THE UNIVERSITY OF MELBOURNE

Investigation of Facial recognition algorithms for use with IP cameras

Author:
Tural Farhadov (601261)

Supervisors:
Dr. Edmund Kazmierczak,
Zaher Joukhadar,
Ken Clarke,
Dr. Chamil Jayasundara

Degree name: Master of Information Technology

Subject code and name: COMP90019 Distributed Computing Project

Credit points: 25 points

Project type: Research

Date: November 3, 2015

Declaration of Authorship

I declare that :

- this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.
- where necessary I have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department.
- the thesis is 8068 words in length (excluding text in images, table, bibliographies and appendices).

Sign: [illegible]

Date: November 3, 2015

Abstract

In this work we have investigated various face recognition algorithms suitable for use with IP cameras. Different types of challenges that are brought by IP cameras, such as illumination, low image quality, pose variance, tilt variance, have been addressed. An environment for experiments has been created and algorithms are experimented over face databases and robustness towards articulated challenges has been researched. Three main facial recognition algorithms are chosen and investigated for their advantages and disadvantages for use in a facial recognition system: Eigenfaces, Fisherfaces, Local Binary Patterns (LBP) . Eignefaces is chosen due ability to dynamical registration of unknown faces; Fisherfaces due to inherent robustness towards illumination; LBP for robustness towards minimal training data and illumination. Moreover, some face standardization and preprocessing techniques have been discussed for improving recognition rate, which include face detection, frontalizing tilted faces. Additionally, for further recognition improvement a method is utilized for dealing with video frames, which is averaging a recognition result within specified amount of successive frames. Furthermore, a system is designed and implemented in a modular manner which allows to plug in and out various face recognition algorithms along with opportunity to choose details for algorithms such as distance metric and classifier

...

Acknowledgements

It would not have been possible for me to work in this project without the kind support and help of people that have supported me throughout the project. I am highly indebted to Melbourne Networked Society Institute for their guidance and constant supervision, specifically Ken Clarke, Zaher Joukhadar, and Chamil Jayasundara. Moreover, special thanks to my supervisor from Melbourne University, Computing and Information Systems (CIS) department, Dr Edmund Kazmierczak for his supervision. . . .

Contents

List of Figures

Chapter 1

Background

1.1 Introduction

Recognition of human faces is a trivial task for human beings. Experiment done by Turati et al in 2006 [3] shows that babies one to three years old can easily recognize faces of known people. However, it appears that this task is fairly complicated for computers. According to research conducted by Hubel and Weisel [10] human brains have specific nerve cells devoted only for recognizing distinctive features of faces like edges, lines, different angles of facial organs such as eyes, ears, nose etc. Part of the human brain intended for processing visual information, which is called visual cortex can easily combine all these information that comes from different sources into useful patterns. The motivation for face recognition actually comes from this fact. If it is somehow possible to extract distinctive features of human faces, the problem of face recognition can be reduced to classification problem, which is one of the main areas of Machine Learning. Thus, the problem of facial recognition can actually be thought as supervised machine learning problem with huge emphasis on the effective feature extraction and selection. A lot of researches have been done in the area of facial recognition since 1960s. In recent years this technology has achieved significant advancement. Therefore area of applications for this technology is getting wider. Facial recognition has found its applications in social networks, surveillance, access control, security and many more others [7]. For example, automatic tagging in Facebook, "Windows Hello" feature in Windows 10 operating system are cutting-edge examples of facial recognition technology. Despite its prominent applications there are further prospects in applying facial recognition. Main motivation behind this project was utilizing facial recognition technology to be able to increase the quality of location-based services. The prospective is that, if it is possible to identify humans' locations using facial recognition technology in areas where standard location services fail, providing services based on location can increase in quality. For instance, if we take university of Melbourne campus as an example, identifying students' and staff members' locations using footages taken by IP cameras all around can be very beneficial, in terms of the quality of service based on location that university provides to its students and staff members. Another motivation is about security. If facial recognition enabled IP cameras are installed in crowded places, it may be possible to track criminals by police as they move from camera to camera. For this project main research question is identifying feasibility of applying facial recognition technology to robustly determine locations of people. Primary direction taken was investigating various facial recognition strategies to find suitable methods that can work in difficult situations. There were various challenges faced during the research process. One of the main challenges was finding a recognition method that can work with minimal training data. Admittedly, in real world it is very hard to collect various images of the recognition subject with different configurations and most classification methodologies in supervised machine learning rely heavily on the training data. Therefore, it is very important to find a way of training a recognition model with minimal training data - ideally only one face image per subject[1]. Another challenge was working with illuminated face footages. It is obvious that IP cameras can take photos of the same person at different times of the day. Thus, it is utterly significant that the recognition method is robust to any kind of illumination change in faces. Some widely known facial recognition methods treat face image as a vector in high dimensional image space. Changes in these vectors due to illumination are non-linear and massive [5]. Therefore identifying patterns that determines uniqueness of a face in these vectors becomes harder. Additionally, there were problems caused due to variance in poses of faces. If the face is not perfectly frontal it becomes even harder to notice distinctive features of faces. In real world situations it is very hard to achieve collecting perfect frontal images from IP cameras. Moreover, recognition subjects can be in different distances to the camera, which makes detection and recognition more difficult. Camera types which causes the quality of an image to change can also have an effect on the recognition rate. Another main problem is difficulty of recognizing faces in sequence of video frames. As not all video frames will have same quality due to blurring and obscurity, it is hard to make an algorithm that works with images, perform reasonably well with video frames as well.

The initial step in face recognition process is face detection. There are variety of face detection methodologies. In this project as a face detection mechanism Haar feature based cascade classification for object detection is used. This mechanism is proposed by Paul Viola and further enhanced by Rainer Reinhard [28] [14] [17]. This is a general-purpose object detection mechanism which can work for human faces as well. The face detector version of this method has been pre-trained on positive and negative examples, so it can distinguish faces from other objects. Although this method has high accuracy of detecting nearly-frontal faces, there is still a need for detecting of a human faces in difficult situations, such as face occlusion, pose variance etc. However, this aspect is not investigated in this project. For implementing and designing purposes various tools have been chosen due to different factors. Mainly the primary components of project have been developed with python programming language. Python provides very nice data structures and libraries for scientific and commercial development. Furthermore, python has very nice wrapper to OpenCV (Open COmputer Vision) library, which is originally developed in C++. OpenCV is an open source computer vision library started by Intel in 1999 that contains significant amount of image processing methodologies. Additionally, *face-rec* library [9] by Philip Wagner, which is originally written in python, is used to experiment various facial recognition methodologies. This library allows a user to plug in and out different components of facial recognition algorithms. For example, a user can try different classification methods, distance measures for one facial recognition algorithm.

The rest of the paper's outline is designed as follows. Initially the literature view in facial recognition is introduced to provide the context for the research. Next, face detection and pre-processing of images for face recognition purpose are discussed. Moreover, different methodologies of face recognition are discussed and justified in terms of suitability for using location detection with IP cameras. For evaluation purposes the process of data collection is explained along with experiments conducted on collected

[1]Subjects are referred to humans that are target to identification process

and standard face databases. Finally, the system design is discussed in a separate section which is succeeded by conclusion and future directions.

1.2 Literature review

Face recognition as a computational task contains various challenges to be dealt with. These challenges include but not limited to: changes in facial expressions, illumination, aging in recognition subjects; change in the poses of faces; difference in sizes of face images, presence or absence of glasses, beard, mustache etc; occlusion [19]. However, most state of the art facial recognition algorithms cannot directly address these challenges. There have been various approaches to face recognition since the start of the this area since 1960s. The very initial and intuitive method was approaching face as a collection of distinct geometric features. One of the very early researches in this areas was done by Kanade [12]. The approach that he took was extracting feature vector that represents geographical information about face such as distance between eyes, nose and mouth etc. Successful application of this method can naturally yield robustness towards illumination. However, it appears that even identifying crucial spots in face accurately such as location of eyes, mouth, nose is very hard problem. Moreover, the work done by Brunelli et al [6] shows that even in case of finding accurate locations of facial organs does not yield accurate recognition performance as these features might not distictively identify faces.
Another mainstream methods of facial recognition, treat images as a vector in high dimensional image space, so called holistic approach, and try to extract distinctive features for the purpose of recognition [26]. Since extraction of useful features causes to end up with fewer number of dimensions, the process of classification becomes faster with these kinds of methods. However, the effects of external resources, such as light, alter the image vectors drastically. Therefore these methods are not robust to illumination changes by nature. However, some pre-processing of images, such as normalization of light or incorporating subject label into image vector can yield desired results.
More recent studies in this area focus on the dividing face into multiple local parts and extracting local features out of those parts. Researches conducted by Wiskot and Messer [2] [16] [29] show that these types of methods are more robust against occlusion, light and small training data. However, the effective way of dividing face spatially is still an open research question [18].

There has been fair amount of research in dealing with problems caused by the change in pose of the faces. One of the most promising methods which is conducted by Roy et al [23] addresses to this challenge via relying on the idea of constructing a perfect frontal faces. This method is utilizing available side-view photos of subject to construct a frontal face by so-called face mosaicking. Although this method is fully automatic there is a slight drawback, which is a requirement of side-view images that might not be available in real world systems.

For tackling an illumination problem specifically the work done by Tang et al [24] has yielded astonishing results. This research has made a contribution of processing an illuminated image to eliminate most changes caused by light while preserving essential appearance details necessary for recognition. According to experiments that they have conducted the normalization method helps a face recognition algorithm to have

0.1% of false acceptance rate. Using this algorithm as a preprocessing step, standard algorithms may also be made robust to illumination changes.

Chapter 2

Facial recognition methodologies

2.1 Detecting and Standardizing faces

Facial detection is an initial step in all Face recognition algorithms. However, it appears that face detection is much easier problem for frontal images. One of the most widely used techniques for face detection is face version of object detection mechanism proposed by Paul Viola and Rainer Leinhart [17]. The mechanism is called "Haar-feature based cascade classifier for object detection". This method is based on Adaboost algorithm, which allows performing efficient classification by dividing classification process into multiple phases, each next phase being advancement on top of previous stages. The word cascade actually comes from this notion. The algorithm has high accuracy (close to 100 %) with a false positive rate 1 in 14084 [28]. Although this algorithm has an exceptional detection rate, the assumption is that the input images are near frontal images. However, for using recognition system along with IP cameras requires a detection of non-frontal images as well. Therefore, there is still a need for a much more advanced detector for situations like this, which can be thought as a separate research direction.
After the face detection process, the next step is standardization of detected faces. Most face recognition algorithms including some of the algorithms that we use for this project, assumes that instances in a face database are of the same size. There are various factors for this assumption. First of all, this assumption makes mathematical calculations easier. For example, in holistic methods all images are treated as a point in high dimensional space. If image sizes vary, their corresponding vectors will not lie in the same space. Therefore, it is appropriate to convert all images to the same size. Moreover, the images with different size will surely contain a different number of pixels. This difference will cause problems when we divide face into different spatial parts and generate histogram vectors out of those parts. We have used linear interpolation technique for standardizing images of different sizes. This technique works quite fast and the quality loss is not severe [25]. Moreover, to increase the recognition rate we have also incorporated eye detection mechanism. The postulation is that if it is possible to detect eyes in face image, it means that the face image has good quality for facial recognition. Thus, the recognition can be performed more accurately. However, it is also obvious that this mechanism can increase missing actual faces with low quality or in case of obscurity of eyes in a footage. Additionally, the eye detection mechanism is also necessary for dealing with "tilted face problem" as it is possible to identify the tilt degree of face using locations of eyes in a face. The mechanism to deal with that issue is described in later parts of the paper 3.4

2.2 Facial recognition methodologies

In this section we will first explain the face recognition methodologies that we consider for using along with IP cameras and then compare them for suitability. Main work regarding facial recognition suitable to work with IP cameras that have been done was finding a method that can work with minimum amount of training examples and robustness towards illumination. For that purpose we have tried three different algorithms: Eigenfaces, Fisherfaces, Local Binary Patterns. First two algorithms are holistic approach towards face recognition. They treat images as a vector of gray scale values and perform manipulations on them in a suitable high dimensional plane. The third method approaches an image in totally different way. With this algorithm an image is treated as spatially divided surface and tries to compare these parts in different images to match the right subject.

2.2.1 Eigenfaces

This method takes an information theory insight into the content of face images [27]. This algorithm supports learning new faces in an unsupervised manner and latter recognize those faces as well. Therefore, EigenFaces was a good candidate for the situation. The approach can emphasize the local and global distinctive features of face for recognition purpose. However, the identified features with that method may or may not directly take intuitive features of faces, such as eyes, ears, nose, mouth etc. into account. Generally the method treats each image as a point in high dimensional feature space. Then each image in a database, are projected to a subspace that has very small dimension size in comparison with actual vector size corresponding to face image. The projection tries to keep the variation maximum among image vectors. In other words, on the face database Principal Component Analysis (PCA) is performed. As a result of PCA only important features are extracted (hopefully) and classification process is performed in only lower dimensional vector space. The procedure of recognition is described briefly as follows [27]:

1. As a first step all face matrices are converted into a gray scale image, which is basically a matrix that is composed of numbers from 1 to 255. However, to be able to treat face images as a point in a plane we have to somehow convert a matrix into a vector without loosing the pattern. One most famous way of achieving this is just simply folding a matrix into a vector (i.e. if we have an image of size $N * N$ we can fold this matrix into a vector of size $1 * N^2$) [18]. Now we have a vector of very high dimension. In our scenario we are using images of size $200 * 200$ as this size is big enough to hold distinctive face features and small enough to keep a good performance for training and query.Moreover, this size means already about 320 Kb (without compression) data to be sent over communication channel between server and IP cameras which is big. Furthermore, with this size of images, we have been working in a 40000 dimensional vector space, that is big for real time mathematical operation as well. Therefore, increasing the size would deteriorate the system performance.

2. After converting all images in the database, we have p number of vectors in an image space where p is number of samples in a face database. Let

$$X =< x_1, x_2 \ldots x_p >$$

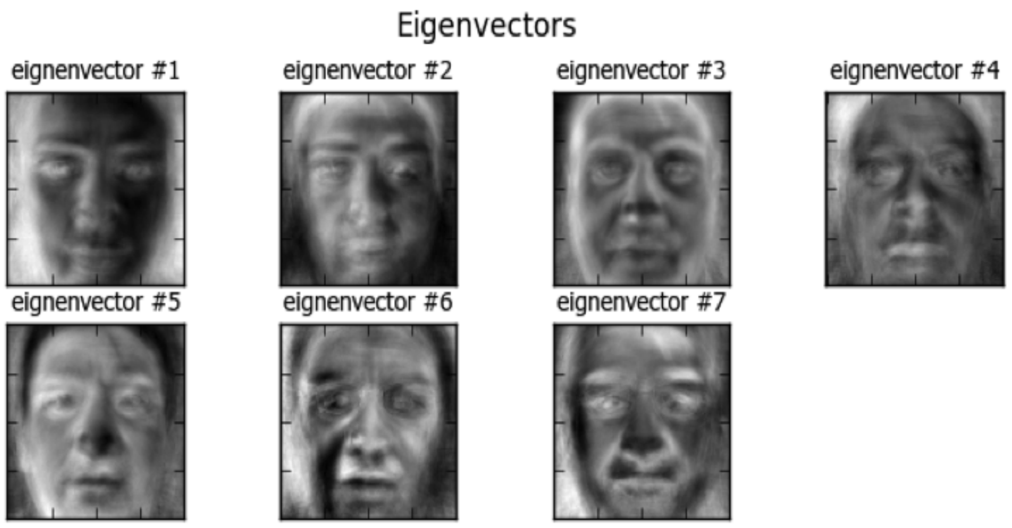

FIGURE 2.1: Egien vectors generated by PCA.

be our image vectors generated as described in the previous stage. After that, mean image vector is calculated across all samples in a database:

$$\mu = 1/p * \sum_{i=1}^{p} x_i \tag{2.1}$$

3. Computing covariance matrix:

$$S = 1/n * \sum_{i=1}^{p} (x_i - \mu)(x_i - \mu)^T \tag{2.2}$$

4. Computing eigenvectors and eigenvalues: This part of the algorithm finds most discriminating face features by calculating eigenvectors and respective eigenvalues.

$$S * v_i = \lambda_i * v_i \tag{2.3}$$

From the last equation specified k number of eigenvectors with largest eigenvalues are taken as primary components. $k = 300$ produces reasonable accuracy and speed[18]. Since eigenvectors are vectors of the high dimensional image space, we can show them as an image by just folding back to the original image size. With that way we can see visually how Eigenfaces encode facial texture for recognizing faces. First 7 eigenvectors generated from the data set that we have collected are shown in Figure 2.1.

From the figure we can see that eigenvectors successfully encodes the facial texture. However, the first - most influential eigenvector encodes the light information as well (the light from the left is clearly visible in the eigenvector), which can have negative impact if the light direction is different in testing data from what we have in training data.

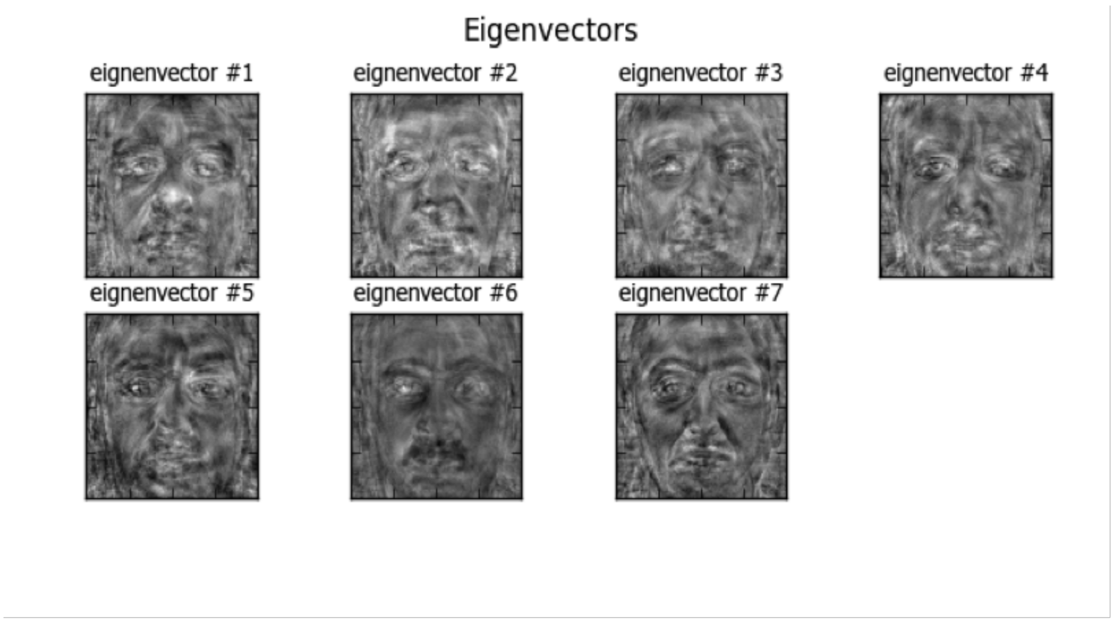

FIGURE 2.2: Eigen vectors generated by LDA.

5. After step 4 recognition process is as simple as projecting all images (including test instance and training set) into subspace of chosen eigenvectors and determining the closest neighbor to the test instance among training instances and pick the label of the closest match as a recognized subject.

Although training phase of this algorithm is not as fast as desired, the query time is exceptionally good which can make this algorithm a good candidate.
The concept of subspace and ignoring subject labels in projection stage allows Eigenfaces algorithm to treat new faces in an unsupervised manner. This method is described briefly as follows: Initially all known faces are projected into a subspace, which creates cluster of points according to subject label. When unidentified face is projected into a subspace, if it is not close to any of the clusters it is saved for future use. If repeating this process, creates a new cluster with in a subspace, emergence of a new face(previously unidentified face) is postulated and is saved as a new subject[27]. However, this methodology requires to run clustering algorithm over image space to define new clusters.

2.2.2 FisherFaces algorithm

Eigenfaces algorithm does a good job by projecting images into smaller space to decrease training time. However, the projection does not consider labels of the training images, which is available in training phase. That may cause carrying features that are dependent on external effects like light into a subspace, which can potentially deteriorate recognition. FisherFaces algorithm deals with that issue by considering subject labels in training phase. More specifically it uses class specific Linear Discriminant Analysis to reduce dimensionality. Fisherfaces algorithm determines within-class and between-class scatter factors and maximize variation of their combination which yields a projection where all points within class are close to each other, whereas points of different classes are as far as possible. Figure 2.2 shows the 7 eigenvectors calculated by Linear Discriminant Analysis over vectors of same database.

In Figure 2.2 we can see that, the light information is eliminated from the basis set. Fisherfaces algorithm approaches a problem as same as Eigenfaces. However, since it emphasizes between class distribution, it can yield better result in terms of illumination robustness. Although it can eliminate the light influence, there can be a problem if we do not have enough training samples per subject. If in the training phase the algorithm has seen only fixed-illumination level of face, it is likely that the algorithm will not be able to recognize huge illumination variance in testing face. This is due to the fact that, the illumination can cause a face image to be projected to a wrong cluster as changes in illumination have higher impact than changes in actual faces in terms of the actual number values of vectors [1]. Therefore, it is assumed that the training set is well illuminated[18].
Moreover, there is another disadvantage of this method in terms of projection to subspace. If as a result of projection, between class scatter is large, it is also highly likely that within class scatter is also high. This fact makes the classifier's problem harder, as it becomes hard to determine a true class for a projected query image in a subspace [11].

2.2.3 Local Binary Patterns (LBP)

This algorithm is different than previous ones in terms of approaching a problem. In this case, the image is not treated as a high dimensional vector. Instead, distinctive features are extracted from the spatially different parts of the face image. The general flow of the method is as follows:

1. For a size of $k * k$ image matrix, each pixel is treated as a center and is assigned a special number calculated as follows: The gray scale values of neighborhood cells are compared to gray scale value of center cell. If the cell value is greater than the center cell's gray scale value it is assigned to 1 otherwise 0. With the clockwise direction starting from the top left corner a binary pattern is constructed. A decimal value corresponding to the binary pattern is assigned to the center cell. Figure 2.3 is an illustration of this operation with $k = 3$.

2. The image is divided into n different spatial parts and from each part the histogram vector is extracted and concatenated together to form a vector that distinctively represents the original image.

3. The histogram vector of a test image is compared to the training set's vectors and nearest neighbor in terms of chosen distance measure is chosen as a prediction.

This algorithm conveys a property of robustness towards illumination via applying LBP operator. Also, since the algorithm does not require significant amount of training set it can be a good candidate for our situation.
Despite LBP algorithm's advantages there are various drawbacks as well. One of the main disadvantage is that, if an algorithm produces long histograms recognition will be sensitive to rotation [13]. Moreover, the algorithm has a sensitivity towards noise, as it generates histograms from different spatial regions of face, in presence of noise the histogram vectors might not represent actual discriminating face feature [13]. Therefore, the algorithm is sensitive towards low quality images.

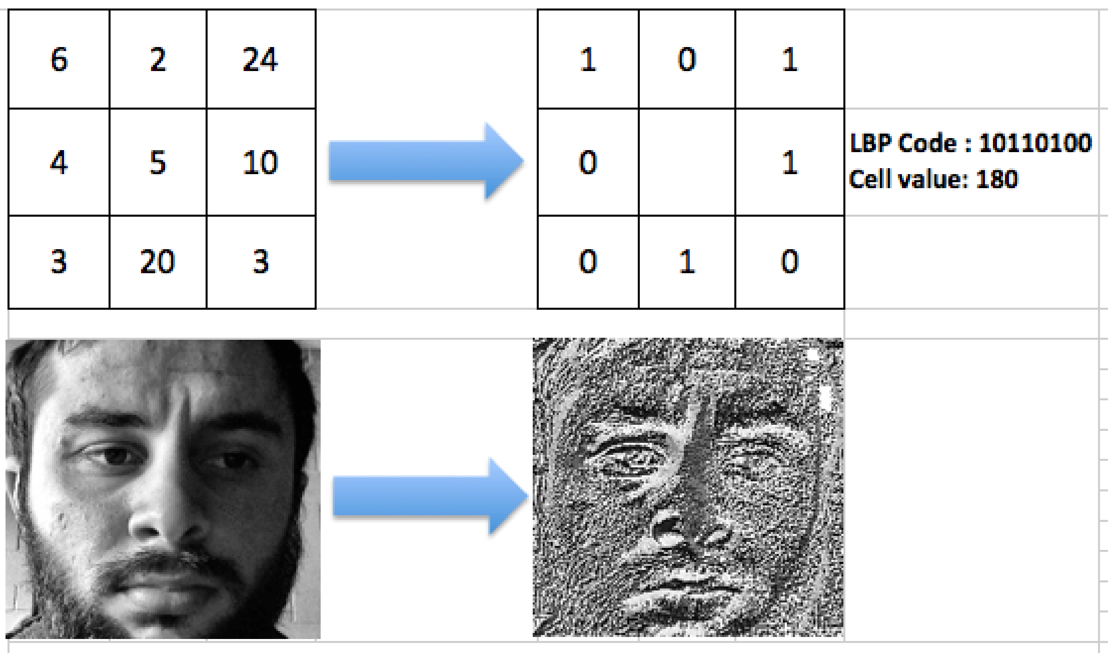

FIGURE 2.3: LBP operator in action.

Chapter 3

Experiments and results

3.1 Data Collection

For the purpose of evaluating algorithms we have collected a database of face images. There were less than 10 subjects in our collected database. However, average number of samples per subject was over 20. This allows us to either test an algorithm with huge number of training set, or train only with few images but test on whole data set. Our collected database contains images with variety of illumination level. However, since our first attempt was to investigate on only near-frontal images and then perform some processing on posed images to create a frontal image, all of the samples are near-frontal. Other than our collected database, we have also tested the algorithms on AT&T [4] database, which has fewer light variances among faces, but with a lot of different facial expressions and slight rotation in poses. These two databases allow evaluating algorithms in terms of robustness towards illumination, facial expressions and very slight change in pose(pose variance is not huge as it still maintains original face). In terms of evaluation of the algorithm it would be better to collect a database of face images of known subjects that represents actual video sequence. That way it can be possible to evaluate how face recognition algorithms perform with actual sequence of videos.

3.2 Experiments

In this section, results of experiments have been discussed. Algorithms have been run with two different face databases with different configurations for the purpose of validating expectations. Algorithms are compared in terms of speed in training time and query time. Training time is important for dynamical registration, if recognition system is required to identify unrecognized faces in an unsupervised manner and later on recognize them as well. Therefore training time will be significant, as we will need to train the model regularly while the system is running. The importance of the query time is due to general performance of the system. Figure 3.1 shows the performance of three algorithms in terms of training time.

Figure 3.1 indicates that there is an approximate quadratic correlation between training time of Fisherfaces,EigenFaces and database size. This is due to time-consuming matrix operations, (such as calculating eignevector and eignevalues) that are performed for feature extraction. Both FisherFaces and EigneFaces algorithms calculate co-variance matrix and set of eigen vectors of the image database, which consumes a lot of computational power. However, LBP algorithm is much more efficient in terms of training time as this algorithm only applies LBP operator to an image(as described above) and generates histogram vector, which is simply creating statistical data of pixel distribution. The importance of training time is due to being able to train the model while the

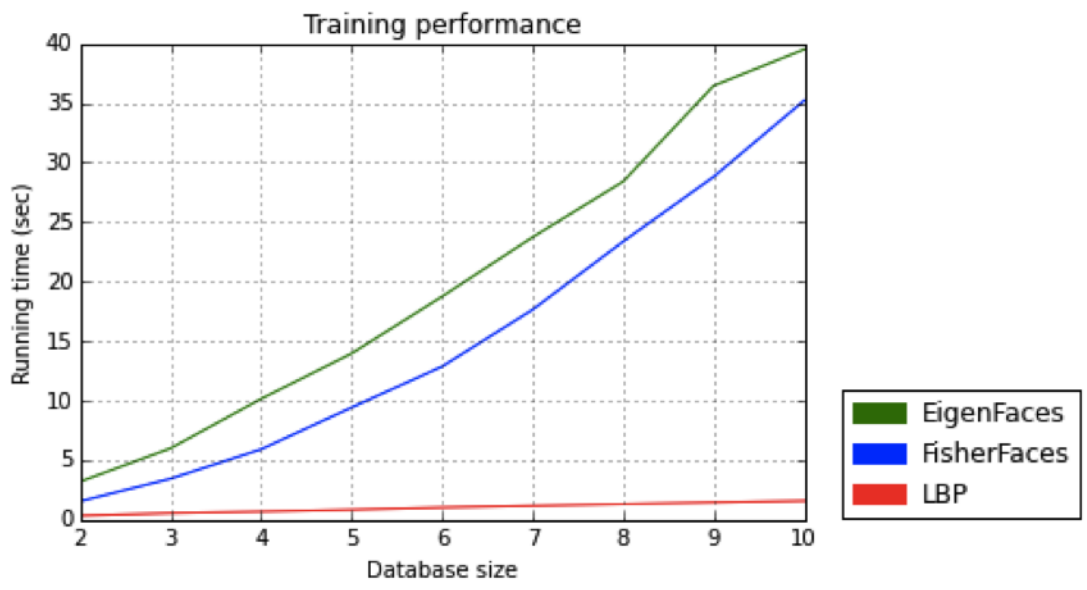

FIGURE 3.1: Training performance.

system is running. If the training time is too long as system identifies new faces, it will be unfeasible for retraining the model on newly identified faces. In that sense, LBP algorithm is significantly superior to both Fisherfaces and Eigenfaces.
Query time is also important. Since the system is intended to dynamically perform recognition on sequence of video frames, it is very important to have an optimal query time, especially when the video frame contains huge number of faces. Figure 3.3 compares query time of three mentioned algorithms.

The Eigenfaces and FisherFaces algorithms' query phase is almost identical. A query image is projected into subspace created on the basis of k most important principal components. Later, the projected vector is compared to the respective vectors in terms of distance of training images and decision is made via chosen classification algorithm. The procedure is shown below (Figure 3.3). The projection contains an operation of multiplying an original image with matrix of principal components

$$y = P^T * (q - \mu) \tag{3.1}$$

where, $P = < p_1, p_2 ... p_k >$ matrix of principal component vectors. q is folded version of query image matrix, μ is calculated mean vector of image database.

This operation expectedly is not computationally expensive as all the component of the equation above is pre-calculated during training phase.

Query process with LBP algorithm contains more processing steps as described in Figure 3.2. Initially LBP operator is applied to an original image; Later on, the histogram vector is generated for comparing with other histogram vectors generated during training phase. Apparently, applying LBP operator is the most time consuming part, as each single element of the matrix is compared with its specified neighborhood for generation of LBP codes[1].

[1]Binary sequence of the corresponding cell

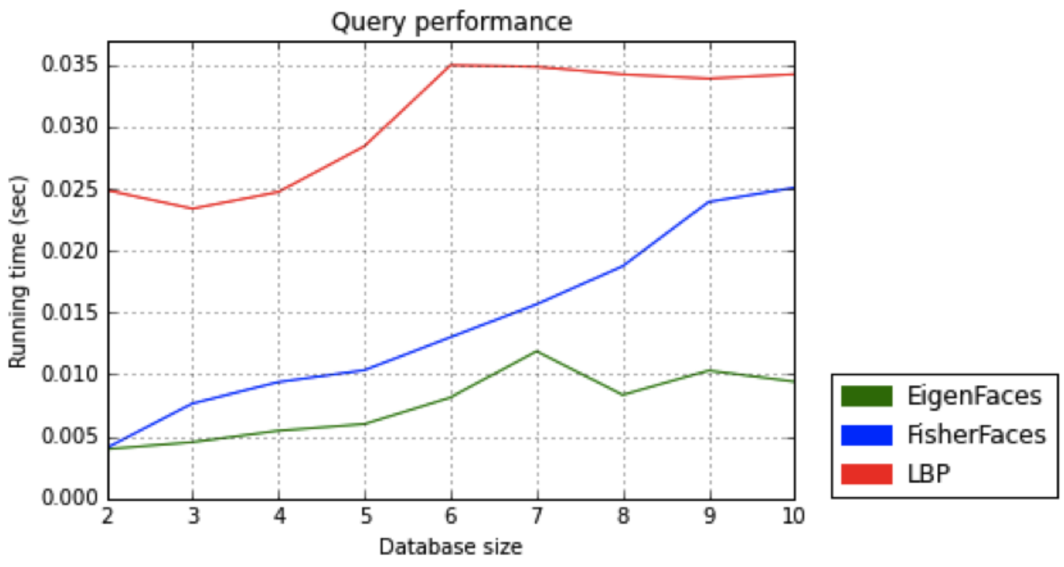

FIGURE 3.2: Query performance.

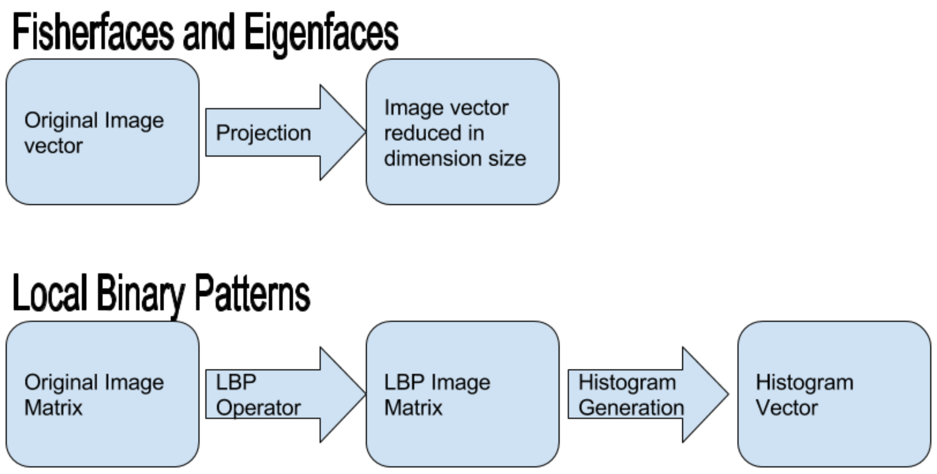

FIGURE 3.3: Query process with Fisherfaces, Eigenfaces and LBP algorithms.

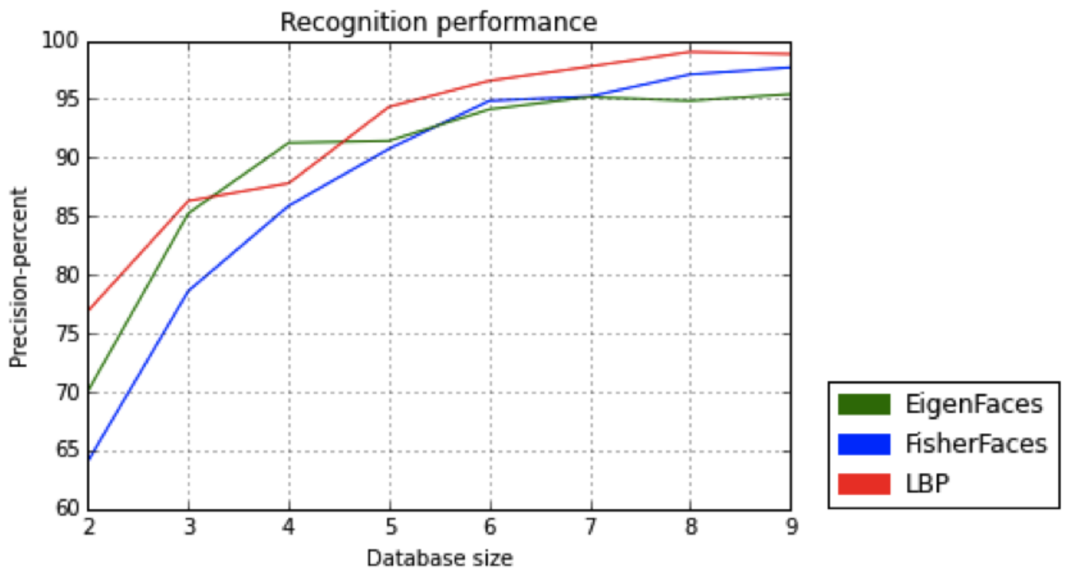

FIGURE 3.4: Precision of algorithms over fixed illumination database

Apart from query and training performance, it is also significant to compare recognition performance of algorithms. Precision is chosen to be decision factor of the recognition performance, because precision indicates how algorithm can pick the true positive. In other words precision is an ability not to attach a label to an instance other than its original label. Precision is defined as follows:

$$p = TP/(TP + FN) \tag{3.2}$$

Where, TP is number of correctly recognized faces, FN is number of incorrectly recognized faces.

There is no need to consider other measures, such as recall, accuracy, as we only care about correct recognition rate.

Figure 3.4 compares algorithms in terms of recognition performance as database size increases. This experiment is conducted with fixed illumination database.

All algorithms perform well on this database. Since there is no external influence on faces such as light, Eignefaces can project faces of the same subject close to each other in the subspace, as the variance of the instance will be dominantly based on the change of actual face texture. Therefore, with perfect light conditions Eigenfaces can be very good candidate as its speed in query time is superior to other algorithms. However in the presence of light, recognition behavior of algorithms vary drastically as expected. Figure 3.5 indicates this behavior.

Eignefaces's previous performance dropped significantly. That is because the change in faces due to light is more influential than changes in actual faces. The work done by Moses et al. [5] confirms this fact, as they state: "the variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity". The behavior of Fisherfaces is better in comparison with EigenFaces. However, in the experiment we have chosen the training and testing instances randomly. That means that there is possibility that, illumination level in testing and training set may vary. Fisherfaces algorithm projects instance to subspace considering the class (subject label) as well, it implicitly ignores the variance in light. Therefore, if either testing or training set is not well illuminated, the drop in

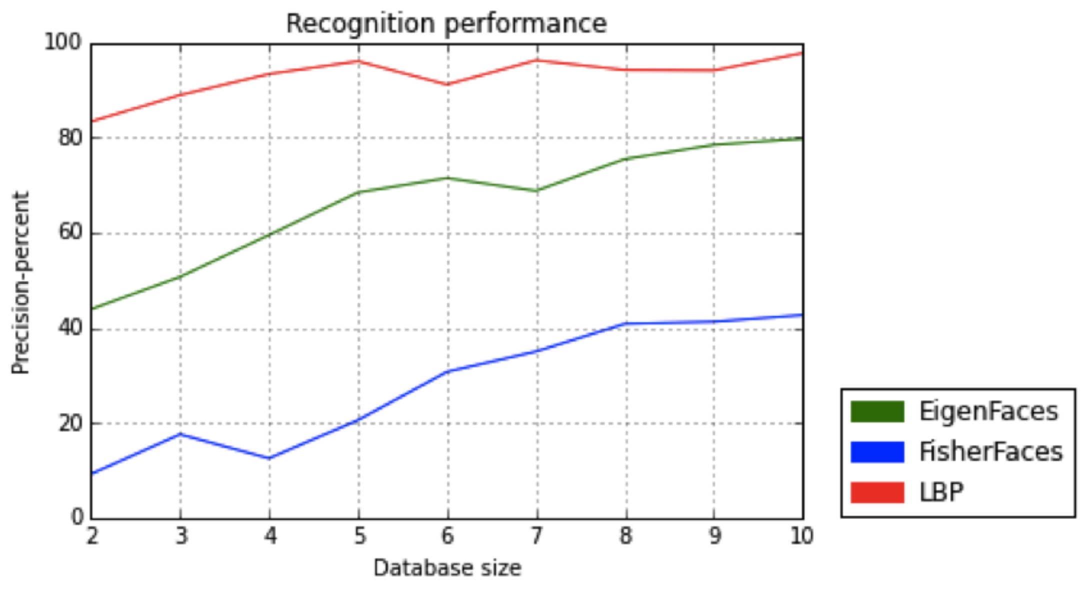

FIGURE 3.5: Precision of algorithms over illuminated database

performance is expected. However, LBP algorithm holds almost stable performance. The training with just few images (1 -2 images per subject) results in reasonable precision. Therefore, in terms of precision, LBP algorithm is more consistent than others.

Using ability of identifying an unrecognized faces and registering them as new faces of Eigenfaces method as described in section 2.2.1, along with LBP algorithm can allow to build a system that can dynamically identify new faces. Initially the query image can be passed to LBP algorithm to perform an accurate recognition. If LBP algorithm is not able to recognize a face, a query image can be forwarded to Eigenfaces algorithm to register an unrecognized face. After forming a cluster, in case of emergence of new subject, that subject can be added to the training set and training of these 2 algorithm can be repeated. With this combination it may be possible to achieve accurate dynamic registration of new faces. However, this idea is not implemented in this project and might be a good direction for future.

3.3 Choice of distance metric

Each of these algorithms uses classifier for training and prediction. In our case we have used a simple k- Nearest Neighbor classifier with k = 1. Basically, the classifier chooses the closest vector to a query vector among all training instances. The closeness factor is determined in terms of distance measure. As EigenFaces and FisherFaces treat images as high dimensional vectors, standard metrics for high dimensional spaces, such as Euclidean, Cosine distance, can be chosen. Since the essential idea behind holistic approach to facial recognition is all face vectors are projected to a subspace and it is expected that the vectors are clustered together respective to their subjects, actual distance between two points or angle between 2 respective vectors is very small if they are of the same subject. For clarity purpose, these two metrics is introduced in equation 3.3 and 3.4

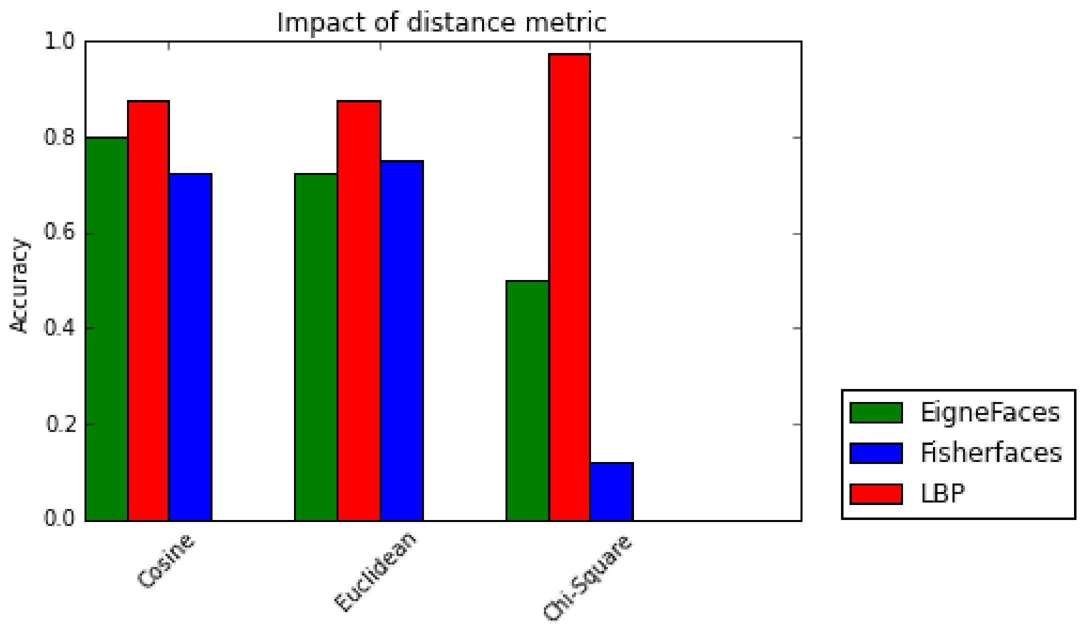

FIGURE 3.6: Impact of distance measure

$$e(v_1, v_2) = \sum_{i=1}^{N} (v_{1i} - v_{2i})^2 \tag{3.3}$$

$$c(v_1, v_2) = \frac{\sum_{i=1}^{N} (v_{1i} * v_{2i})}{|v_1| * |v_2|} \tag{3.4}$$

However, using this type of metric is not useful for LBP algorithm. The reason is that, LBP algorithm uses histogram vectors and histogram vectors should not be treated as a point in a plane, rather it is a statistical distribution of pixels of a face image. For comparing histograms chi-square test is widely used technique and go hand in hand with LBP algorithm [22]. Chi-square distance is defined as in equation 3.5

$$cs(H_1, H_2) = \sum_{i=1}^{N} \frac{(H_{1i} - H_{2i})^2}{H_{1i}} \tag{3.5}$$

From Figure 3.6, the effect described above can be seen. Chi-Square distance impacts LBP algorithms to perform high recognition performance. However, with Chi-square distance Eigenfaces and Fisherfaces perform very poorly. With Cosine and Euclidean distance, algorithms perform moderate.

Conclusively, it can be said that, in case of LBP algorithm it is better to use histogram comparison methods, such as Chi-Square similarity measure, rather than actual distance based metrics.

3.4 Dealing with problems produced by video presence

Face recognition algorithm that performs well with image data set, might not work well with video frames. One important reason is that in sequence of frames, some of them might be very bad in quality due to blurring(as movement happens some frames might be blurred). Moreover, apart from pose variance, the tilting of the face can increase false

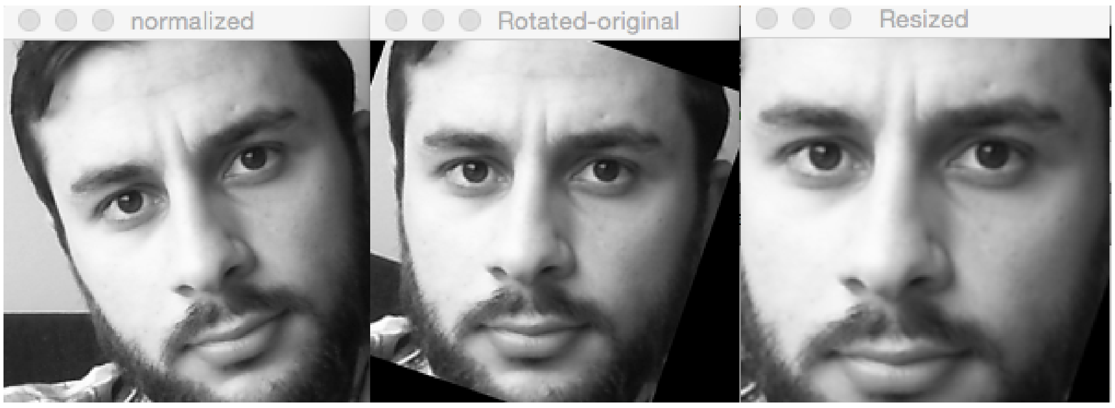

FIGURE 3.7: Rotation of tilted face and eliminating background

positive rate. For example, tilting a face can cause an LBP algorithm to choose different parts of two faces for comparison
For dealing with low quality frames, very simple approach has been applied. The approach is very simple and improves video recognition slightly. The process is as follows:

1. Each frame is extracted and is put into face detector to determine possible faces.
2. After detection process subjects in each frame is saved for future use.
3. The process is repeated specified k number of times. After k frames. The most common recognized subject is taken as a classified subject. For confidence, average of all k confidences is taken

However, choosing k might be camera specific as some cameras take more images per second than others.

For solving a "tilted face problem"following sequence of followed.

1. Detecting left and right eyes of the face using eye version of Haar object detector algorithm.
2. Calculate a tilt degree, by calculating an angle between horizontal line and a line that passes from location of both eyes.
3. Rotate a face image by the degree identified in previous stage
4. Take maximum possible rectangle that omits the black background caused during rotation.
5. Resize a final result to a size that is valid for recognition algorithm.

One example of the flow above is shown in Figure 3.7

Chapter 4

System design and implementation

4.1 Choice of tools

The system is implemented in python. There are variety reasons for this choice. First of all, python has good libraries intended for scientific development such as numpy, scipy, matplot etc. [20]. These libraries allow conducting scientific experiments without too much effort. Moreover, python is also one of the famous programming languages for software development as it can be used as object oriented, functional or scripting language. Thus, developing the system in python allows both scientific and commercial development, which is a desirable combination for our situation. For processing images, very famous computer vision library – OpenCV(Open Computer Vision) is utilized. The library is originally developed by Intel using C and C++. However, now it has python binding which is officially supported by OpenCV documentation. Therefore, majority of function that can be referenced from C or C++ implementation of OpenCV is available for python version as well with a cost of wrapping. We have used OpenCV primarily for processing images such as histogram equalizing of an image for emphasizing important pixels, resizing images with different interpolation methods, rotating tilted faces etc. OpenCV also supports a face detection mechanism. This functionality is also used with pre-trained detection model which is called Haar Cascaded face detection [17]. Although OpenCV has very rich functionality in terms of image processing, the flexibility of face recognition is slightly limited. Face recognition methodologies in OpenCV do not allow to tweak algorithms, such as choosing distance metric, classification algorithm, automatic pre-processing, unless there is a change to actual C++ implementation. Therefore, for face recognition purpose *facerec* [9] library is used. This library is developed purely in python and is flexible to make changes to operations of algorithm.

4.2 System design

The purpose of the system design was to come up with an architecture where most components are flexible for change. That way the system is not dependent on any detection or recognition algorithm. Overall architecture is a single server multiple clients. All clients are IP cameras connected to the single server. All processing is done in a single server. Following figure (Figure 4.1) is a general architecture of the system.

For each single IP camera, the thread is created in the server upon receiving a request. Each thread uses shared training data to train the model. After processing, results are saved to a central database. The result contains subject identity, camera identity, time stamp and confidence of recognition. With that way subject can be tracked based on time (time stamp field) and location (assuming each camera is attached a

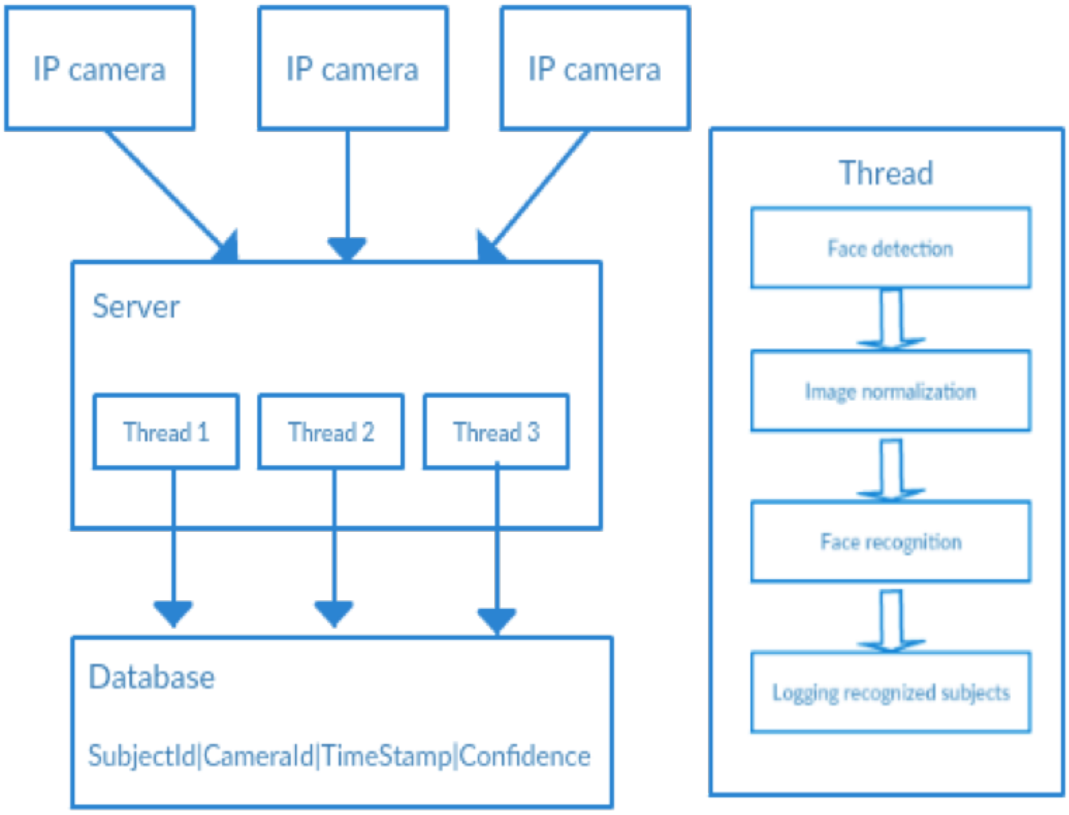

FIGURE 4.1: System architecture

static location). With that architecture, separate independent solution can be developed for user interface with just talking directly to the database. In each thread five general steps are followed (Figure 4.1). In the first step thread is initialized by training the recognition algorithm on the local image database. Next step is face detection, which is simply detecting faces in a video frame. For simplicity and performance only 1 frame in 100 is processed. However, it would be better, if frame with best quality can be chosen among 100 frames. The method explained in section 3.4 can be very simple solution to that problem. However, this method reduces dynamicity of the system as it reduces concurrent recognition rate in a footage. Next step is normalization of detected faces in terms of size. That step is important for facial recognition algorithm. All recognition algorithms assume that the training set and testing set are of the same size as mathematical operations, such as calculating principal components, comparing image vectors, are performed on only one high dimensional space. Therefore, all detected faces are resized to a specific size using linear interpolation method. Linear interpolation is relatively fast and can maintain important pixels of the face image reasonably well. Forth step is actual recognition step. In this step, pre-trained facial recognition algorithm predicts a subject label of the detected face. If the prediction confidence is above the threshold (as threshold is a specific distance point) the face is treated as an unrecognized subject. Defining the threshold is environment specific. Thus, threshold with one light condition might not be useful for other light conditions for all algorithms. Therefore, defining an adaptive threshold value is still an open discussion. The last step is saving relevant information to the database for tracking purposes.

4.3 UML diagram

Simplified version of the class diagram of the system is in below figure (Figure 4.2).

Description of this model is as follows: Recognizer is an abstract class that contains general structure for a typical face recognizer. FisherFaces, EigenFaces and SpatialHistogram classes are concrete recognition classes of EigenFaces, FisherFaces and LBP algorithms respectively. Detector is an abstract class that contains general structure for a typical face detector CascadedDetector is concrete class that uses Haar's cascaded detection mechanism. FaceTrack is a thread, which is intended to run recognition process. Main is a primary class of the system, which is responsible to create FaceTrack instances (threads) for each camera. Database class is a mean for connecting to database and is created to separate the business logic from database operations. With this class architecture the system is expandable. Hence, new recognition or detection algorithms can easily be attached to the system by following the general structure. However, the system does not support dynamic registration of new cameras. All cameras must be known upon starting the system. Moreover, since the number of servers is only one, scalability cannot be achieved as number of cameras increases. To achieve scalability the load balanced server architecture can be followed. With that architecture, complete transparency of resource management, scalable performance, and system availability can be provided. The work done by Xiadong Lui et al [15] propose a master- slave server architecture to provide characteristics articulated above. This type of parallel architecture is important, since the training of model require huge computing resource. In case of dynamical registration of faces, training on the fly is inevitable. The architecture proposed by Xiadong Lui et al. for face recognition system allows dynamically adding and removing slave servers from the system, which allows having a cost efficient system. Moreover, they also provide a scalable architecture for storing image

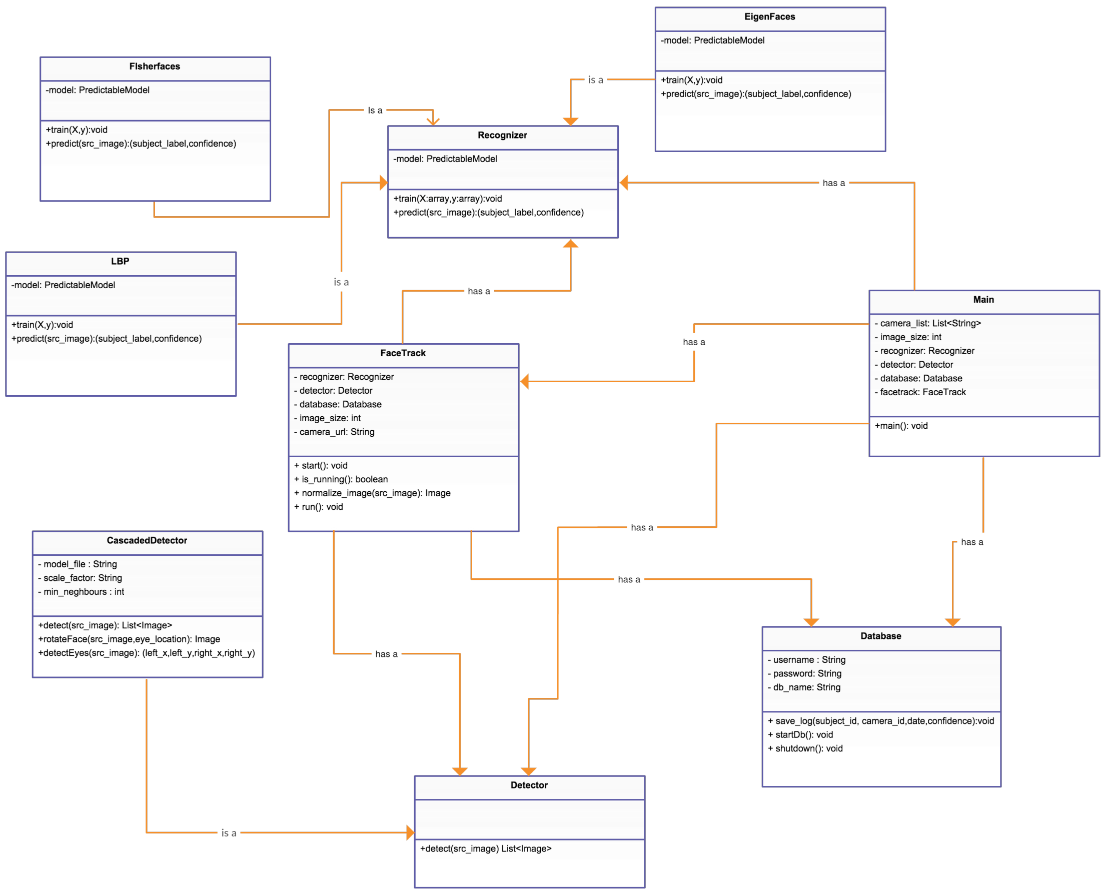

FIGURE 4.2: UML diagram

database, if the size of the database is huge enough not to be stored in a single machine.

Chapter 5

Conclusion and Future directions

5.1 Conclusion

Primarily We have investigated different face recognition algorithms suitable to work with illumination variance and perform well with minimum training data. Also, the performance of the algorithms in terms of speed for query and training also compared to test algorithms suitability to real time systems. Moreover, some simple methodologies have been discussed to overcome problems introduced by video sequences and tilted faces in video footage. According to conducted experiments, Eignefaces algorithm perform poorly when there is high illumination variance, but results with perfect light conditions are very high when enough amount of training data is provided. FisherFaces algorithm's robustness towards illumination is based on the availability of well-illuminated training data. However, LBP algorithm performs quite well with both varied illumination and fixed illumination level data sets. In terms of training performance LBP algorithm again outperforms Eigenfaces and Fisherfaces as latter algorithms perform computationally expensive mathematical operations for training the model. These two factors makes LBP algorithm to be a good candidate for real time usage. However, in terms of query time performance EigenFaces and Fisherfaces are superior to LBP. This creates a trade-off between query time and training time performance between these algorithms. Moreover, availability of using Eigenfaces algorithm for dynamical registration of unrecognized faces have promising future direction of this algorithm in this project. The sensitivity to noise in image data and long histogram vectors are another disadvantage of LBP algorithm that can be avoided with preprocessing to some extent. In terms of recognition performance with minimum training set, LBP algorithm significantly outperforms others. LBP algorithm can provide approximately 90 % of accuracy with just 2 to 3 training instances per subject, whereas Fisherfaces and Eigenfaces require at least 7 images for reasonable accuracy. This is another advantage of LBP algorithm. Conclusively, it can be said that LBP is a good candidate if there is a need for minimum training data with minimum time for training and huge variance in illumination level. However, in case of availability to allow training phase enough amount of time, but necessity of very good query performance Eignefaces (with perfect light conditions) or Fisherfaces(with well-illuminated training data for illumination robustness) can be a valid choice.

5.2 Future Directions

Currently the system has various issues regarding scalability and pose variance. There are two possible directions to be taken for solving scalability issue. First possibility is using the architecture proposed by Xiadong Lui et al, which is explained above in System design section. This architecture is suitable for large-scale systems. However, for systems covering small areas such as the area of university campus, this type of architecture may become expensive as it requires having various servers classified as master and slaves. Another possible solution for that problem is attaching small cpu to each camera which is responsible only for prediction of faces. As explained above predicting does not require huge computing power. Therefore, Raspberry pi [21] cpu, which costs around $50, should be enough for that purpose. Whenever there is a need for training, request can be sent to central server and training can be performed there and saved for further access of other IP cameras. This way, the central server can be free of processing prediction and dedicated to entirely training. For the second issue, reasonable solution can be generating a frontal face from non-frontal face footages. That way, it is possible for face recognition algorithms to perform at their best. There have been various works related to frontal face generation. The method suggested by Hiranmoy Roy et al [23] approaches a problem by determining the pose angle and using Face mosaicking to construct a frontal face. This approach can solve pose variance problem to some extent. As described in their paper, the method has problems when there is huge variance in face pose. Also, when eyebrow and hair overlap, detection of eyebrows becomes difficult. Hence, mosaicking a face becomes almost impossible. Method proposed by Zhenyao Zhu [30] tries to solve the pose variance by applying deep learning techniques. The work suggests that it is possible to generate distinctive features from faces with different pose variation without constructing a frontal face. Upon successful extraction of distinctive features there is no need to use frontal face recognition as any classifier can perform prediction on these set of features. However, the performance of such algorithm is questionable in terms of feasibility of using in real time settings as it relies on neural networks for prediction and training which obviously not as scalable as algorithms that are discussed above. Neural networks are well-known to be slow for training. This fact is supported in the paper by emphasizing the challenge of training with neural networks [30]. Regardless availability of these methods there is a previous step, which should be taken before dealing with pose variance. It is also difficult to detect faces when there is an occlusion in faces or huge pose variance. Therefore, initially it is better to take a direction to investigate different methods of face detection under difficult situations such as occlusion and pose variance. Moreover, if it is possible to access a set of frontal faces from a security database, the possibility of training and testing models based on those types of databases can be a future direction. Another possible future direction of this project can be relying on body shape and gait of a subject for identification rather than facial textures. There is an interesting research conducted by Robert T Collins et al. [8] in that area. Their method puts an emphasis on finding patterns of 2D silhouettes from a sequence of gait cycle for a specific subject. The method that they provide performs well with different viewing angles and background conditions. However, the drawback is that the method only supports single side view.

Moreover, for obtaining a good measure in terms of evaluation for real world situation, it would be better to collect a face database that represents actual video sequence labeled with subjects. In presence of this type of database one can perform a test on a video sequence and obtain more realistic evaluation measures for algorithms that are supposed to run on videos rather than static images.

Bibliography

[1] Yael Adini, Yael Moses, and Shimon Ullman. “Face recognition: The problem of compensating for changes in illumination direction”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19.7 (1997), pp. 721–732.

[2] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. “Face description with local binary patterns: Application to face recognition”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.12 (2006), pp. 2037–2041.

[3] AS Arnold, JS Wilson, and MG Boshier. “A simple extended-cavity diode laser”. In: *Review of Scientific Instruments* 69.3 (1998), pp. 1236–1239.

[4] *AT&T Face Database*. http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. Accessed: 2015-08-02.

[5] Peter N Belhumeur, João P Hespanha, and David J Kriegman. “Eigenfaces vs. fisherfaces: Recognition using class specific linear projection”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19.7 (1997), pp. 711–720.

[6] Roberto Brunelli and Tomaso Poggio. “Face recognition through geometrical features”. In: *Computer Vision—ECCV’92*. Springer. 1992, pp. 792–800.

[7] Rama Chellappa, Charles L Wilson, and Saad Sirohey. “Human and machine recognition of faces: A survey”. In: *Proceedings of the IEEE* 83.5 (1995), pp. 705–741.

[8] Robert T Collins, Ralph Gross, and Jianbo Shi. “Silhouette-based human identification from body shape and gait”. In: *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*. IEEE. 2002, pp. 366–371.

[9] *Face Recognition Library*. https://github.com/bytefish/facerec. Accessed: 2015-09-02.

[10] David H Hubel and Torsten N Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *The Journal of physiology* 160.1 (1962), p. 106.

[11] Sushma Jaiswal. “Comparison between face recognition algorithm-eigenfaces, fisherfaces and elastic bunch graph matching”. In: *Journal of Global Research in Computer Science* 2.7 (2011), pp. 187–193.

[12] Takeo Kanade. “Picture processing system by computer complex and recognition of human faces”. In: *Doctoral dissertation, Kyoto University* 3952 (1973), pp. 83–97.

[13] Liu Li, Paul W Fieguth, and Gangyao Kuang. “Generalized Local Binary Patterns for Texture Classification.” In: *BMVC*. 2011, pp. 1–11.

[14] Rainer Lienhart and Jochen Maydt. “An extended set of haar-like features for rapid object detection”. In: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 1. IEEE. 2002, pp. I–900.

[15] Xiaodong Liu and Guangda Su. “A Cluster-Based Parallel Face Recognition System.” In: *PDPTA*. 2006, pp. 737–743.

[16] Kieron Messer et al. "Performance characterisation of face recognition algorithms and their sensitivity to severe illumination changes". In: *Advances in Biometrics*. Springer, 2005, pp. 1–11.

[17] *OpenCV Documentation - Face Detection*. http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html. Accessed: 2015-08-02.

[18] *OpenCV Documentation - Face Recognition*. http:docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html. Accessed: 2015-08-02.

[19] Shailaja A Patil and PJ Deore. "Face Recognition: AS urvey". In: (2013).

[20] *Python Official Documentation*. http://www.python.org/doc/. Accessed: 2015-08-02.

[21] *Raspberry pi*. http://www.raspberrypi.org/. Accessed: 2015-09-02.

[22] Jianfeng Ren, Xudong Jiang, and Junsong Yuan. "A Chi-Squared-Transformed Subspace of LBP Histogram for Visual Recognition". In: *Image Processing, IEEE Transactions on* 24.6 (2015), pp. 1893–1904.

[23] Hiranmoy Roy et al. "Construction of Frontal Face from Side-view Images using Face Mosaicing". In: *International Journal of Recent Trends in Engineering* 2.2 (2009), pp. 55–59.

[24] Xiaoyang Tan and Bill Triggs. "Enhanced local texture feature sets for face recognition under difficult lighting conditions". In: *Image Processing, IEEE Transactions on* 19.6 (2010), pp. 1635–1650.

[25] Philippe Thévenaz, Thierry Blu, and Michael Unser. "Image interpolation and resampling". In: *Handbook of medical imaging, processing and analysis* (2000), pp. 393–420.

[26] Stewart Tseng. "Comparison of holistic and feature based approaches to face recognition". PhD thesis. School of Computer Science and Information Technology, Faculty of Applied Science, Royal Melbourne Institute of Technology, 2003.

[27] Matthew Turk and Alex Pentland. "Eigenfaces for recognition". In: *Journal of cognitive neuroscience* 3.1 (1991), pp. 71–86.

[28] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–511.

[29] Laurenz Wiskott et al. "Face recognition by elastic bunch graph matching". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19.7 (1997), pp. 775–779.

[30] Zhenyao Zhu et al. "Deep learning identity-preserving face space". In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 113–120.