Second Progress Report

System Study

- **Overview of the System**
  - Detection of unique face image amidst the other natural components such as walls, backgrounds etc.
  - Extraction of unique characteristic features of a face useful for face recognition.
  - Detection of faces amongst other face characters such as beard, spectacles etc.
  - Effective recognition of unique faces in a crowd (individual recognition in crowd).
  - Automated update in the database without human intervention.

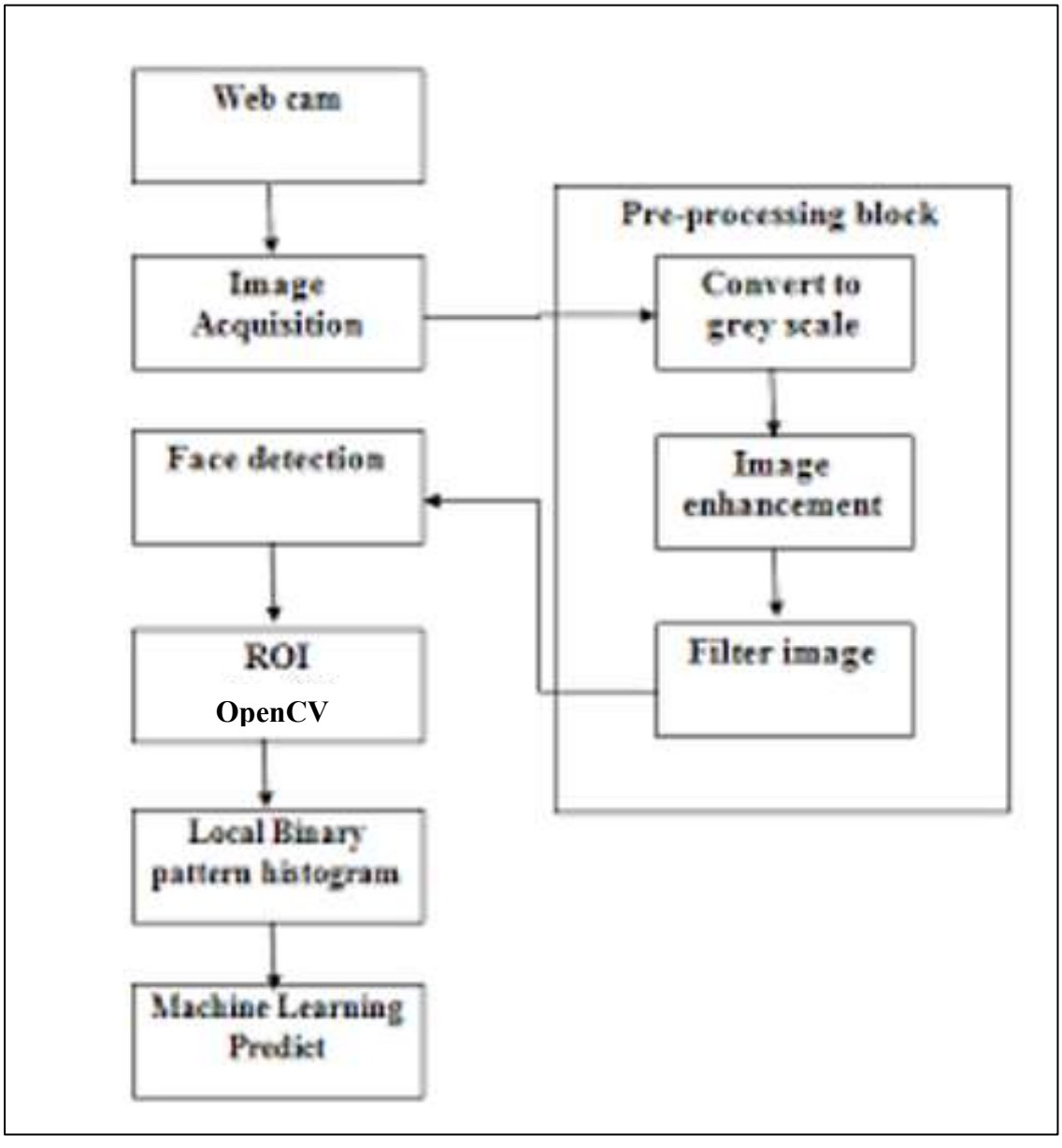
When the system is instantiated it starts processing the image for which we want to mark the attendance. Image Capturing phase is one in which we capture the image. This is basic phase from which we start initializing our system. We capture an image from a camera which is predominantly checked for certain constraints like lightning, spacing, density, facial expressions. We take individuals different frontal postures so that the accuracy can be attained to the maximum extent. This is the training database in which every individual has been classified based on labels. For the captured image, from an every object we detect only frontal faces from viola-jones algorithm which detects only the frontal face posture of an every individual from the captured image. This detects only faces and removes every other parts. Features are extracted in this extraction phase. The detected bounding boxes are further queried to look for features extraction and the extracted features are stored in matrix. For every detected phase this feature extraction is done. Features we look here are Shape, Edge, Color, Wavelet, Auto-Correlation and LBP. Face is recognized once we completed extracting features. The feature which is already trained with every individual is compared with the detected faces feature and if both features match then it is recognised. Once, it recognizes it is going to update in the student attendance database.

- **Existing System Overview**

  The previous approach in which manually taking and maintaining the attendance records was very inconvenient task. Traditionally, student's attendances are taken manually by using attendance sheet given by the faculty members in class, which is a time consuming event. Moreover, it is very difficult to verify one by one student in a large classroom environment with distributed branches whether the authenticated students are actually responding or not. The ability to compute the attendance percentage becomes a major task as manual computation produces errors, and also wastes a lot of time. This method could easily allow for impersonation and the attendance sheet could be stolen or lost. An automatic attendance management system using biometrics would provide the needed solution. The results showed improved performance over manual attendance management system.

- **Proposed System**

The present system of attendance marking i.e., manually calling out the roll call by the faculty have quite satisfactorily served the purpose. With the change in the educational system with the introduction of new technologies in classroom such as virtual classroom, the traditional way of taking attendance may not be viable anymore. Even with rising number of course of study offered by universities, processing of attendance manually could be time consuming. Hence, in our project we aim at creating a system to take attendance using facial recognition technology in classrooms and creating an efficient database to record them.



1. **Capturing Camera:**

Camera is installed in a classroom to capture the face of the student. The camera has to be places such that it captures the face of all the students effectively. This camera has to be interfaced to computer system for further processing either through a wired or a wireless network. In our prototype we use the in-built camera of the laptop.

2. **Image Processing:**

Facial recognition algorithm is applied on the captured image. The image is cropped and stored for processing. The module recognizes the images of the students face.
The whole process requires the following steps:

a. Train Database:

Initially we take facial image of the enrolled students. In our system we have taken 150 images each. This data is used later used in the facial recognition algorithm

b. Face Detection:

The captured image of the classroom is initially scanned to detect faces. This is done using Computer Vision Toolbox by the function vision. CascadeObjectDetector(). This function work on the basis of Viola-Jones algorithm. This algorithm focusses more on speed and reliability.

c. Face Recognition:

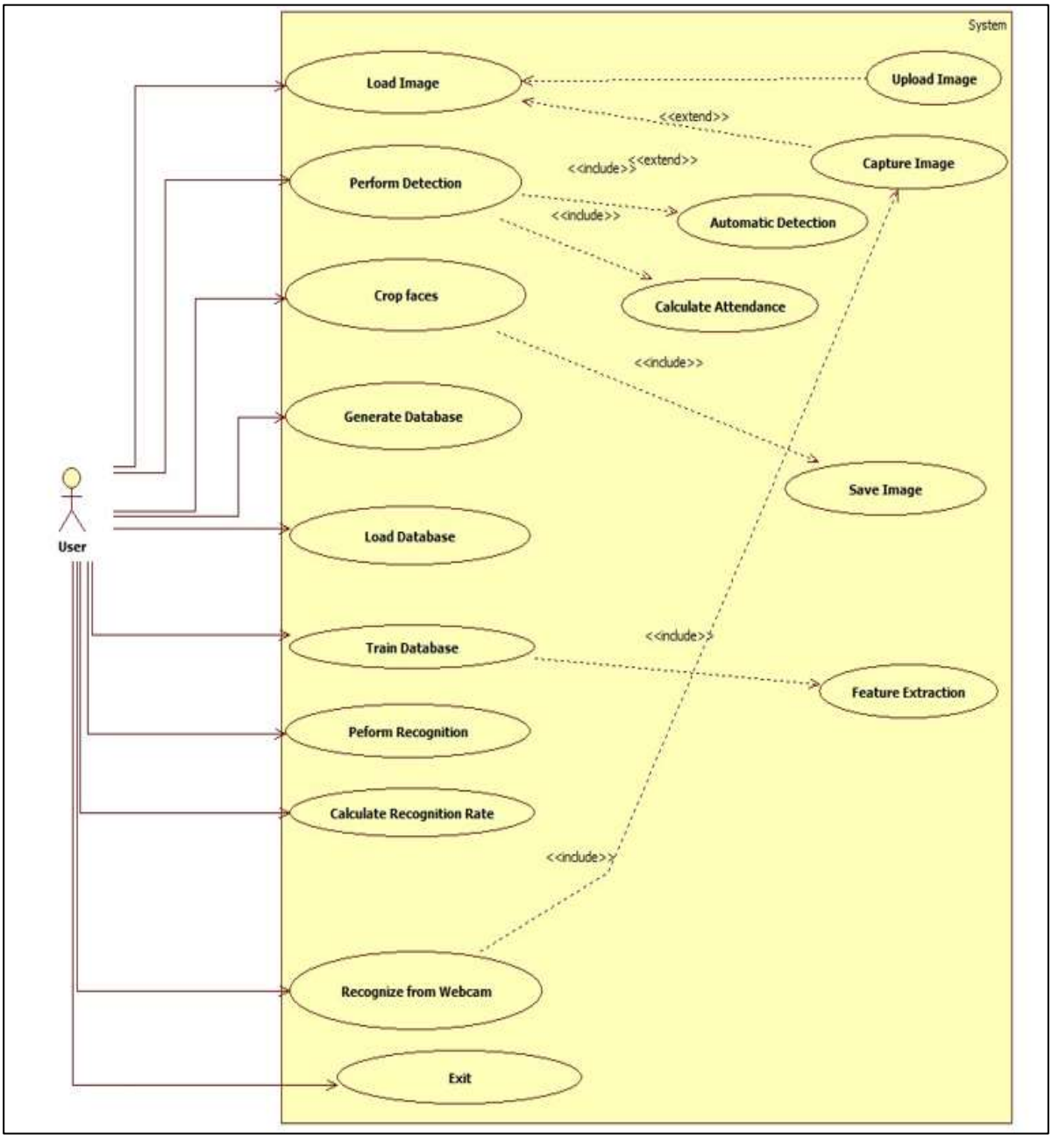
LDA is widely used to find linear combinations of features while preserving class separability. Specifically, it requires data points for different classes to be far from each other, while points from the same class are close. Consequently, LDA obtains differenced projection vectors for each class. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid over fitting (“curse of dimensionality”) and also reduce computational costs

3. **Attendance Recording:**

We use Excel spreadsheet to store the recorded attendance for easy-to-use output format, which is also the software which is familiar to majority of the institution staffs. Using the formatting in the Excel, we can effectively retrieve the information effectively.

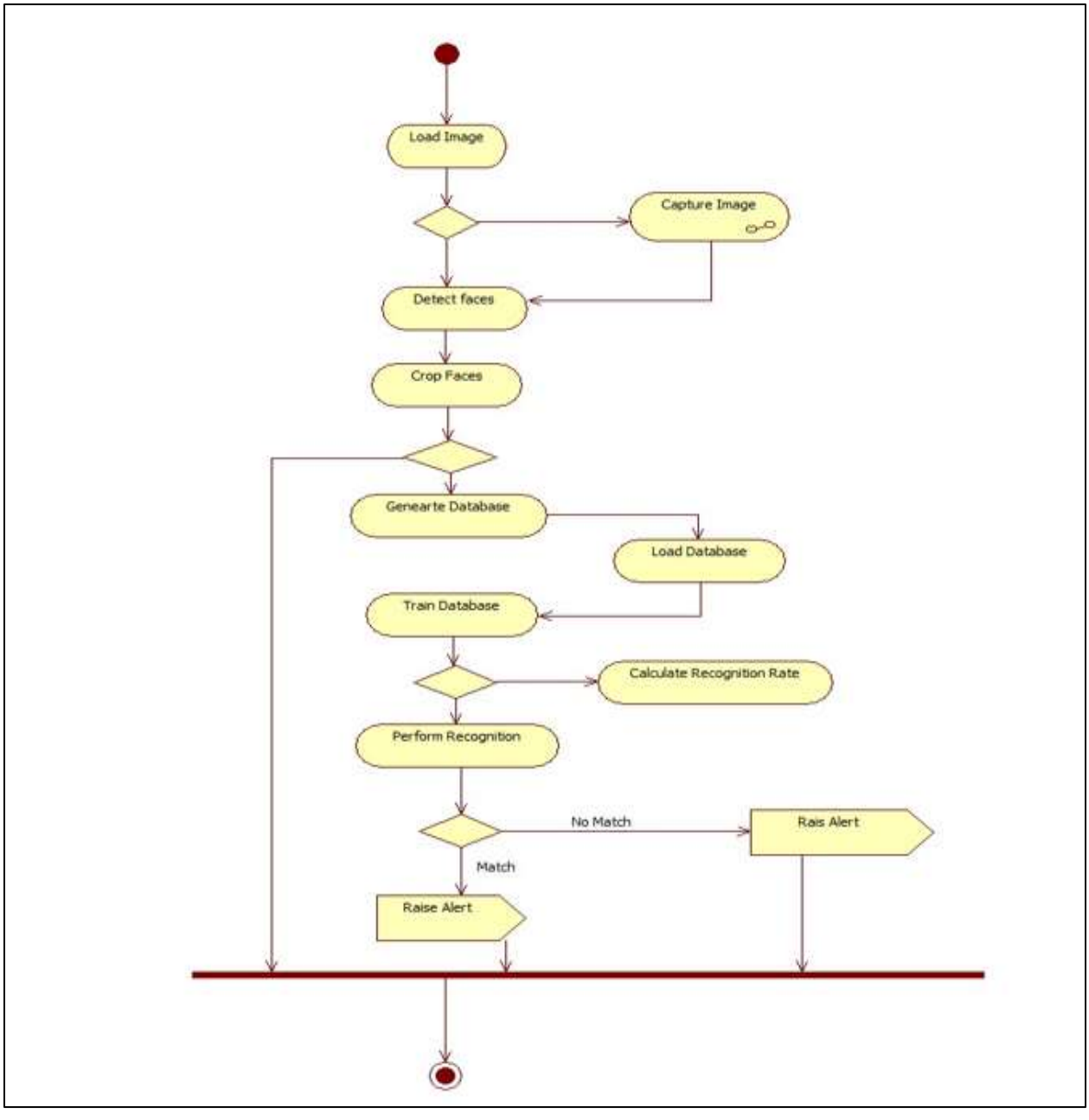
- **Use Case Diagram:**

A use case diagram is a representation of a set of description of actions (use cases) that the system can perform in collaboration with an external factor (user/actor).



- **The Activity Diagram:**

As mentioned above, this will represent the flow of actions that will breakdown most of the interactions into activities in the system.



- **Overall Functionality of the proposed system**

  Our system follows the three tier architecture

  1. **GUI:**

     The GUI(Graphical User Interface) in our project deals with the interface for the user where the user enters the name of the product he/she wants to search. The GUI provides a platform for the user to communicate with the database.

  2. **Trainer:**

     At initial level the system is trained by providing 50-150 images of an individual.

     - Detection :

       Whenever any image is uploaded they are stored for further comparison.

     - Face Recognition:

       Here the detected faces are compared with the trained images from the database. If any of the image is recognized then that id would be marked present in the attendance data sheet.

  3. **Database:**

     Every detected image compares itself with the trained images in this databases i.e retrieval is done and also the attendance sheet is generated using this database i.e updation is done.

- **Implementation**

First, we need to install OpenCV package for python. This package can be downloaded from python website or by using pip install command. Then we use two cascade files haarcascade_frontalface_default.xml and haarcascade_eye.xml files which are available under GNU licence and can be used without permission also. The whole code is implemented in python and need a working webcam to capture images.

- **OpenCV**

OpenCV (Open Source Computer Vision) is a library started by Intel in 1999. It is a classifier that helps in processing the images. The main focus is mainly on image processing and can be implemented on the latest algorithms. It comes with a programming interface to Python. It is used under a BSD license to be used in academic projects. It comes with the Face Detection class for face detection. To perform the face detection, the following modules need to be imported.

1. cv2 – It is an OpenCV package object or module and has the functions image processing.
2. numpy – The image will be stored in numpy arrays.
3. sys–to perform console related functions like taking input from the console.

The first step will be detecting the face in each image and use the HaarCascade which is given by OpenCV. The haarcascadesare present inOpenCV and can be found in the location /data/haarcascades> directory of theOpenCV installation. the haarcascade_frontalface_default.xml and haarcascsade_eye.xml are used to detect the face and eyes respectively. The xml files will be loaded using cv2.CascadeClasssifier function which takes the path to it.

- **Haarcascades**

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

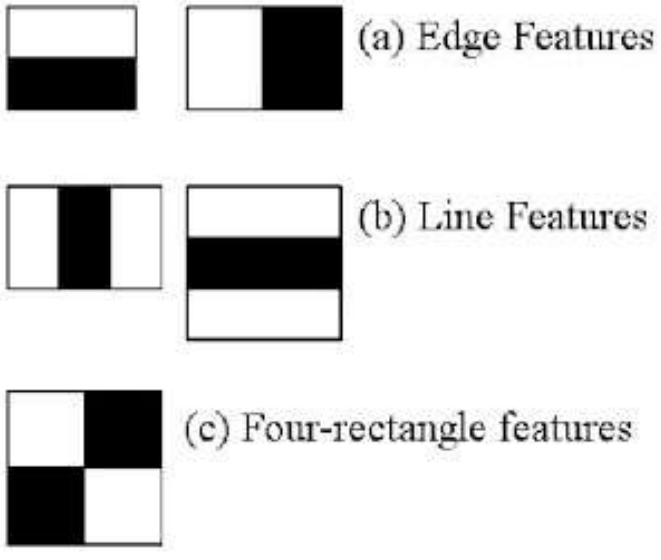
It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm has four stages:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

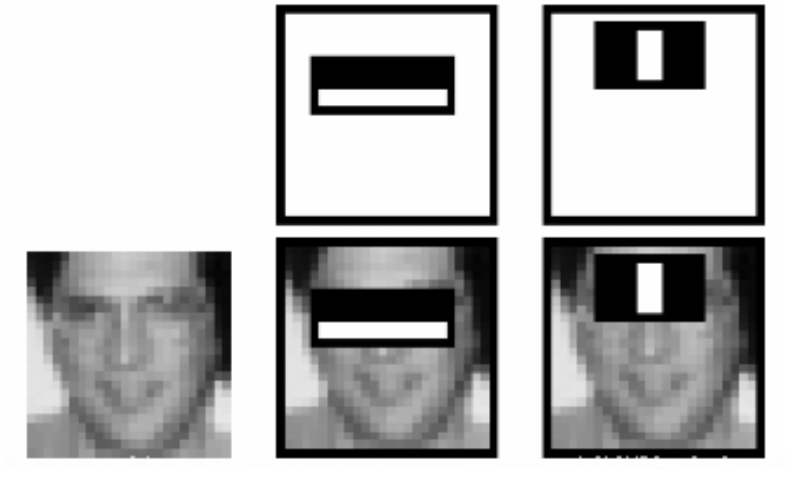
Lets take face detection as an example. Initially, the **algorithm needs a lot of positive images of faces and negative images** without faces to train the classifier. Then we need to extract features from it.

First step is to **collect the Haar Features.** A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.



Integral Images are used to make this super fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.



➢

➢ **Local Binary Pattern** (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number..

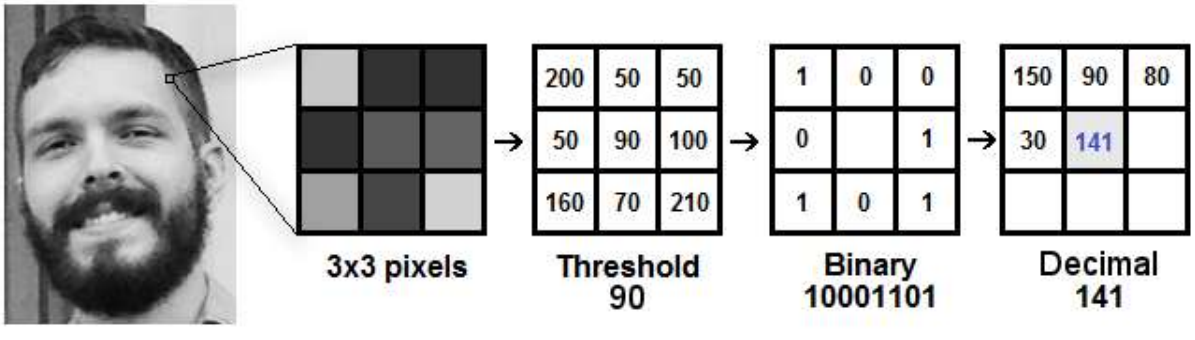
1. Training the Algorithm:

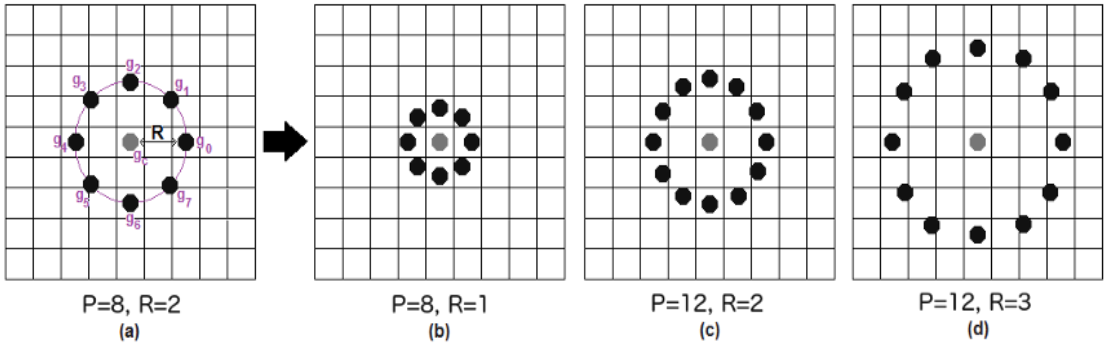
First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give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

2. Applying the LBP operation:

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

The image below shows this procedure:





3. Extracting the Histograms:

Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image

The LBPH algorithm is pretty much it.

4. **Performing the face recognition**: In this step, the algorithm is already trained.

- **Requirements and Analysis:**

This stage will discuss the various tools that are used to achieve the project goals and objectives.

Functional Requirements:

Functional requirements are features that the system will need in order to deliver or operate. In the case of this project, it was important to gather some requirements that will be needed to achieve the objectives set out previously. With client (user) story a use case analysis was implemented which resulted in the following functional and non-functional requirements were captured. The functional requirements have been gathered from the user story developed from the minutes collected during meetings with the client and are outlined here.

- Capture face images via webcam.
- Faces on an image must be detected.
- Store the faces to a folder.
- Load faces on database.
- Train faces for recognition.
- Perform recognition for faces stored on database.
- Compute recognition rate of the system.
- Perform recognition for each face by Face Detector.
- Display the name of the output image above the image in the plot area.

Non-Functional Requirements:

Non-functional requirements are set of requirements with specific criteria to judge the systems operation. These requirements have been collected based on the following after meetings with the client. They cover ease of use to the client, security, support availability, operational speed, and implementation considerations. More specifically:

- The user will find it very convenient to take photos.
- The user will inform the students when taking a photo with clear instructions on how to position their faces.
- The system is very secure.
- The system will have a response time of 10 seconds.
- The system can be easily installed.
- The system is 100% efficient.
- The system must be fast and reliable.