Model View Controller Report

(Smart Attendance System with Face Recognition using OpenCV)

Shumbul Arifa, **Keerti Chaudhary**

Students of the Department of Computer Science and Engineering, National Institute of Technology Karnataka,

Surathkal, Karnataka, India

shumbul.181co152@nitk.edu.in, keerti2001.kc@gmail.com

Abstract

In human interactions, the face is the most significant factor because it contains important information about an individual. All humans will acknowledge people from their faces. The proposed solution is to develop an operating prototype of a system that may facilitate class attendance management for the lecturers within the lecture rooms by detecting the faces of scholars from an image taken in a classroom. The database can store the faces of scholars, once the face of the individual matches with one in all the faces held within the database then the attendance is recorded. In recent years, analysis has been dispensed and face recognition and detection systems are developed. The model-view-controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

Keywords: MVC, Face Detection, Face Recognition, Attendance System, Data Flow Diagram

1 Introduction

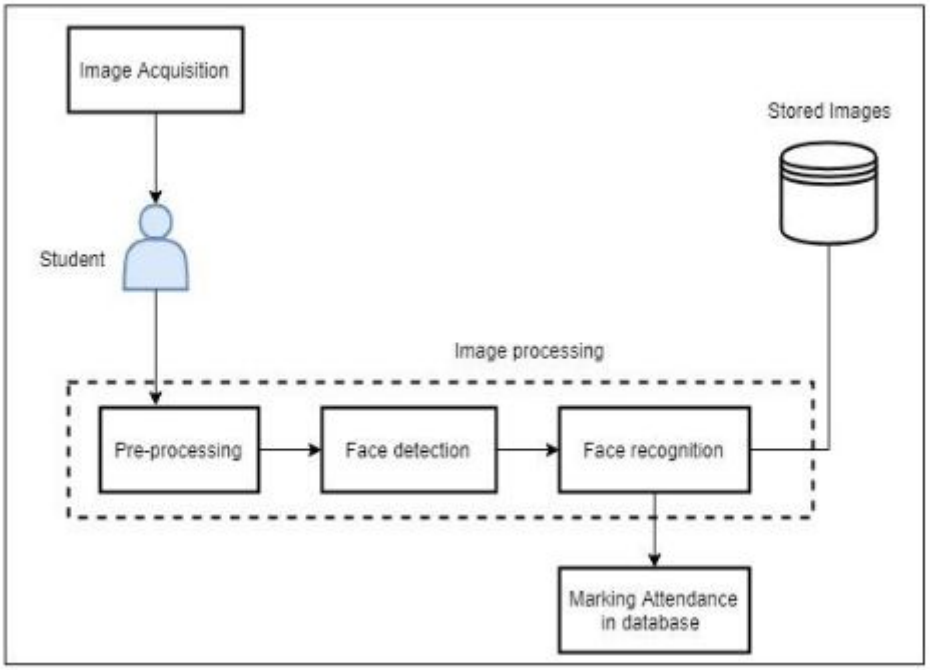
Face detection involves separating image windows into two classes; one containing faces (turning the background (clutter). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin color, and facial expression. The problem is further complicated by differing lighting conditions, image qualities, and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height). After taking the picture the system will compare the equality of the pictures in its database and give the most related result.

2 Software Pattern- Model View Controller (MVC)

The proposed approach focuses on the use of facial recognition of a student mark attendance. The main objective of this system is to capture and identify/verify the identity to ensure the attendance is entered. Thus, this system is based on Model-View-Controller (MVC) architecture to store the captured facial images of the user, and his user details through a recognition module that handles the following tasks:

• Biometric data construction. In this step, the system acquires the face image of the visitor and stores it with his details.
• Comparison between extracted features of the requested image and those of reference images (obtained in the registration phase) in the phase of authentication.

• Redirection toward the controller in the case of a positive authentication. The remote server sends the decision (the result of comparison) to the client. Each time, this system demand using the same module forms a plug-in for taking facial images:
• Registration of a new user by extracting his information (user details, face images).
• Automatic updating of attendance for the verified face.



MVC Software Pattern for attendance system using face recognition

1. Model
   This level is extremely vital because it represents the info to the user. This level defines wherever the application's knowledge objects are held on. The model doesn't recognize something concerning views and controllers. So, whenever there are changes done in the model it'll mechanically give notice to observers that the changes are created. The model could also be one object or a structure of objects.

2. View
   A view may be a visual illustration of the MVC model. This level creates an interface to point out the particular output to the user. However, a view won't show something itself. It's the controller or model that tells the reader what to show to the user. It conjointly handles requests from the user and informs the controller. A view is connected to its model and gets the info necessary for the presentation by asking sure queries. Sometimes, it conjointly updates the model by causation applicable messages. of these queries and messages, are sent back to the model in such a straightforward word that may simply perceive the information sent by a model or a controller.

3. Controller
   The controller may be a level that acts as the brain of the whole MVC system. A controller conjointly acts as a link between a user and a system. It provides the user with the input by providing applicable views to present it fitly on the screen. The controller understands user output, converts it into suitable messages, and passes a similar view.

2.1 Reference Data Construction

The main elements utilized in the implementation approach are open supply computer vision library (OpenCV). One of all OpenCV's goals is to produce a simple-to-use laptop vision infrastructure that helps folks build fairly refined vision applications quickly. The OpenCV library contains over five hundred functions that span several areas of vision. The first technology behind Face recognition is OpenCV. The user stands ahead of the camera keeping a minimum distance of 50cm associate degreed his image is taken as an input. The frontal face is extracted from the image then regenerated to grayscale and kept. The Principal part Analysis (PCA) rule is performed on the pictures and also the eigenvalues are kept in the associate degree XML file.

Face Detection:

Start capturing images through a web camera of the client-side:

```
Begin:
1.    Pre-process the captured image and extract face image
2.    Calculate the eigenvalues of the captured face image and compare it with
eigenvalues of existing faces in the database.
3.    If the eigenvalue does not match with existing ones, save the new face image
information to the face database (XML file).
4.    If the eigenvalue is matched with the existing one then the recognition step
will be done.
End
```

1. Compute the mean feature vector

   $\mu = \frac{1}{p}\sum_{k=1}^{p} x_k$, where, x_k is a pattern ($k = 1$ to p), p = number of patterns, x is the feature matrix
2. Find the covariance matrix

   $C = \frac{1}{p}\sum_{k=1}^{p}\{x_k - \mu\}\{x_k - \mu\}^T$ where, T represents matrix transposition
3. Compute Eigen values λ_i and Eigen vectors v_i of covariance matrix

   $Cv_i = \lambda_i v_i$ $\quad (i = 1, 2, 3,....q)$, q = number of features
4. Estimating high-valued Eigen vectors
   (i) Arrange all the Eigen values (λ_i) in descending order
   (ii) Choose a threshold value, θ
   (iii) Number of high-valued λ_i can be chosen so as to satisfy the relationship

   $\left(\sum_{i=1}^{s}\lambda_i\right)\left(\sum_{i=1}^{q}\lambda_i\right)^{-1} \geq \theta$, where, s = number of high valued λ_i chosen

   (iv) Select Eigen vectors corresponding to selected high valued λ_i
5. Extract low dimensional feature vectors (principal components) from raw feature matrix.

   $P = V^T x$, where, V is the matrix of principal components and x is the feature matrix

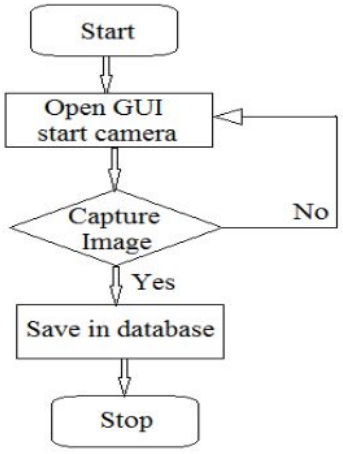
PCA Pseudocode

Face Recognition

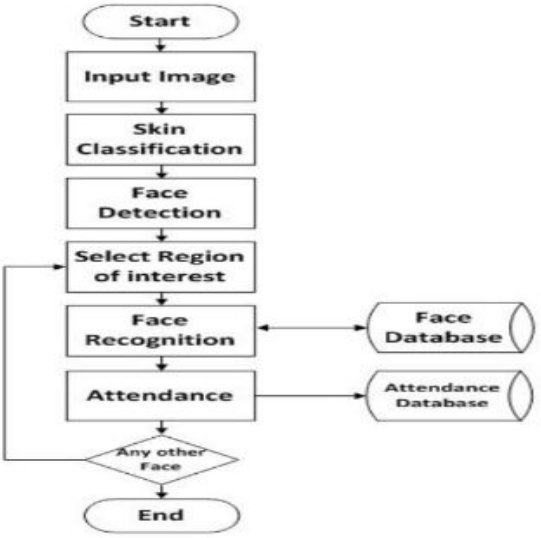
When a user requests for recognition the frontal face is extracted from the captured video frame through the camera. Using the PCA algorithm the following steps would be followed in for Face Recognition:

```
Begin:
1.    Find the face information of the matched face image from the database.
2.    Update the log table with corresponding face image and system time that makes
the completion of attendance for individual students.
End:
```

This section presents the results of the experiment conducted to capture the face into a greyscale image of 50x50 pixels. The eigenvalue is re-calculated for the test face and it is matched with the stored data for the closest neighbor.



Face Detection



Face Recognition

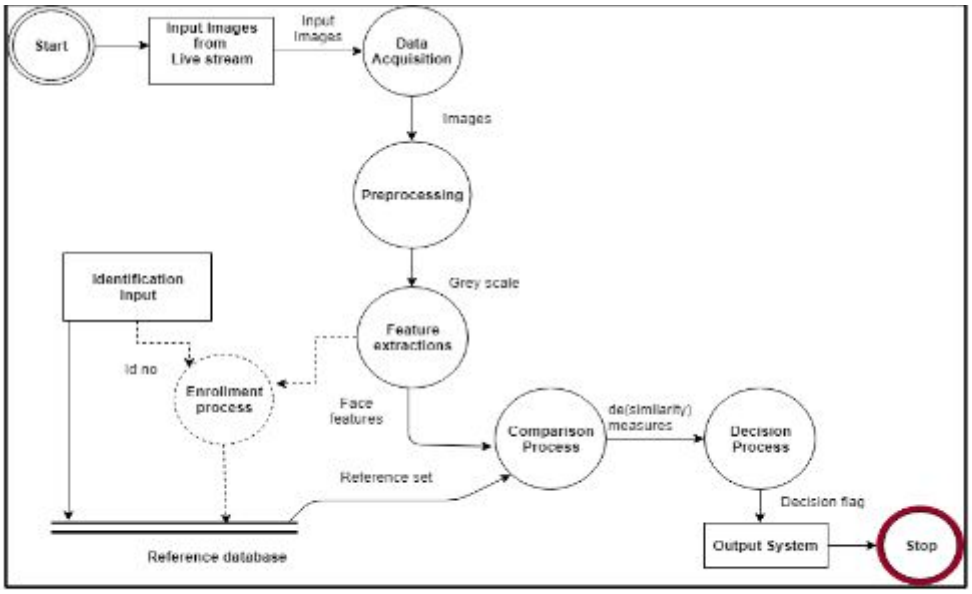
2.3 Attendance Management

The extracted features are compared to those stored in the database and decisions are made according to the sufficient confidence in the match score. Classification is performed by comparing the feature vectors of the face library members with the feature vector of the input face image.

The input facial image issued by the user is compared with a set of images of the Training set. The query image is associated with the closest image of the set of the training images that has the smallest calculated Euclidean distance, and after comparing it with a studied threshold, the decision determines the degree of similarity for giving a rejection response or acceptance.

Data Flow Diagram

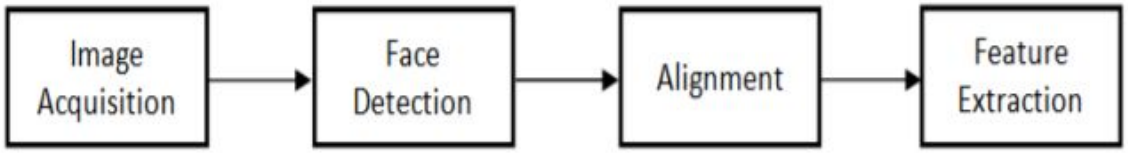
These images are converted to greyscale as LBPH works with images in greyscale. From the greyscale images, features of the face are extracted. Features refer to the gradients in the face. The features are then compared with existing records to check if there is a match. If the face matches it is displayed and output is in the form of attendance being marked for the person whose face was recognized.



Data flow diagram

Phases of Face Recognition System (FRS)

In this model, we proposed an authentication scheme using a face recognition system (FRS).



Face Recognition System (FRS)

Let’s elaborate on each phase in detail and understand it.

1) Image Capture – It is the step where an image of the person is captured wherein his or her face is visible. In the case of 2D facial recognition, a digital camera with a normal resolution is needed.

2) Face Detection – Face detection involves identifying the face in the captured image. In simple words, only the face of the person is seized & all other parts of the images are eliminated.

3) Alignment – The face captured in the camera may not be completely perpendicular to the camera and hence the alignment needs to be determined and compensated so that it is ready to use the recognition process.

4) Feature Extraction – Feature extraction involves a process of measuring various facial features and creating a facial template, for matching and identification.

Conclusion

MVC is a software architecture - the structure of the system - that separates domain/application/business (whatever you prefer) logic from the rest of the user interface. It does this by separating the application into three parts: the model, the view, and the controller.MVC architecture is useful for faster application development. It is convenient for multiple developers to collaborate and work together. It helps in effortless updating of the application and debugging as we have multiple levels properly written in the application. It also satisfies the Single Responsibility Principle of SOLID.

References

[1] Visar Shehu, Agni Dika, Using Real-Time Computer Vision Algorithms in Automatic Attendance ITI 2010 32nd Int. Conf. on Information Technology Interfaces, June 21-24, 2010, Cavtat, Croatia.

[2] M. Turk and A. Pentland (1991). "Face recognition using eigenfaces". Proc. IEEE Conference on Computer Vision and Pattern Recognition.

[3] Yohei Kawaguchi, Tetsuo Shoji, Weijane Lin, Koh Kakusho, Michihiko Minoh. Face Recognition-based Lecture Attendance System.

[4] Jalled, Fares. (2017). Face Recognition Machine Vision System Using Eigenfaces.

[5] Gaddam, Sarath & Ramesh, Nvk. (2016). Attendance Management and User Security System based on Eigenfaces Algorithm using Raspberry pi 2 and Ethernet. Indian Journal of Science and Technology.