Intelligent Attendance and Participation Monitoring

Neeraj Godbole, **Electrical and Computer Engineering, Carnegie Mellon University**
Kevan Dodhia, **Electrical and Computer Engineering, Carnegie Mellon University**
Omar Delen, **Electrical and Computer Engineering, Carnegie Mellon University**

I. Introduction

For our project, we are building an intelligent attendance and participation system. The system will be designed for a classroom environment where professors my want to keep track of student attendance and participation without having to do this manually. The system will be computer vision and machine learning based and will span the Software System and Signals & Systems ECE areas.

In order to track attendance and participation we will set up a camera in a classroom and feed this footage into a laptop that will run all the software. To keep track of attendance, we will detect when students enter and leave a class room. We will use facial recognition to identify these students. In order to keep track of attendance, we will detect when students raise their hands to speak.

II. Design Specification

The application demands the following: a system that is able to track and record student attendance and participation in a classroom setting.

A camera-based system naturally arises where the camera records the class and provides video input to a laptop (or other PC) that performs all of the monitoring through an application/program.

The application running on the laptop can then address the problem by performing the following three functions:

- Detecting faces and performing facial recognition on identified faces, matching them against the database of members of the class.
- As part of recognition, also be able to detect strangers, i.e. faces detected that are not in the database
- Upon detecting faces, detect and identify raised hands. This is a means of tracking participation during a class.
- As a reach goal, detect mouth movement to identify when a member of the class speaks, matching the mouth movement to the appropriate face (easy once facial recognition is achieved).
- Consolidating this information into a table that shows attendance and displays participation data for each member of the class.

III. Approach

In this section we provide an overview of the proposed system along with the system block diagram.

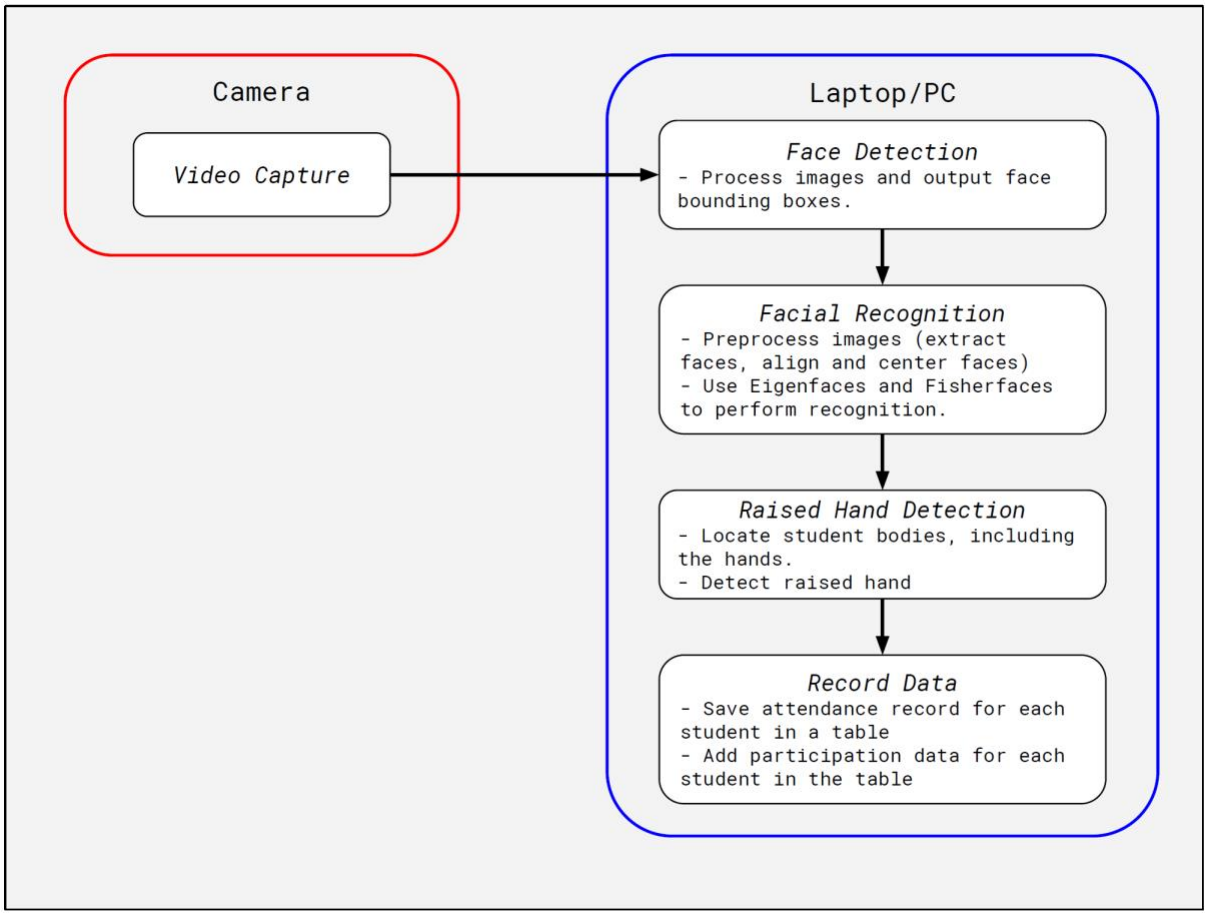

Fig. 1: Overall system block diagram

From the block diagram above, the flow of information is fairly clear.

Video capture is input to the Face Detection module that processes the video in real time computing bounding boxes for the faces in the camera's field of view.

The bounding box data is passed as input to the Facial Recognition module, which uses PCA and LDA (mostly) to perform facial recognition on the detected faces, associated each detected face with the appropriate matching student in the database, and marking absence of a face (i.e. student not present). This module also handles stranger detection, recognizing that a person is not part of the class if the detected face is not matched with any in the database.

The results of the facial detection/recognition are then passed on to Raised-hand detection module, which uses this along computer vision-based foreground extraction techniques to locate student bodies. Once we have localized student bodies, we will define a shape-like descriptor to represent the human bodies based on the SIFT descriptor. Finally, we will learn a classifier to recognize the student gestures.

These results, along with the facial detection/recognition results will be sent to a basic frontend application that will display the updated attendance and participation records.

IV. Implementation Detail

In this section we provide detail on our implementation plans for each of the modules in our proposed solution, discussing the experimentation and questions that allowed us to arrive at our proposed implementation plan for each module.

A. Face Detection Module

We will be implementing the well-known Viola-Jones algorithm in order to detect faces. We will be implementing it from scratch. The algorithm can be summarized into the following components:

1. Haar Features construction, which are used to match regularities in human faces.
2. Integral image, which allows the features used by the detector be computed very quickly.
3. AdaBoost training, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers.
4. Cascading Classifiers, which results in a classifier that consist of several simpler classifiers that are applied to a region of interest until at some stage the candidate is rejected, or all the stages are passed.

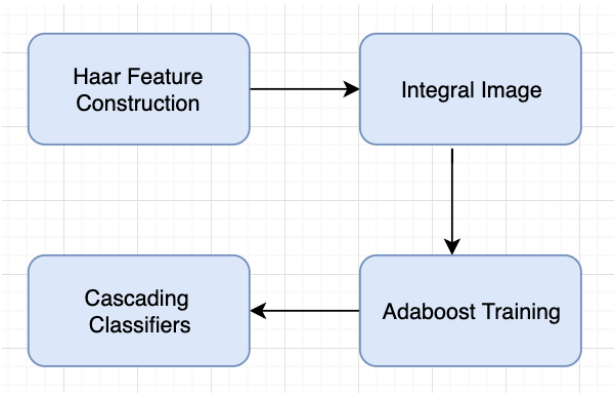

Fig. 2 – Block diagram outlining the Viola-Jones Facial Detection Algorithm

Upon running this algorithm, we will get individual bounding boxes around all faces in a frame. The faces will then be cropped out and used for facial recognition.

B. Facial Recognition Module

The recognition module comprises many components. Overall, it consists of preprocessing, PCA and LDA based recognition and stranger detection.

The input to the module is the snapshot images provided by the detection module along with face bounding boxes/rectangles for each image. The output of the module the associated person name for each face.

Preprocessing

The preprocessing section deals with standardizing the images that are fed to the recognition algorithm and rejecting detected faces that are not suitable for recognition (bad data). This mainly involves aligning and centering each image and setting consistent dimensions. Aligning, centering of the images and rejection of bad images will be done using eye detection through the *OpenCV* and *dlib* modules in Python.

The procedure for alignment and centering is simple:

1. Use dlib to extract facial landmarks for the face with bounding rectangle provided.
2. Compute centroid of left and right eye landmarks and use these to compute angle from horizontal and position from center.
3. Combine the angle and position into an affine transformation matrix and rotate the image using *OpenCV*.

Recognition

Our method of recognition involves using PCA, LDA and K-means clustering.

We begin with using PCA and eigenfaces for recognition. PCA allows you to take a higher dimensional space and project it into a smaller space. This is useful because images are $m \cdot n$ dimensionality (where m is width and n *is* height) and we cannot compare and cluster images for recognition with such high dimensionality. So PCA develops a set of basis vectors, eigenvectors of a covariance matrix to be precise, which span the space of the face.

The next part of recognition is projecting a new image onto the subspace and clustering the data. We initially tried local PCA using the following algorithm:

1. Perform PCA on each class (person), thus deriving a set of k eigenvectors spanning the space of that person's face
2. In the k-dimensional space of the eigenvectors, we plot each training image for every class. Each image for person P is a point whose coordinates are the basis coefficients of that image in person P's set of k eigenvectors.
3. We do K-means clustering.
4. Given a new arbitrary image, we can see in which eigenface space the image best "fits in" (i.e. minimize the measured E2 distance between the point and centroid across potential class eigenvectors).

This led to poor results when used alone for clustering as shown below.

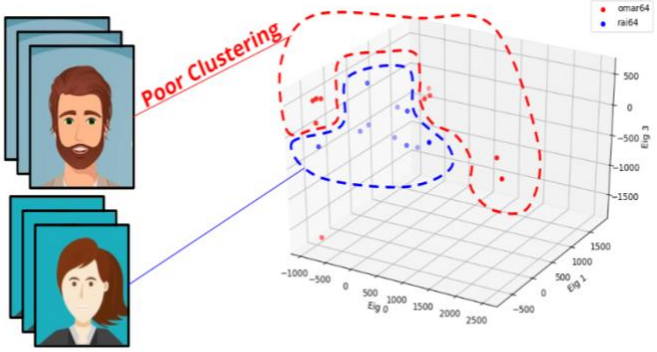

Fig. 3– Local PCA results for two people (top image is red data points, while bottom image is blue data points)

As a result, we shifted to Global PCA, which is used to project the face data, followed by LDA to aid clustering.
While our algorithms for Global PCA and LDA have not been fully developed yet, we can provide the initial outline for each algorithm and show results that have been obtained through early implementation. The focus of our recognition work over the rest of the implementation phase is to refine and modify the recognition algorithm to achieve high accuracy.

The Global PCA algorithm has the following outline:

1. Perform PCA on all classes, thus deriving a set of *k* eigenvectors spanning the space of all the training data.
2. In the *k*-dimensional space of the eigenvectors, we plot each training image for every class. Each image for person P is a point whose coordinates are the basis coefficients of that image in person P's set of *k* eigenvectors.
3. We do K-means clustering.
4. Given a new arbitrary image, we predict its class by finding the closest class centroid.

We extend this using LDA to finally lead to the following algorithm:

1. We do Global PCA with N total images across C classes. To prevent a singularity in the LDA algorithm, we must project our D dimensional images (where $D = k^2$ for *k* by *k* images) into a space with less than N – C dimensions
2. We do LDA on this lower space, producing less than C – 1 independent eigenvectors.
3. We project all of the training data onto this C – 1 space and do K-means clustering.
4. Given a new image, we find the closest cluster centroid to it after projecting the image into the C – 1 dimensional space.

Current results for Global PCA and LDA are shown below:

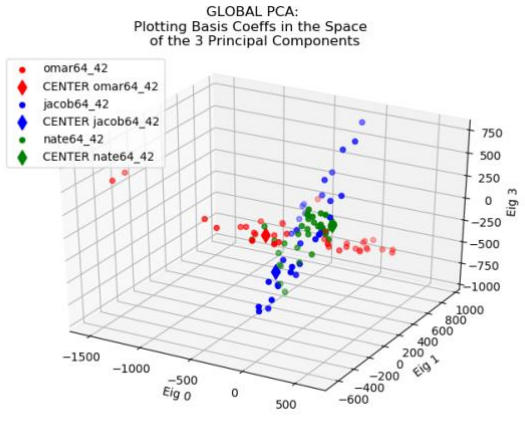

Fig. 4 – Global PCA results for different data people with each axis representing an eigenvector of the 3 selected, and each color represents a different person.

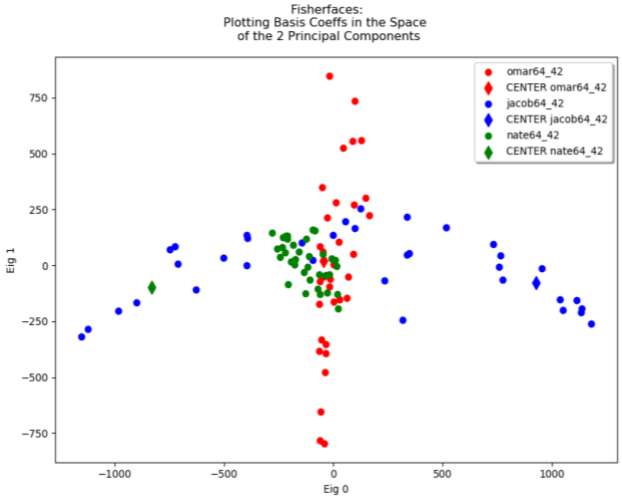

Fig. 5– Fisherfaces (LDA) results obtained. Once again, each color of data points is associated with a different person.

We believe that experimentation with eigenvectors and preprocessing of images will significantly improve the current results that we have obtained for facial recognition and that will be a core focus for the initial part of the implementation.

Stranger Detection

We devised two methods for detecting strangers.

1. Use an image's E2 K-means distance. This distance is its distance to the nearest centroid. If this distance is larger than some threshold, we consider the image as that of a stranger. We decide the threshold value by taking it as the E2 K-means distance of the training data (e.g. 75% of training data images had E2 distance < 500 so the threshold is 500)
2. Similar threshold to before except we use the reconstruction mean-squared error as a metric.

This method seemed promising, but it did not work out. Below is a chart summarizing the source of inaccuracies we detected in our training and testing. The False Positive part of the stacked bar represent errors where our k-means correctly predicted the identity but because of our thresholding we designated that image as a stranger.

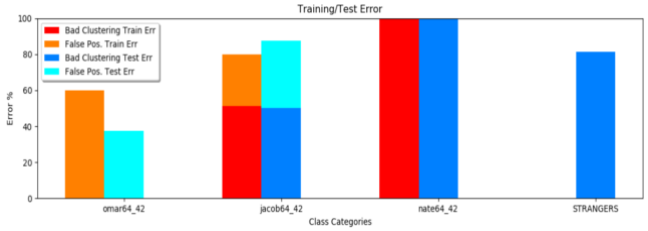

Fig 6– Chart depicting source of inaccuracies, with dark blue as poor clustering test error, light blue as false positive test error and red and orange depicting the same for train error respectively. Each set of bars represents a different person

In order to understand why, we visualized how the E2 and MSE metrics for training data and stranger data compared (below). Our implicit hypothesis was that stranger images would have larger E2 distances and larger MSE's than the

training data. This was not the case unfortunately. We are still working on a solution to the problem of detecting strangers.

C. Participation

Participation will involve detecting when students raise their hands to speak. For this we have two potential approaches. The first involves both computer vision and machine learning techniques. The approach is as follows:

1. The foreground parts that only contain human bodies are first segmented. In order to do this, we will first extract the background using techniques such as Gaussian Mixture Models and temporal differencing. The remaining foreground areas, however, may contain objects such as bags etc. Thus, at this point, we will use the facial detection/recognition results from earlier to locate student bodies in the classroom.
2. Once we have localized student bodies, we will define a shape-like descriptor to represent the human bodies based on the Scale-invariant feature transform (SIFT) descriptor.
3. Finally, we will learn a classifier to recognize the student gestures. We will use Support Vector machines for this task.

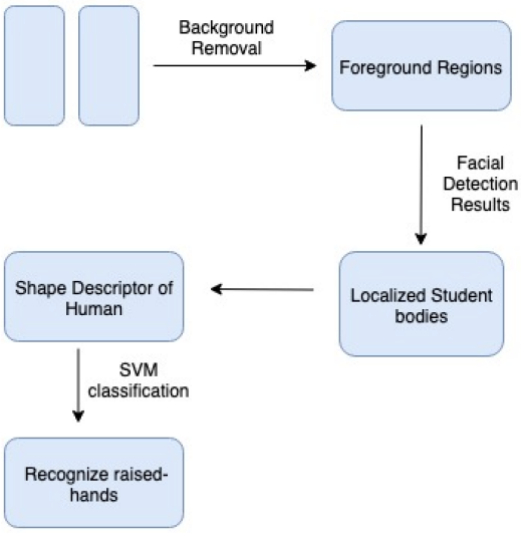

Fig. 7 – Block diagram outlining our first approach to raised-hand detection

A second approach we will consider is purely computer vision based and may be simpler to implement. It may, however, not lead to as promising results as the approach mentioned above. Thus, this second approach is currently only a backup option in case we get stuck with the first approach. The second backup approach is as follows:

1. The image is converted to a Hue-Saturation-Value (HSV) representation, meaning that now for each pixel in the image there are three values – Hue, Saturation and Value. Since different people have different hand colors, we cannot rely on the intensity of any one of the colors in order to extract the hand and its features. Therefore, the RGB representation is converted to a Hue-Saturation-Value (HSV) representation.
2. Facial detection results from earlier will be used to find the coordinates of the face. The face will be subtracted from the image.
3. The image is then sent for blob detection specifying a skin-tone thresholding value. This thresholding skin-tone value is based on that of the head region that was found earlier. Finally, we will use the blob detection results to determine if the hand is raised or not.

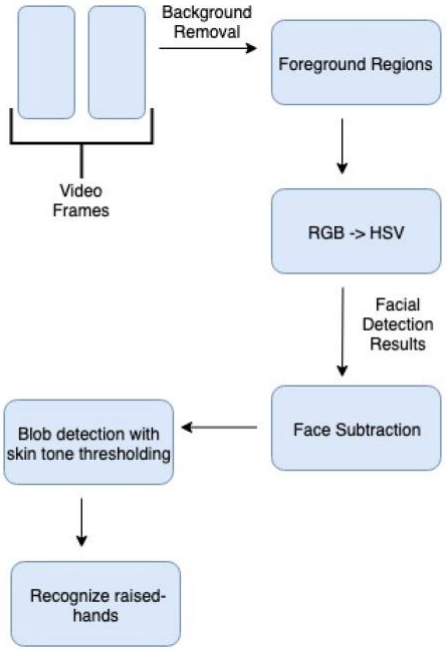

Fig. 8– Block diagram outlining the purely vision-based approach to raised-hand detection

D. Summary

In summary, the various components of the implementation and whether they will be library handled or developed from scratch are below:

Detection
We will develop Viola/Jones algorithms from scratch, with the help of OpenCV and numpy.

Recognition
We will develop all algorithms (PCA, K means) from scratch with the exception of basic math functions like eigenvalue computation, matrix multiplication. For eye detection, we will use OpenCV and Dlib.

Participation
We will develop most of this from scratch, with the help of OpenCV for foreground extraction and using the SIFT descriptor.

V. Metrics and Validation

The testing for our project will be two-fold, automated and manual. Since all the separate algorithms can be developed and tested individually, we will use automated testing for each of them as they are developed. The metrics for each algorithm is detailed in Fig.9.

	Input Image	Output	Metric	Desired accuracy
Detection	Known # people	How many people? Where?	% of **correct** detections	>70%
Recognition	Known people IDs	Who is in picture?	% of correct recognitions made	>70%
Hand Raising	Known # hand raises	How many hands raised?	% of correct hand raises detected	>75%

Fig. 9 – Metrics and input/output details for automated testing

Once we have fused all the algorithms together we will begin manual testing. We will set up small rooms with between 3-8 students for 5mins. Students will be asked to raise their hands and leave/enter the classroom at their own will. We will measure the accuracy for attendance and participation records displayed by the system during that time period.

VI. Project Management

A. *Member Responsibilities and Scheduling*

For the broad tasks of the assignment the following division of labor been currently planned:

1. Facial Recognition algorithm - Neeraj and Omar
2. Raised hand recognition algorithm - Kevan
3. Prototype mouth detection - Kevan
4. Final mouth detector - All
5. Integration of Facial Recognition with an Attendance system – All

The planned schedule of work is shown on the right column of this page.

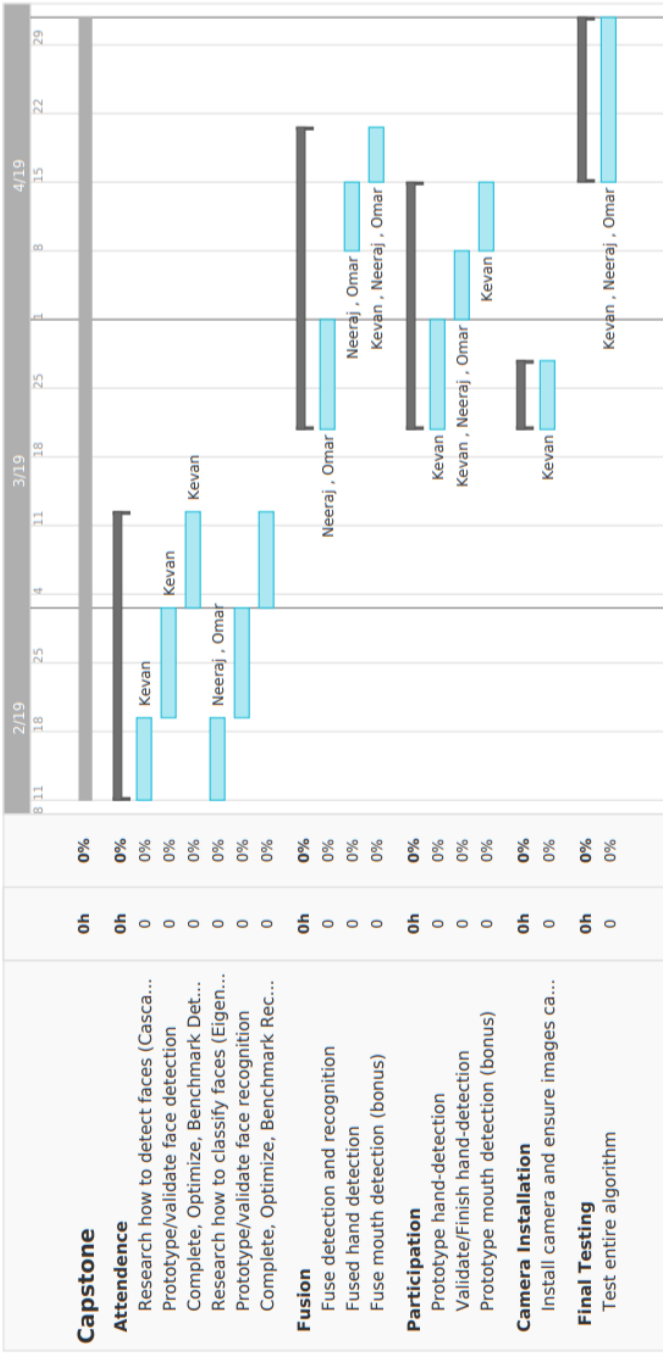

Fig. 9 – Semester long schedule of tasks to complete

B. *Tools Used, Bill of Materials*

The overall budget for the project is essentially $0 at the moment since the main piece of equipment needed is a camera which will be provided by Prof. Marios Savvides. The camera specification is 5 MP, with zoom capability.

Otherwise all of the other requirements are software requirements which can all be satisfied using free and open source software.