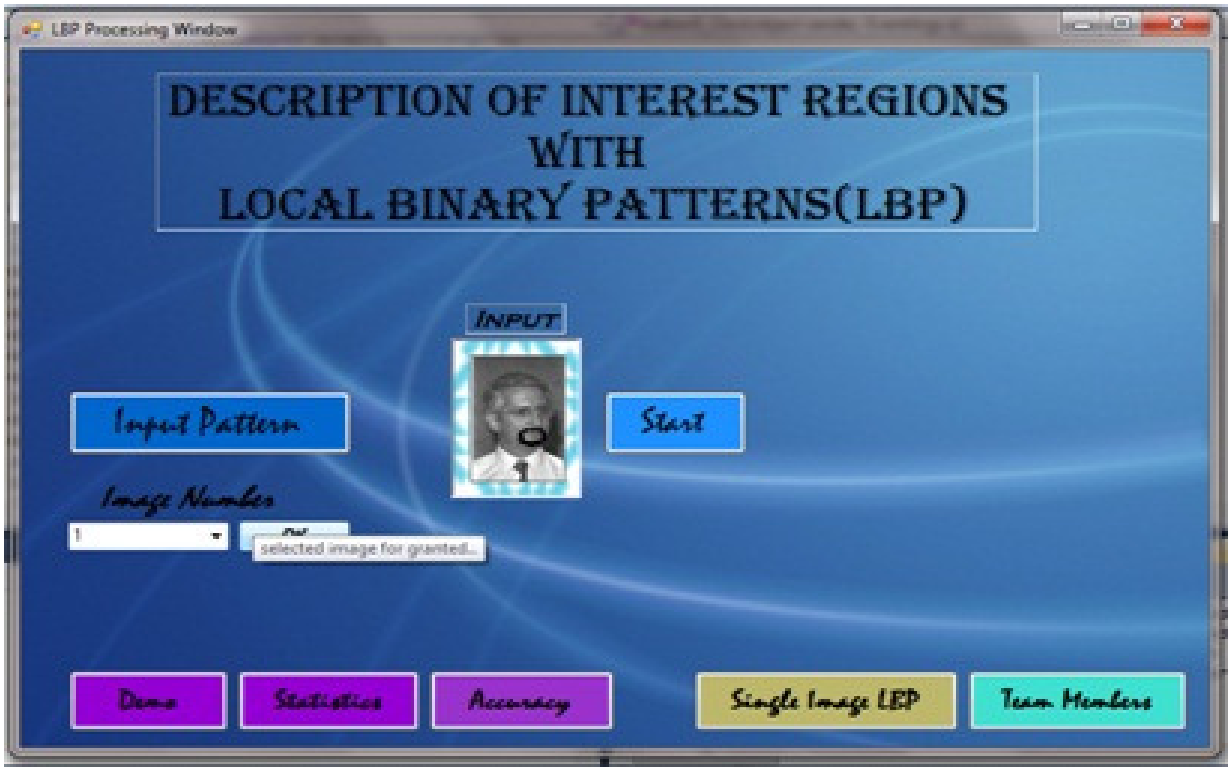

Editor
File Edit Text Go Cell Tools Debug Window Help
function []=singlelbp(xa)
cd C:\Users\add\Desktop\editimg\;
p1=imread(xa);
sz=size(p1);
if(length(sz)==3)
p2=rgb2gray(p1);
else
p2=p1;
end
fid=fopen('f1.m','w');
[a1 b1]=size(p2);
for i11=2:3:a1-1
for j11=2:3:b1-1
x =[p2(i11-1,j11) p2(i11-1,j11-1) p2(i11,j11-1) p2(i11
x= x-p2(i11,j11);
temp=0;
for k=1:8
if(x(k) > 0)
x(k)=1;
else
singlelbp
Ln 1 Col 1
OVR
GIONS
LBP)
Demo
Statistics
Accuracy
Single Image LBP
Team Members

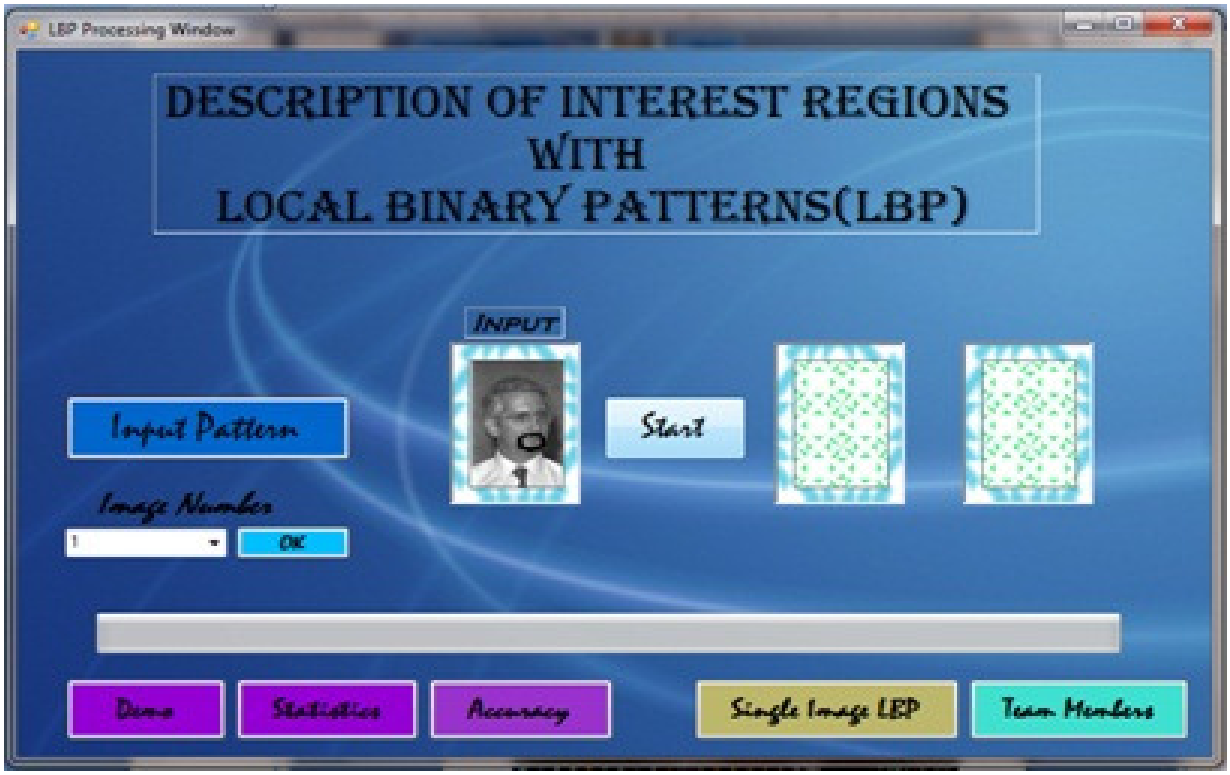

LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
Input Pattern
Image Number
OK
Demo
Statistics
Accuracy
Single Image LBP
Team Members

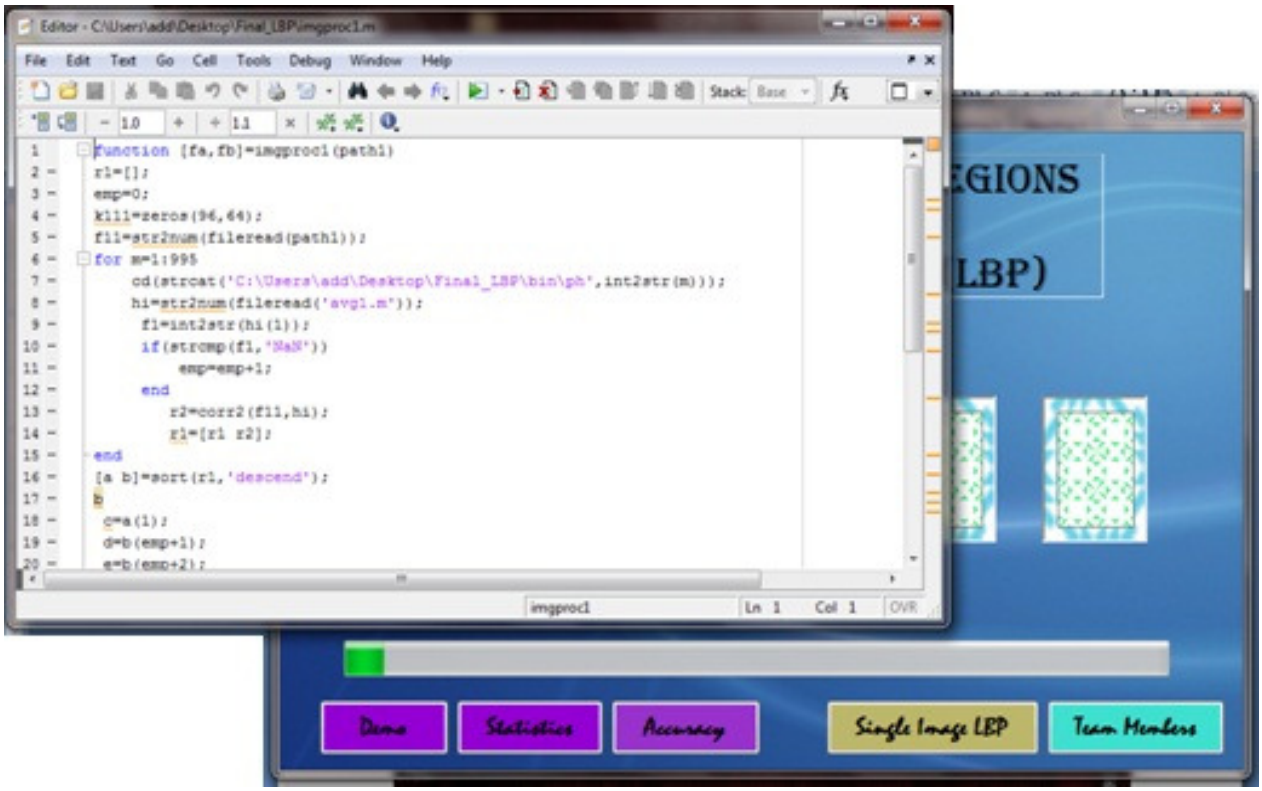

LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
Input Pattern
select the Input Folder
Image Number
OK
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
Browse For Folder
Select the input class pattern folder
asmlibs
Docs
editimg
expGUI
Final_LB
LBP
MY .net
my desk
OK
Cancel
Input Pattern
Image Number
OK
Demo
Statistics
Accuracy
Single Image LBP
Team Members

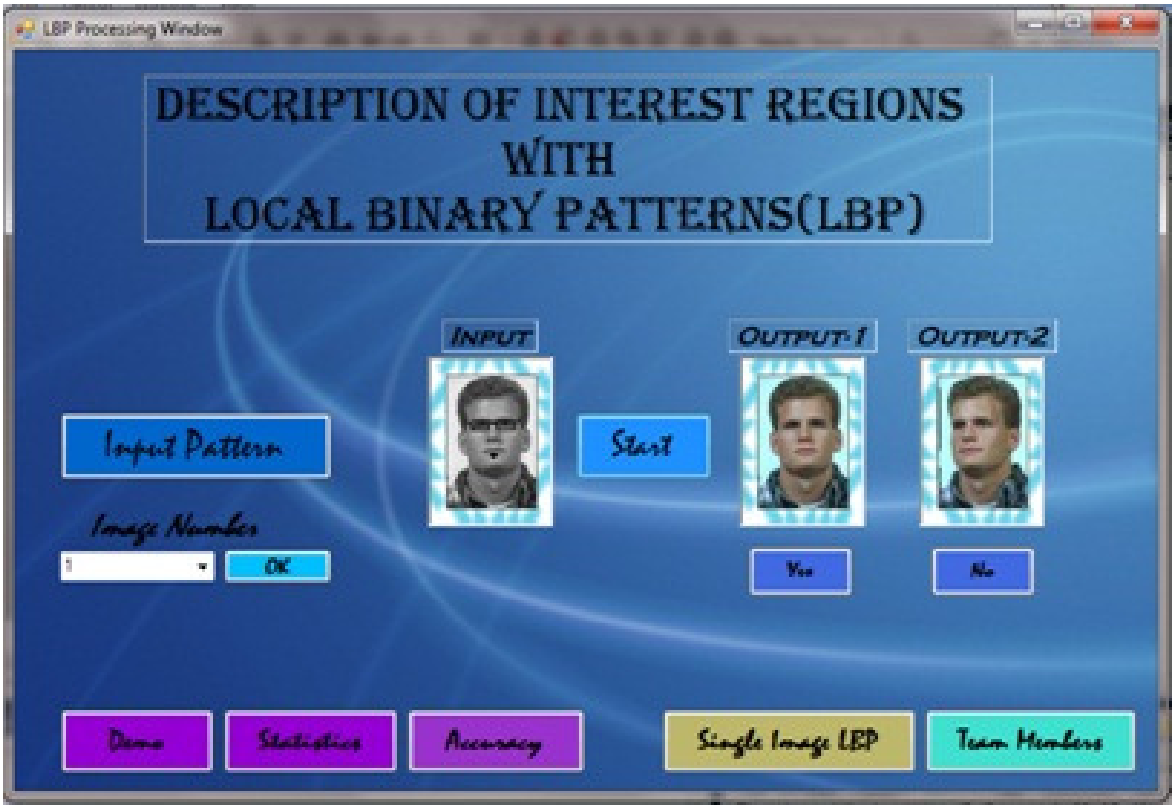

LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
Input Pattern
Image Number
OK
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
INPUT
Input Pattern
Start
Image Number
selected image for granted..
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
INPUT
Input Pattern
Start
Image Number
OK
Demo
Statistics
Accuracy
Single Image LBP
Team Members

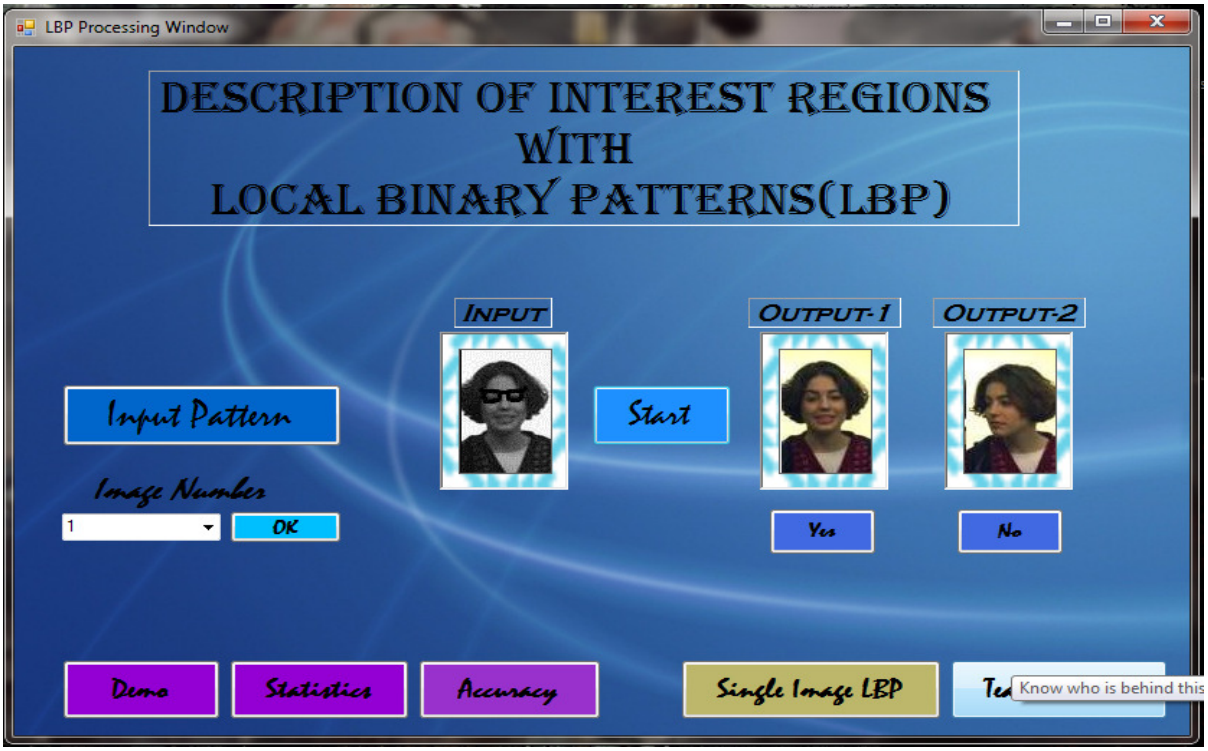

Editor - C:\Users\add\Desktop\Final_LBP\imgproc1.m
File Edit Text Go Cell Tools Debug Window Help
function [fa,fb]=imgproc1(path1)
rl=[];
emp=0;
k111=zeros(96,64);
fi1=str2num(fileread(path1));
for m=1:995
cd(strcat('C:\Users\add\Desktop\Final_LBP\bin\gh',int2str(m)));
hi=str2num(fileread('avg1.m'));
f1=int2str(hi(1));
if(strcmp(f1,'NaN'))
emp=emp+1;
end
r2=corr2(fi1,hi);
r1=[r1 r2];
end
[a b]=sort(r1,'descend');
b
c=a(1);
d=b(emp+1);
e=b(emp+2);
imgproc1
Ln 1 Col 1
OVR
GIONS
LBP)
Demo
Statistics
Accuracy
Single Image LBP
Team Members

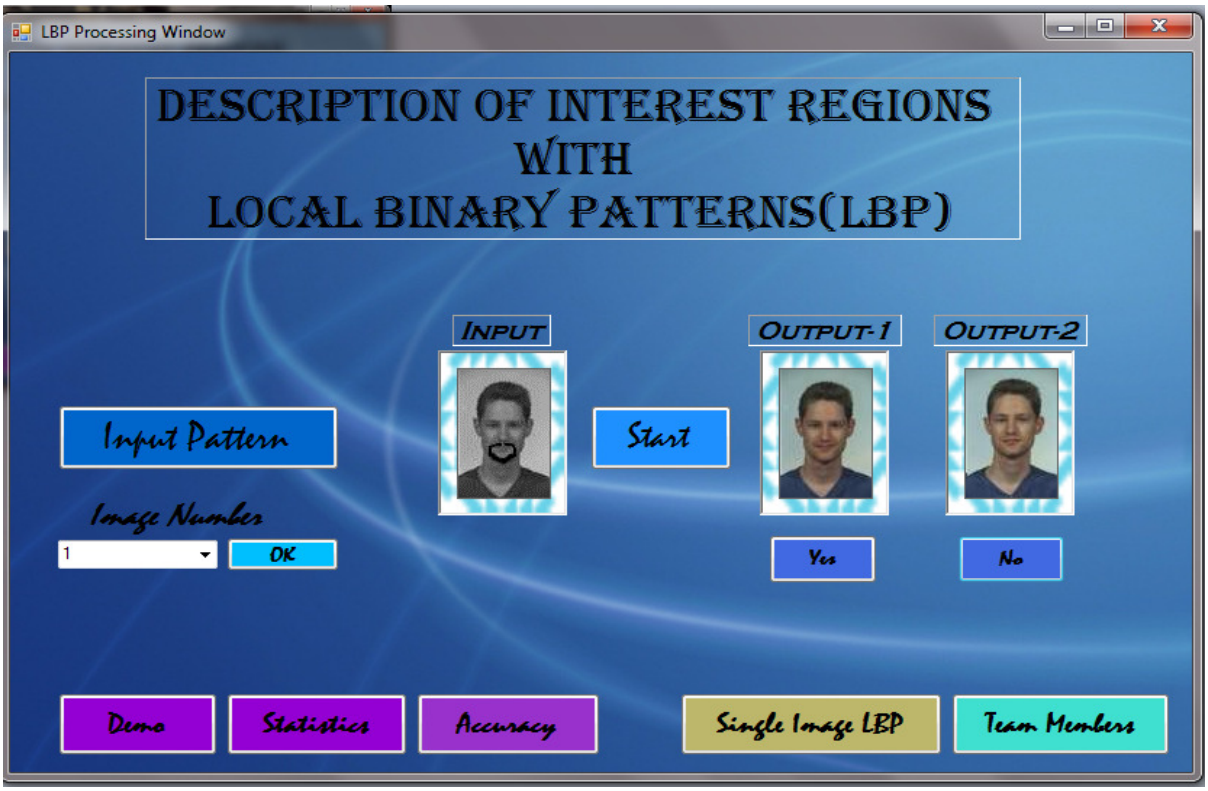

LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
INPUT
OUTPUT-1
OUTPUT-2
Input Pattern
Image Number
1
OK
Start
start the process...
Yes
No
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
INPUT
OUTPUT-1
OUTPUT-2
Input Pattern
Start
Image Number
OK
Yes
No
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
INPUT
OUTPUT-1
OUTPUT-2
Input Pattern
Start
Image Number
OK
Yes
No
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
INPUT
OUTPUT-1
OUTPUT-2
Input Pattern
Start
Image Number
1
OK
Yes
No
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS
WITH
LOCAL BINARY PATTERNS(LBP)
INPUT
OUTPUT-1
OUTPUT-2
Input Pattern
Start
Image Number
1
OK
Yes
No
Demo
Statistics
Accuracy
Single Image LBP
Know who is behind this


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS WITH LOCAL BINARY PATTERNS(LBP)
INPUT
OUTPUT-1
OUTPUT-2
Input Pattern
Start
Image Number
1
OK
Yes
No
Demo
Statistics
Accuracy
Single Image LBP
Team Members


LBP Processing Window
DESCRIPTION OF INTEREST REGIONS WITH LOCAL BINARY PATTERNS(LBP)
INPUT
OUTPUT-1
OUTPUT-2
Input Pattern
Start
Image Number
1
OK
Yes
No
Demo
Statistics
Accuracy
Single Image LBP
Team Members

References

I. Digital Image Processing Using Matlab - Gonzalez Woods & Eddins.

II. A Guide to MATLAB for Beginners and Experienced Users - Hunt Lipsman & Rosenberg.

III. Digital Signal and Image Processing Using MATLAB - Gerard Blanchet & Maurice Charbit.

IV. Software Engineering, a Practitioner Approach 5th Edition by Roger S. Pressman.

V. www.wikipedia.org

VI. http://www.mathworks.com/products/netbuilder/

VII. http://www.mathworks.com/products/image/ (online documentation for MATLAB Image Processing Toolbox).

VIII. http://www.mathworks.com/products/imaq/ (online documentation for MATLAB Image Acquisition Toolbox).

IX. http://www.mathworks.com/products/netbuilder/ (online documentation for MATLAB and .NET Builder Guide).

X. http://www.mathworks.com/products/matlabxl/ (online documentation for MATLAB and Microsoft Excel).

XI. http://ieeexplore.ieee.org/ for specifications of IEEE project.

XII. www.support.mircosoft.com (.NET installation).

XIII. www.developer.com (Deployment overview).

XIV. www.15seconds.com (Deployment overview).