Face Recognition Technology

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree

Of

Bachelor of Technology

In

Computer Science and Engineering

By

Harneet Kumar (11172662)

Under the supervision of

Dr. Sandhya Bansal

Supervisor Designation

Department of Computer Science & Engineering

MAHARISHI
MARKANDESHWAR
(DEEMED TO BE UNIVERSITY)

M. M. Engineering College, Mullana, Ambala

Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala, Haryana, India

December 2019

INDEX

CONTENTS

Face Recognition Technology

Face recognition is the task of identifying an already detected object as a known or unknown face.

Candidate Declaration

I hereby certify that the work which is being presented in the project entitled “ Face Recognition Technology ” in fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering of M.M Engineering College, Mullana, Ambala, Haryana, India is an authentic record of my own work carried out during a period from July 2018 to December 2018, under the supervision of Dr. Sandhya Bansal (Designation). The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute/University.

Harneet Kumar (11172662)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: **Name & Signature of supervisor**

Head of Department
Department of Computer Engineering

Acknowledgement

I wish to express my deep sense of indebtedness and sincerest gratitude to my guide, **Dr. Sandhya Bansal (Designation), Department of Computer Science & Engineering, M. M. Engineering College, Mullana, Ambala, Haryana,** for his invaluable guidance and constructive criticism throughout this dissertation. He has displayed unique tolerance and understanding at every step of progress and encourages me. I deem it my privilege to have carried out my Dissertation work under his able guidance.

I would especially like to thank **Dr. Sandip Kumar Goel (Professor and Head), and Coordinator Dr Suneet Kumar (Associate Professor) Department of Computer Science & Engineering, M.M Engineering College, Mullana, Ambala**, without whom, this work would not have been as it is now.

As a Final Personal Note, I am grateful to my parents, who are inspirational to me in their understanding, patience and constant encouragement.

Harneet Kumar (11172662)

FACE RECOGNITION TECHNOLOGY

ABSTRACT

The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. There are two kinds of methods that are currently popular in developed face recognition pattern namely, Eigenface method and Fisher face method. Facial image recognition Eigenface method is based on the reduction of face dimensional space using Principal Component Analysis (PCA) for facial features. The main purpose of the use of PCA on face recognition using Eigen faces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project face detection system with face recognition is Image processing. The software requirements for this project is matlab software.

Keywords: face detection, Eigen face, PCA, matlab

Extension: There are vast number of applications from this face detection project, this project can be extended that the various parts in the face can be detect which are in various directions and shapes.

CHAPTER-1

INTRODUCTION

Face recognition is the task of identifying an already detected object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

1.1 FACE RECOGNIZATION:

DIFFERENT APPROACHES OF FACE RECOGNITION:

There are two predominant approaches to the face recognition problem: Geometric (feature based) and photometric (view based). As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature.

Recognition algorithms can be divided into two main approaches:

1. **Geometric:** Is based on geometrical relationship between facial landmarks, or in other words the spatial configuration of facial features. That means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified on the basis of various geometrical distances and angles between features. (Figure 3)
2. **Photometric stereo:** Used to recover the shape of an object from a number of images taken under different lighting conditions. The shape of the recovered object is defined by a gradient map, which is made up of an array of surface normals (Zhao and Chellappa, 2006) (Figure 2)

Popular recognition algorithms include:

1. Principal Component Analysis using Eigenfaces, (PCA)
2. Linear Discriminate Analysis,
3. Elastic Bunch Graph Matching using the Fisher face algorithm,

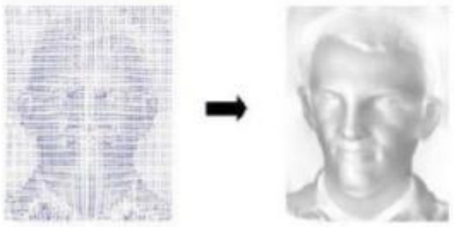


Figure 2 -Photometric stereo image.

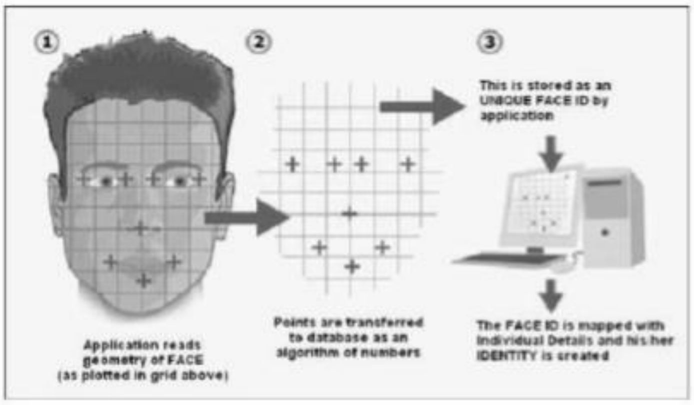

①
②
③
This is stored as an UNIQUE FACE ID by application
Application reads geometry of FACE (as plotted in grid above)
Points are transferred to database as an algorithm of numbers
The FACE ID is mapped with Individual Details and his/her IDENTITY is created

Figure 3 - Geometric facial recognition.

1.2 FACE DETECTION:

Face detection involves separating image windows into two classes; one containing faces (tarning the background (clutter). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin colour 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).

The face detection system can be divided into the following steps :-

1. **Pre-Processing:** To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping

images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.

2. Classification: Neural networks are implemented to classify the images as faces or nonfaces by training on these examples. We use both our implementation of the neural network and the Matlab neural network toolbox for this task. Different network configurations are experimented with to optimize the results.

3. Localization: The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on:- Position Scale Orientation Illumination

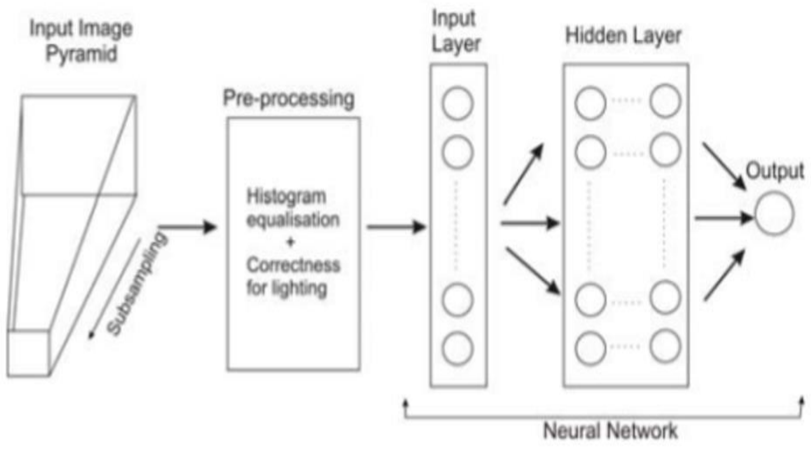


Fig: Face detection algorithm

CHAPTER-2

LITERATURE SURVEY

Face detection is a computer technology that determines the location and size of human face in arbitrary (digital) image. The facial features are detected and any other objects like trees, buildings and bodies etc are ignored from the digital image. It can be regarded as a _specific‘ case of object-class detection, where the task is finding the location and sizes of all objects in an image that belong to a given class. Face detection, can be regarded as a more _general‘ case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). Basically there are two types of approaches to detect facial part in the given image i.e. feature base and image base approach.Feature base approach tries to extract features of the image and match it against the knowledge of the face features. While image base approach tries to get best match between training and testing images.

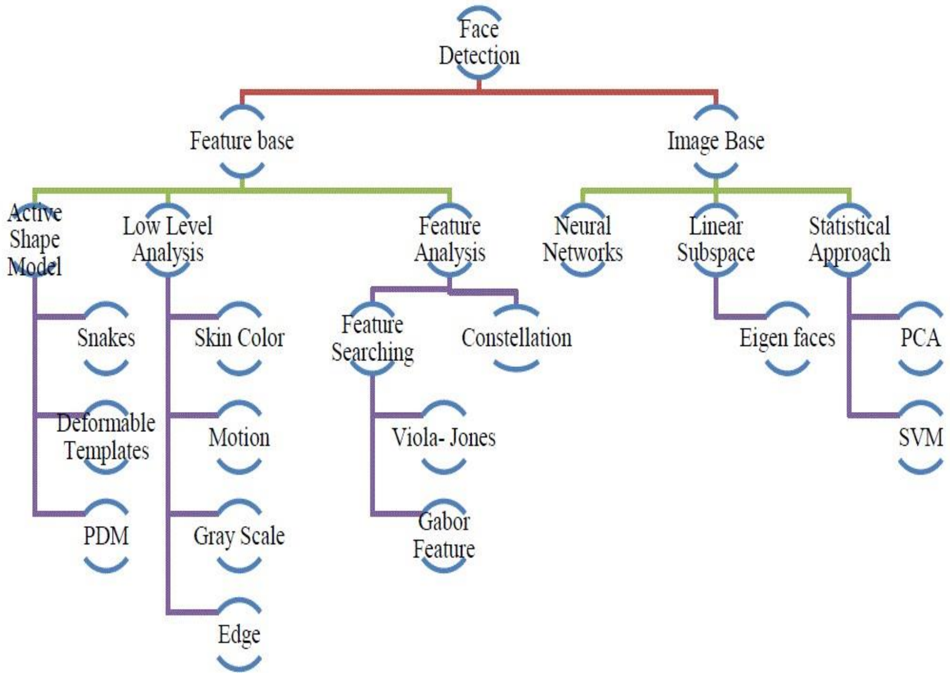


Fig 2.1 detection methods

FEATURE BASE APPROCH:

Active Shape ModelActive shape models focus on complex non-rigid features like actual physical and higher level appearance of features Means that Active Shape Models (ASMs) are aimed at automatically locating landmark points that define the shape of any statistically modelled object in an image. When of facial features such as the eyes, lips, nose, mouth and eyebrows. The training stage of an ASM involves the building of a statistical

a) facial model from a training set containing images with manually annotated landmarks. ASMs is classified into three groups i.e. snakes, PDM, Deformable templates

b) 1.1) Snakes:The first type uses a generic active contour called snakes, first introduced by Kass et al. in 1987 Snakes are used to identify head boundaries [8,9,10,11,12]. In order to achieve the task, a snake is first initialized at the proximity around a head boundary. It then locks onto nearby edges and subsequently assume the shape of the head. The evolution of a snake is achieved by minimizing an energy function, E snake (analogy with physical systems), denoted as E snake = E internal + E External Where E internal and E External are internal and external energy functions. Internal energy is the part that depends on the intrinsic properties of the snake and defines its natural evolution. The typical natural evolution in snakes is shrinking or expanding. The external energy counteracts the internal energy and enables the contours to deviate from the natural evolution and eventually assume the shape of nearby features—the head boundary at a state of equilibria. Two main consideration for forming snakes i.e. selection of energy terms and energy minimization. Elastic energy is used commonly as internal energy. Internal energy is vary with the distance between control points on the snake, through which we get contour an elastic-band characteristic that causes it to shrink or expand. On other side external energy relay on image features. Energy minimization process is done by optimization techniques such as the steepest gradient descent. Which needs highest computations. Huang and Chen and Lam and Yan both employ fast iteration methods by greedy algorithms. Snakes have some demerits like contour often becomes trapped onto false image features and another one is that snakes are not suitable in extracting non convex features.

2.1.1 Deformable Templates:

Deformable templates were then introduced by Yuille et al. to take into account the a priori of facial features and to better the performance of snakes. Locating a facial feature boundary is not an easy task because the local evidence of facial edges is difficult to organize into a sensible global entity using generic contours. The low brightness contrast around some of these features also makes the edge detection process. Yuille al. took the concept of snakes a step further by incorporating global information of the eye to improve the reliability of the extraction process.

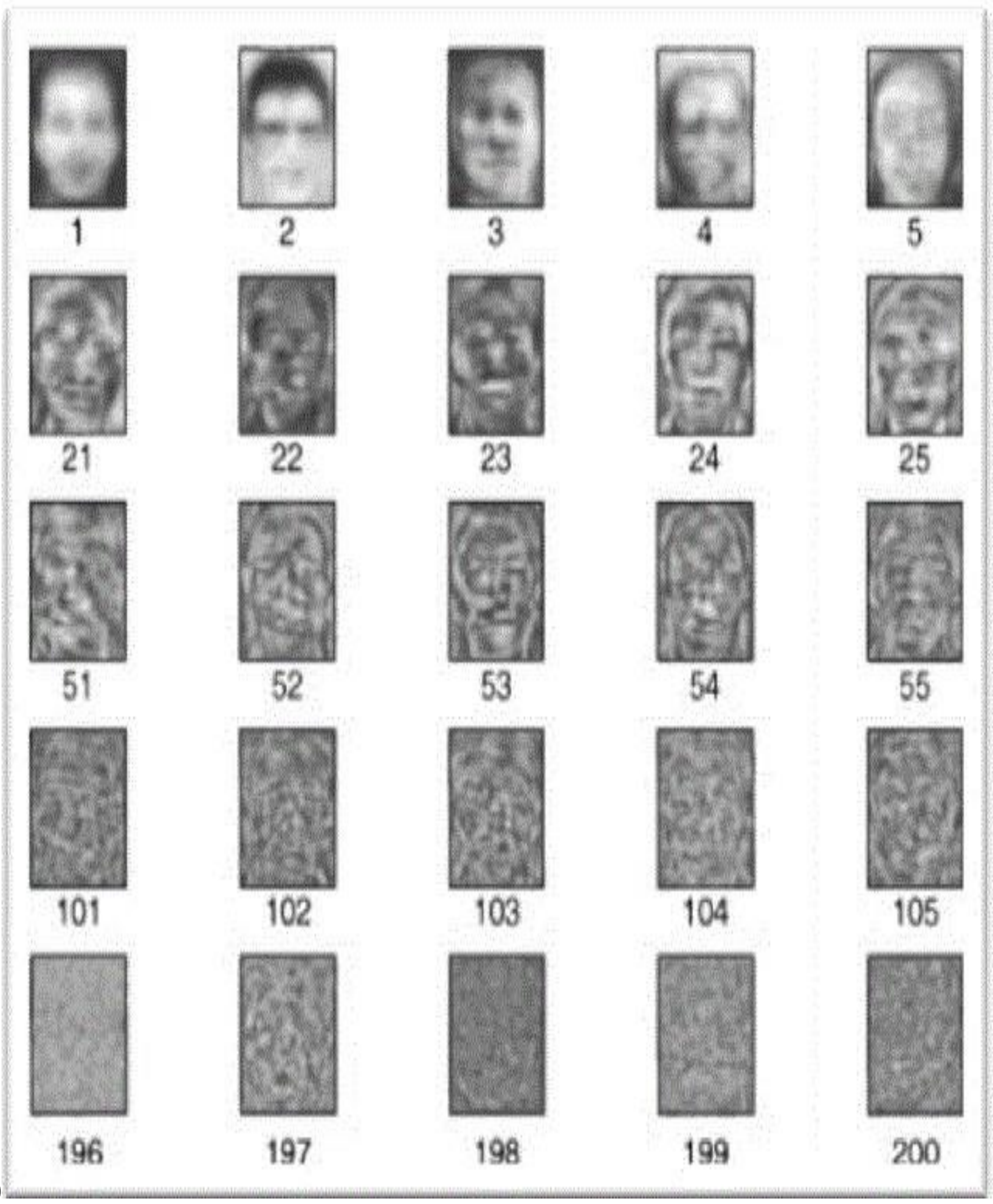
Deformable templates approaches are developed to solve this problem. Deformation is based on local valley, edge, peak, and brightness .Other than face boundary, salient feature (eyes, nose, mouth and eyebrows) extraction is a great challenge of face recognition = Ev + Ee + Ep + Ei + Enteral ; where Ev , Ee , Ep , Ei , Enteral are external energy due to valley, edges, peak and image brightness and internal energy

2.1.2 PDM (Point Distribution Model):

Independently of computerized image analysis, and before ASMs were developed, researchers developed statistical models of shape . The idea is that once you represent shapes as vectors, you can apply standard statistical methods to them just like any other multivariate object. These models learn allowable constellations of shape points from training example sand use principal components to build what is called a Point Distribution Model. These have been used in diverse ways, for example for categorizing Iron Age broaches. Ideal Point Distribution Models can only deform in ways that are characteristic of the object. Coots and his colleagues were seeking models which do exactly that so if a beard, say, covers the chin, the shape model can \override the image" to approximate the position of the chin under the beard. It was therefore natural (but perhaps only in retrospect) to adopt Point Distribution Models. This synthesis of ideas from image processing and statistical shape modelling led to the Active Shape Model. The first parametric statistical shape model for image analysis based on principal components of inter-landmark distances was presented by Coots and Taylor in. On this approach, Coots, Taylor, and their colleagues, then released a series of papers that cumulated in what we call the classical Active Shape Model.

2.2) LOW LEVEL ANALYSIS:

Based on low level visual features like color, intensity, edges, motion etc. Skin Color Base Color is avital feature of human faces. Using skin-color as a feature for tracking a face has several advantages. Color processing is much faster than processing other facial features. Under certain lighting conditions, color is orientation invariant. This property makes motion estimation much easier because only a translation model is needed for motion estimation. Tracking human faces using color as a feature has several problems like the color representation of a face obtained by a camera is influenced by many factors (ambient light, object movement, etc.



.)

Fig 2.2. face detection

Majorly three different face detection algorithms are available based on RGB, YCbCr, and HIS color space models. In the implementation of the algorithms there are three main steps viz.

(1) Classify the skin region in the color space,

(2) Apply threshold to mask the skin region and

(3) Draw bounding box to extract the face image.

Crowley and Couto suggested simplest skin color algorithms for detecting skin pixels. The perceived human color varies as a function of the relative direction to the illumination.

The pixels for skin region can be detected using a normalized color histogram, and can be normalized for changes in intensity on dividing by luminance. Converted an [R, G, B] vector is converted into an [r, g] vector of normalized color which provides a fast means of skin detection. This algorithm fails when there are some more skin region like legs, arms, etc. and Ngan [27] suggested skin color classification algorithm with CyBC color space. Research found that pixels belonging to skin region having similar Cob and Cr values. So that the thresholds be chosen as [Cr1, Cr2] and [Cb1, Cb2], a pixel is classified to have skin tone if the values [Cr, Cb] fall within the thresholds. The skin color distribution gives the face portion in the color image. This algorithm is also having the constraint that the image should be having only face as the skin region. Kjeldsen and Kinder defined a color predicate in HSV color space to separate skin regions from background. Skin color classification inHSI color space is the same as YCbCr color space but here the responsible values are hue (H) and saturation (S). Similar to above the threshold be chosen as [H1, S1] and [H2, S2], and a pixel is classified to have skin tone if the values [H,S] fall within the threshold and this distribution gives the localized face image. Similar to above two algorithm this algorithm is also having the same constraint.

MOTION BASE:

When use of video sequence is available, motion information can be used to locate moving objects. Moving silhouettes like face and body parts can be extracted by simply thresholding accumulated frame differences . Besides face regions, facial feature scan be located by frame differences .

Gray Scale Base:

Edge Base:

Face detection based on edges was introduced by Sakai et al. This work was based on analyzing line drawings of the faces from photographs, aiming to locate facial features. Than later Craw et al. proposed a hierarchical framework based on Sakai et al.‘work to trace a human head outline. Then after remarkable works were carried out by many researchers in this specific area. Method suggested by Anile and Devarajan was very simple and fast. They proposed frame work which consist three steps i.e. initially the images are enhanced by applying median filter for noise removal and histogram equalization for contrast adjustment. In the second step the edge image is constructed from the enhanced image by applying sober operator. Then a novel edge tracking algorithm is applied to extract the sub windows from the enhanced image based on edges. Further they used Back propagation Neural Network (BPN) algorithm to classify the sub-window as either face or non-face.

FEATURE ANALYSIS

These algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions vary, and then use these to locate faces. These methods are designed mainly for face localization.

Feature Searching

Viola Jones Method:

Paul Viola and Michael Jones presented an approach for object detection which minimizes computation time while achieving high detection accuracy. Paul Viola and Michael Jones [39] proposed a fast and robust method for face detection which is 15 times quicker than any technique at the time of release with 95% accuracy at around 17 fps. The technique relies on the use of simple Haar-like features that are evaluated quickly through the use of a new image representation. Based on the concept of an ―Integral Image‖ it generates a large set of features and uses the boosting algorithm AdaBoost to reduce the overcomplete set and the introduction of a degenerative tree of the boosted classifiers provides for robust and fast interferences. The detector is applied in a scanning fashion and used on gray-scale images, the scanned window that is applied can also be scaled, as well as the features evaluated.

CONSTELLATION METHOD

All methods discussed so far are able to track faces but still some issue like locating faces of various poses in complex background is truly difficult. To reduce this difficulty investigator form a group of facial features in face-like constellations using more robust modelling approaches such as statistical analysis. Various types of face constellations have been proposed by Burl et al. . They establish use of statistical shape theory on the features detected from a multiscale Gaussian derivative filter. Huang et al. also apply a Gaussian filter for pre-processing in a framework based on image feature analysis.Image Base Approach.

Neural Network

Neural networks gaining much more attention in many pattern recognition problems, such as OCR, object recognition, and autonomous robot driving. Since face detection can be treated as a two class pattern recognition problem, various neural network algorithms have been proposed.

The advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class conditional density of face patterns. However, one demerit is that the network architecture has to be extensively tuned (number of layers, number of nodes, learning rates, etc.) to get exceptional performance. In early days most hierarchical neural network was proposed by Ague et al. [43]. The first stage having two parallel subnetworks in which the inputs are filtered intensity values from an original image. The inputs to the second stage network consist of the outputs from the sub networks and extracted feature values. An output at the second stage shows the presence of a face in the input region. Propp and Samael developed one of the earliest neural networks for face detection [44]. Their network consists off our layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units. Fraud and Bernier presented a detection method using auto associative neural networks [45], [46], [47]. The idea is based on [48] which shows an auto associative network with five layers is able to perform a nonlinear principal component analysis. One auto associative network is used to detect frontal view faces and another one is used to detect faces turned up to 60 degrees to the left and right of the frontal view. After that Lin et al. presented a face detection system using probabilistic decision-based neural network (PDBNN) [49]. The architecture of PDBNN is similar to a radial basis function (RBF) network with modified learning rules and probabilistic interpretation.

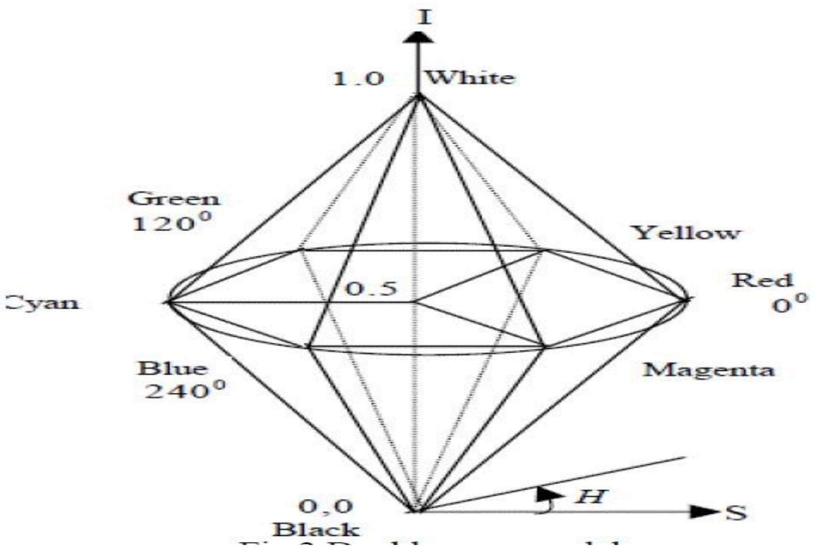
LINEAR SUB SPACE METHOD

Eigen faces Method:

An early example of employing eigen vectors in face recognition was done by Korhonen in which a simple neural network is demonstrated to perform face recognition for aligned and normalized face images. Kirby and Siro ich suggested that images of faces can be linearly encoded using a modest number of basis images. The idea is arguably proposed first by Pearson in 1901 and then hoteling in 1933 .Given a collection of n by m pixel training.

Images represented as a vector of size m X n, basis vectors spanning an optimal subspace are determined such that the mean square error between the projection of the training images onto this subspace and the original images is minimized. They call the set of optimal basis vectors Eigen pictures since these are simply the eigen vectors of the covariance matrix computed from the vectorized face images in the training set. Experiments with a set of 100 images show that a face image of 91 X 50 pixels can be effectively encoded using only50 Eigen pictures.

A reasonable likeness (i.e.,capturing 95 percent of thevariance)



STATISTICAL APPROCH

Support Vector Machine (SVM):

SVMs were first introduced Osuna et al. for face detection. SVMs work as a new paradigm to train polynomial function, neural networks, or radial basis function (RBF) classifiers.SVMs works on induction principle, called structural risk minimization, which targets to minimize an upper bound on the expected generalization error. An SVM classifier is a linear classifier where the separating hyper plane is chosen to minimize the expected classification error of the unseen test patterns. In Ozonate al. developed an efficient method to train an SVM for large scale problems, and applied it to face detection. Based on two test sets of 10,000,000 test patterns of 19 X 19 pixels, their system has slightly lower error rates and runs approximately30 times faster than the system by Sung and Piaggio . SVMs have also been used to detect faces and pedestrians in the wavelet domain.

CHAPTER-3

DIGITAL IMAGE PROCESSING

DIGITAL IMAGE PROCESSING

Interest in digital image processing methods stems from two principal application areas:

1. Improvement of pictorial information for human interpretation
2. Processing of scene data for autonomous machine perception

In this second application area, interest focuses on procedures for extracting image information in a form suitable for computer processing.

Examples includes automatic character recognition, industrial machine vision for product assembly and inspection, military recognizance, automatic processing of fingerprints etc.

Image:

Am image refers a 2D light intensity function f(x, y), where(x, y) denotes spatial coordinates and the value of f at any point (x, y) is proportional to the brightness or gray levels of the image at that point. A digital image is an image f(x, y) that has been discretized both in spatial coordinates and brightness. The elements of such a digital array are called image elements or pixels.

A simple image model:

To be suitable for computer processing, an image f(x, y) must be digitalized both spatially and in amplitude. Digitization of the spatial coordinates (x, y) is called image sampling. Amplitude digitization is called gray-level quantization.

The storage and processing requirements increase rapidly with the spatial resolution and the number of gray levels.

Example: A 256 gray-level image of size 256x256 occupies 64k bytes of memory.

Types of image processing

- Low level processing
- Medium level processing
- High level processing

Low level processing means performing basic qperations on images such as reading an image resize, resize, image rotate, RGB to gray level conversion, histogram equalization etc…, The output image obtained after low level processing is raw image. Medium level processing means extracting regions of interest from output of low level processed image. Medium level processing deals with identification of boundaries i.e edges .This process is called segmentation. High level processing deals with adding of artificial intelligence to medium level processed signal.

FUNDAMENTAL STEPS IN IMAGE PROCESSING

Fundamental steps in image processing are

1. Image acquisition: to acquire a digital image
2. Image pre-processing: to improve the image in ways that increases the chances for success of the other processes.
3. Image segmentation: to partitions an input image into its constituent parts of objects.
4. Image segmentation: to convert the input data to a from suitable for computer processing.
5. Image description: to extract the features that result in some quantitative information of interest of features that are basic for differentiating one class of objects from another.
6. Image recognition: to assign a label to an object based on the information provided by its description.

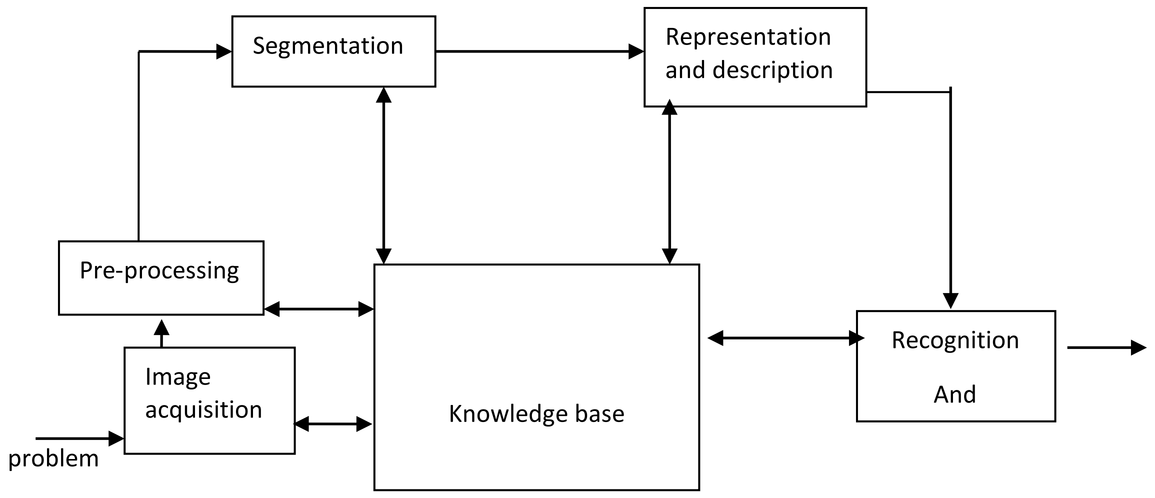


fig.3.1. Fundamental steps in digital image processing

ELEMENTS OF DIGITAL IMAGE PROCESSING SYSTEMS

A digital image processing system contains the following blocks as shown in the figure

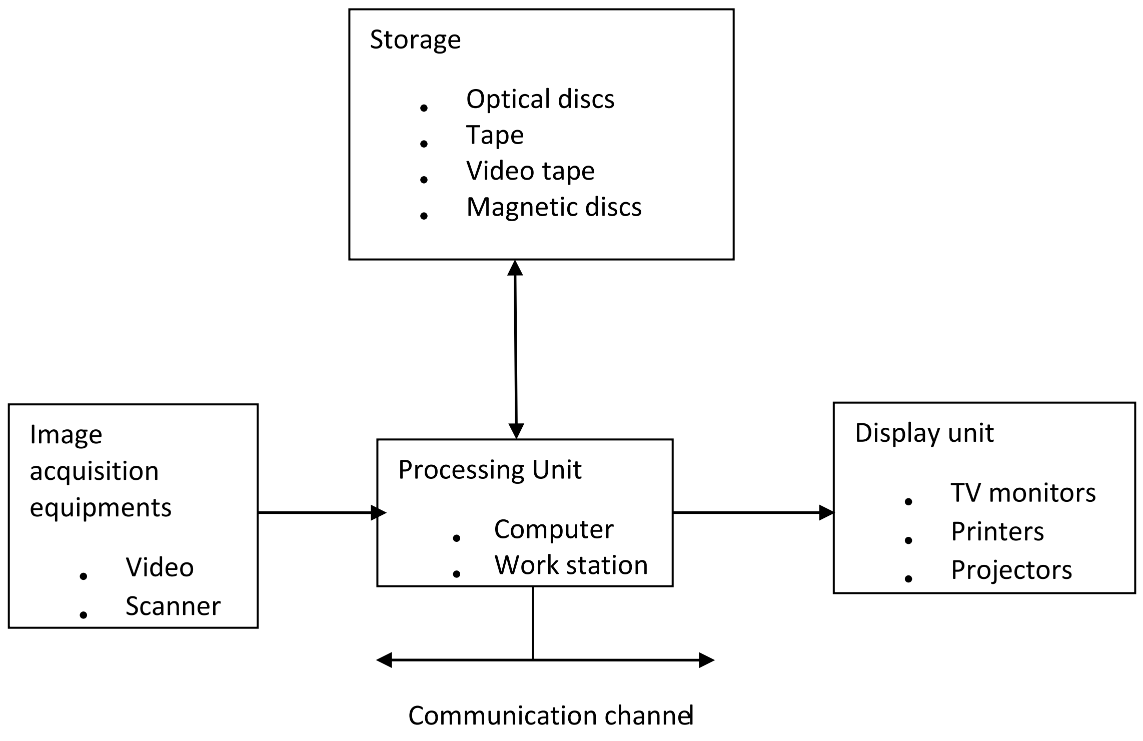


Fig.3.3. Elements of digital image processing systems

The basic operations performed in a digital image processing system include

1. Acquisition
2. Storage
3. Processing
4. Communication
5. Display

A simple image formation model

Image are denoted by two-dimensional function f(x, y).f(x, y) may be characterized by 2 components:

1. The amount of source illumination i(x, y) incident on the scene
2. The amount of illumination reflected r(x, y) by the objects of the scene
3. $f(x, y) = i(x, y)r(x, y)$, where $0 < i(x,y) <$ and $0 < r(x, y) < 1$

Typical values of reflectance r(x, y):

- 0.01 for black velvet
- 0.65 for stainless steel
- 0.8 for flat white wall paint
- 0.9 for silver-plated metal
- 0.93 for snow Example of typical ranges of illumination i(x, y) for visible light

(average values)

- Sun on a clear day: ~90,000 lm/m^2,down to 10,000lm/m^2 on a cloudy day
- Full moon on a clear evening :-0.1 lm/m^2
- Typical illumination level in a commercial office. ~1000lm/m^2 image Formats (supported by MATLAB Image Processing Toolbox)

Format name	Full name	Description	Recognized extensions
TIFF	Tagged Image File Format	A flexible file format supporting a variety image compression standards including JPEG	.tif, .tiff
JPEG	Joint Photographic Experts Group	A standard for compression of images of photographic quality	.jpg, .jpeg
GIF	Graphics Interchange Format	Frequently used to make small animations on the internet	.gif
BMP	Windows Bitmap	Format used mainly for simple uncompressed images	.bmp
PNG	Portable Network Graphics	Compresses full color images with trasparency(up to 48bits/p	.png

Table.3.3. Image Formats Supported by Matlab

CHAPTER-4

FACE DETECTION

The problem of face recognition is all about face detection. This is a fact that seems quite bizarre to new researchers in this area. However, before face recognition is possible, one must be able to reliably find a face and its landmarks. This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task. In fact the actual recognition based on features extracted from these facial landmarks is only a minor last step.

There are two types of face detection problems:

1) Face detection in images and
2) Real-time face detection

FACE DETECTION IN IMAGES

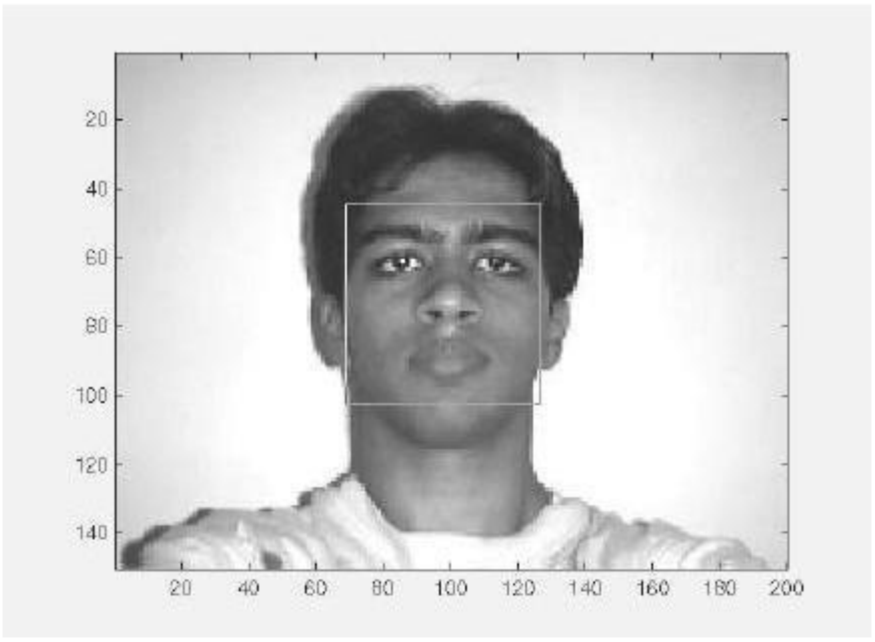


Figure 5.1 A successful face detection in an image with a frontal view of a human face.

Most face detection systems attempt to extract a fraction of the whole face, thereby eliminating most of the background and other areas of an individual's head such as hair that are not necessary for the face recognition task. With static images, this is often done by running a across the image. The face detection system then judges if a face is present inside the window (Brunelli and Poggio, 1993). Unfortunately, with static images there is a very large search space of possible locations of a face in an image

Most face detection systems use an example based learning approach to decide whether or not a face is present in the *window* at that given instant (Sung and Poggio,1994 and Sung,1995). A neural network or some other classifier is trained using supervised learning with 'face' and 'nonface' examples, thereby enabling it to classify an image (*window* in face detection system) as a 'face' or 'non-face'.. Unfortunately, while it is relatively easy to find face examples, how would one find a representative sample of images which represent non-faces (Rowley et al., 1996)? Therefore, face detection systems using example based learning need thousands of 'face' and 'nonface' images for effective training. Rowley, Baluja, and Kanade (Rowley et al.,1996) used 1025 face images and 8000 non-face images (generated from 146,212,178 sub-images) for their training set!

There is another technique for determining whether there is a face inside the face detection system's *window* - using Template Matching. The difference between a fixed target pattern (face) and the window is computed and thresholded. If the window contains a pattern which is close to the target pattern(face) then the window is judged as containing a face. An implementation of template matching called Correlation Templates uses a whole bank of fixed sized templates to detect facial features in an image (Bichsel, 1991 & Brunelli and Poggio, 1993). By using several templates of different (fixed) sizes, faces of different scales (sizes) are detected. The other implementation of template matching is using a deformable template (Yuille, 1992). Instead of using several fixed size templates, we use a deformable template (which is non-rigid) and there by change the size of the template hoping to detect a face in an image.

A face detection scheme that is related to template matching is image invariants. Here the fact that the local ordinal structure of brightness distribution of a face remains largely unchanged under different illumination conditions (Sinha, 1994) is used to construct a spatial template of the face which closely corresponds to facial features. In other words, the average grey-scale intensities in human faces are used as a basis for face detection. For example, almost always an individuals eye region is darker than his forehead or nose. Therefore an image will match the template if it satisfies the 'darker than' and 'brighter than' relationships (Sung and Poggio, 1994).

REAL-TIME FACE DETECTION

Real-time face detection involves detection of a face from a series of frames from a videocapturing device. While the hardware requirements for such a system are far more stringent, from a computer vision stand point, real-time face detection is actually a far simpler process thandetecting a face in a static image. This is because unlike most of our surrounding environment, people are continually moving. We walk around, blink, fidget, wave our hands about, etc.

Figure: Frame 1 from camera

Figure: Frame 2 from camera

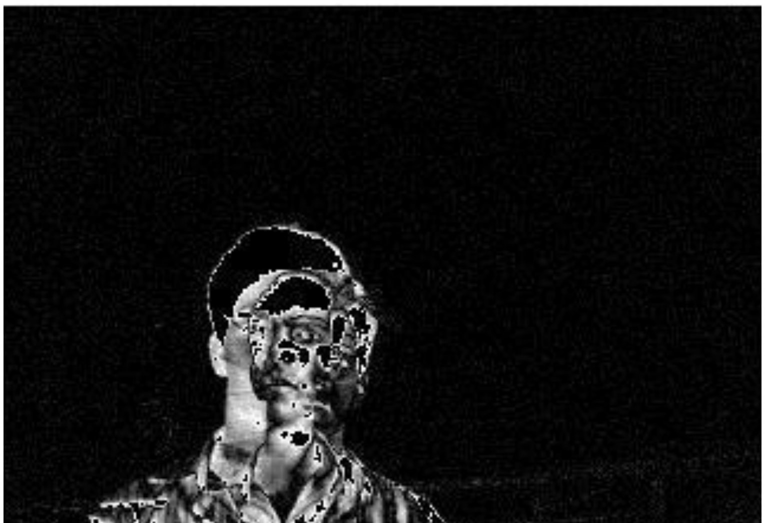


Figure: Spatio-Temporally filtered image

Since in real-time face detection, the system is presented with a series of frames in which to detect a face, by using spatio-temperal filtering (finding the difference between subsequent frames), the area of the frame that has changed can be identified and the individual detected (Wang and Adelson, 1994 and Adelson and Bergen 1986).Further more as seen in Figure exact face locations can be easily identified by using a few simple rules, such as,

1)the head is the small blob above a larger blob -the body

2)head motion must be reasonably slow and contiguous -heads won't jump around erratically (Turk and Pentland 1991a, 1991b).

Real-time face detection has therefore become a relatively simple problem and is possible even in unstructured and uncontrolled environments using these very simple image processing techniques and reasoning rules.

FACE DETECTION PROCESS

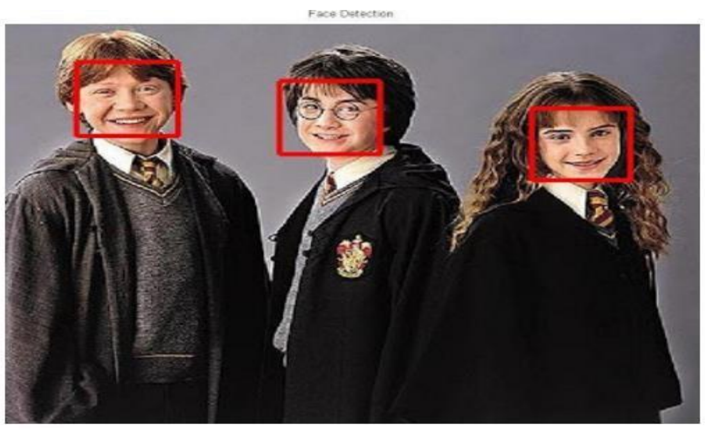
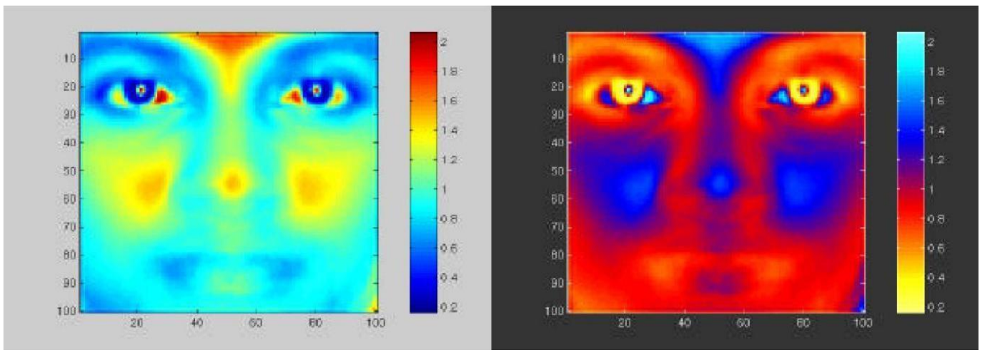


Fig Face detection

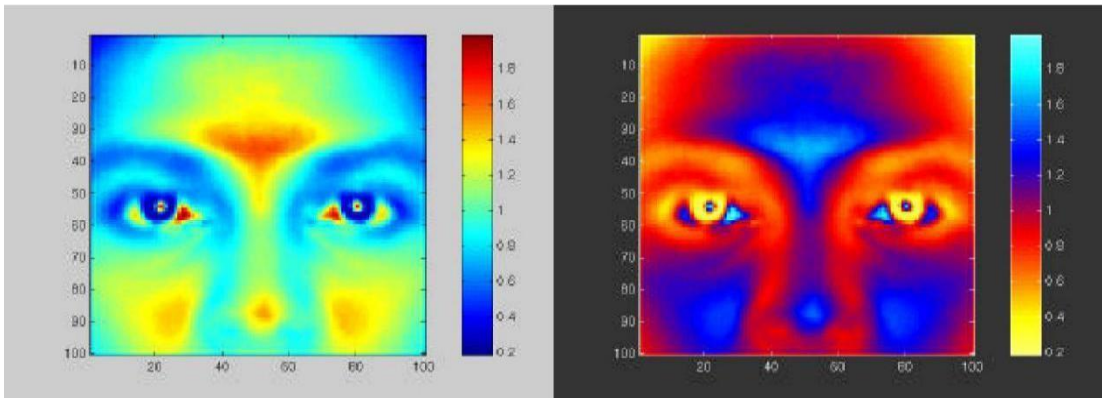
It is process of identifying different parts of human faces like eyes, nose, mouth, etc… this process can be achieved by using MATLAB codeIn this project the author will attempt to detect faces in still images by using image invariants. To do this it would be useful to study the greyscale intensity distribution of an average human face. The following 'average human face' was constructed from a sample of 30 frontal view human faces, of which 12 were from females and 18 from males. A suitably scaled colormap has been used to highlight grey-scale intensity differences.



scaled colormap scaled colormap (negative)

Figure 5.3.1 Average human face in grey-scale

The grey-scale differences, which are invariant across all the sample faces are strikingly apparent. The eye-eyebrow area seem to always contain dark intensity (low) gray-levels while nose forehead and cheeks contain bright intensity (high) grey-levels. After a great deal of experimentation, the researcher found that the following areas of the human face were suitable for a face detection system based on image invariants and a deformable template.



scaled colormap scaled colormap (negative)

Figure 5.3.2 Area chosen for face detection (indicated on average human face in gray scale)

The above facial area performs well as a basis for a face template, probably because of the clear divisions of the bright intensity invariant area by the dark intensity invariant regions. Once this pixel area is located by the face detection system, any particular area required can be segmented based on the proportions of the average human face After studying the above images it was subjectively decided by the author to use the following as a basis for dark intensity sensitive and bright intensity sensitive templates. Once these are located in a subject's face, a pixel area

33.3% (of the width of the square window) below this.

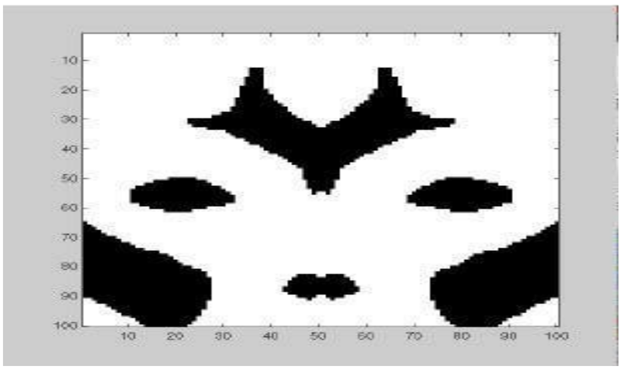


Figure 5.3.3: Basis for a bright intensity invariant sensitive template.

Note the slight differences which were made to the bright intensity invariant sensitive template (compare Figures 3.4 and 3.2) which were needed because of the pre-processing done by the system to overcome irregular lighting (chapter six). Now that a suitable dark and bright intensity invariant templates have been decided on, it is necessary to find a way of using these to make 2 A-units for a perceptron, i.e. a computational model is needed to assign neurons to the distributions displayed .

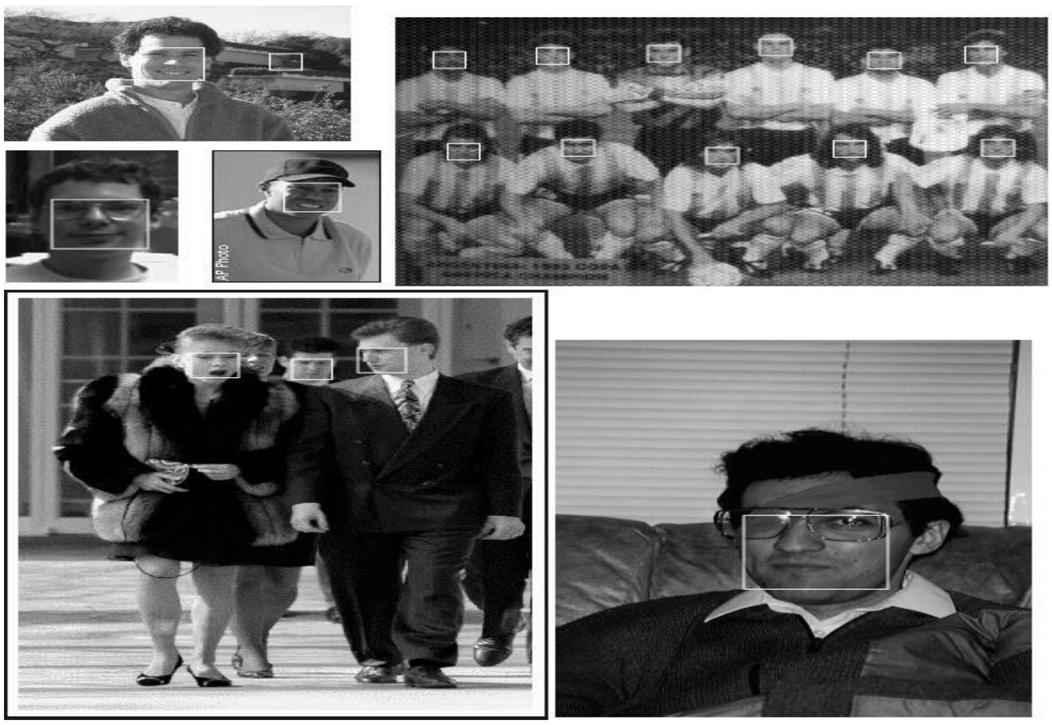


Fig Scaned image detection

- San window over image

- Clasify window as either
    1. Face
    2. Non - Face

FACE DETECTION ALGORITHM

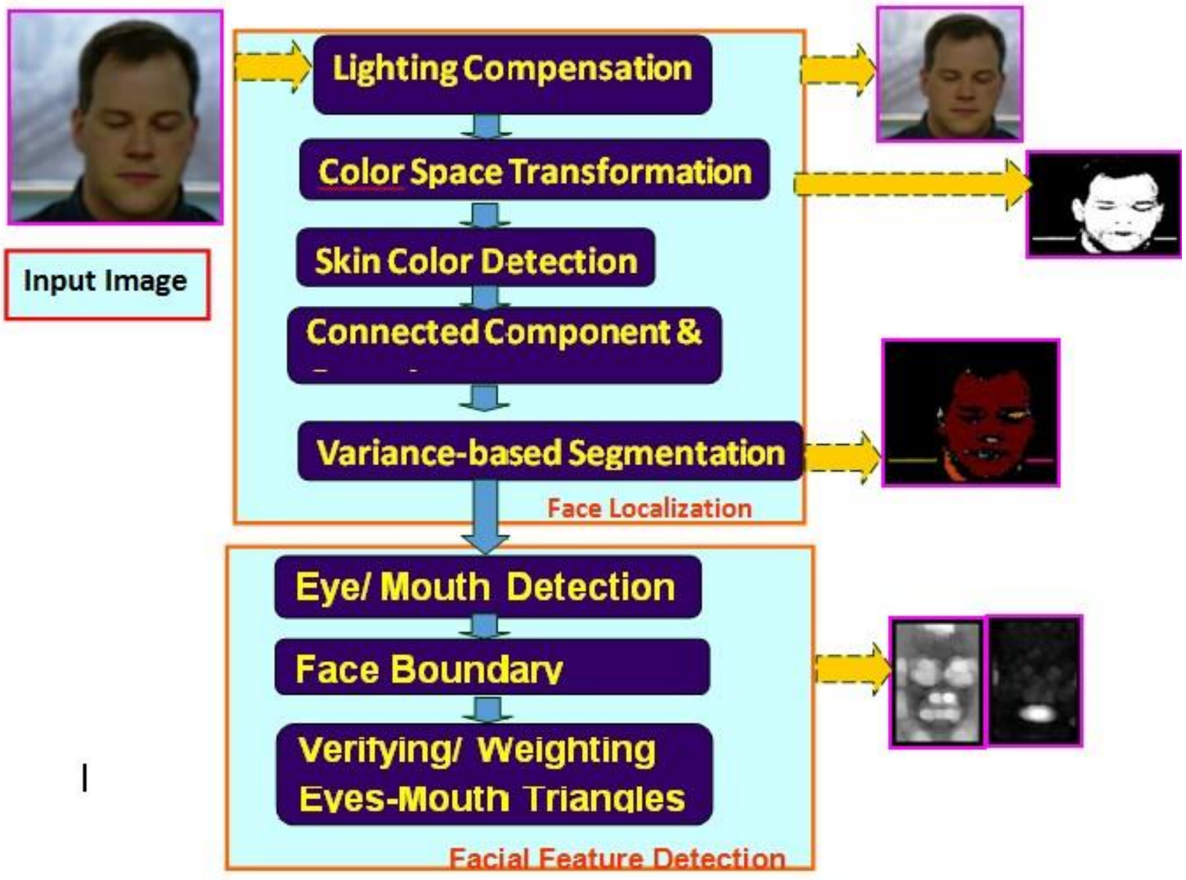


Fig 5.4 Face detection algorithm

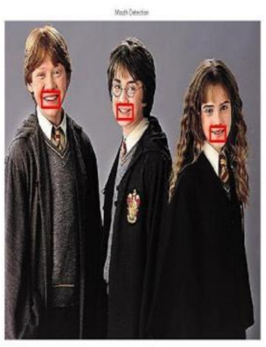
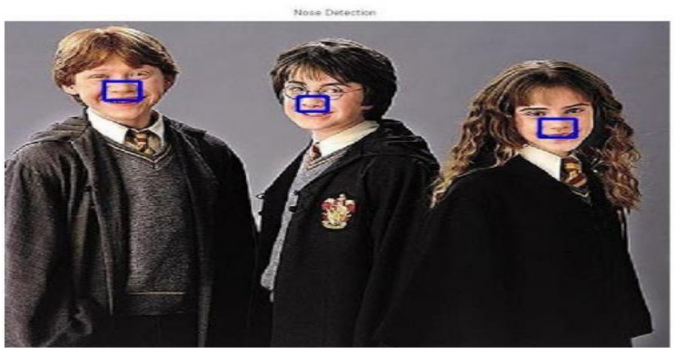



Fig 5.4.1 mouth detection

Fig 5.4.2 Noise detection

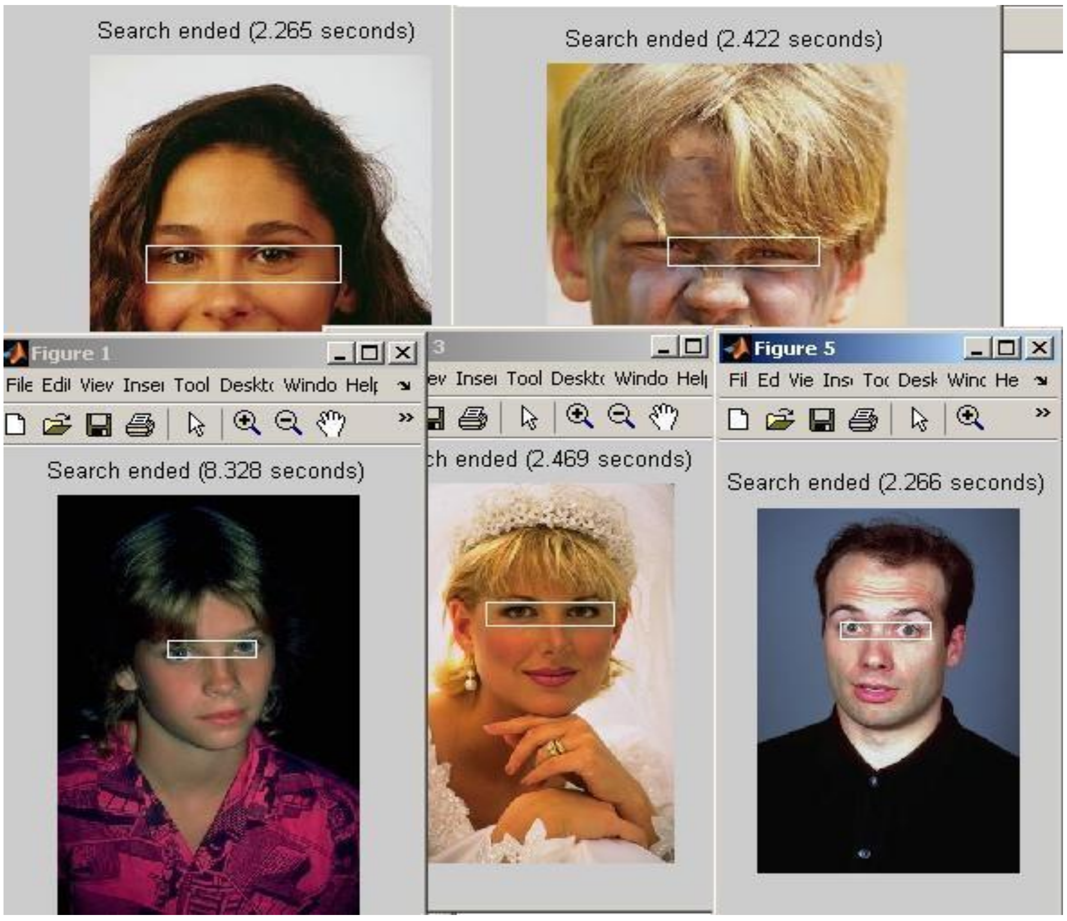

Search ended (2.265 seconds)
Search ended (2.422 seconds)
Figure 1
File Edit View Insert Tools Desktop Window Help
Search ended (8.328 seconds)
ch ended (2.469 seconds)
Figure 5
Search ended (2.266 seconds)

Fig 5.4.3 Eye detection

CHAPTER-5

FACE RECOGNITION

Over the last few decades many techniques have been proposed for face recognition. Many of the techniques proposed during the early stages of computer vision cannot be considered successful, but almost all of the recent approaches to the face recognition problem have been creditable. According to the research by Brunelli and Poggio (1993) all approaches to human face recognition can be divided into two strategies:

(1) Geometrical features and
(2) Template matching.

FACE RECOGNITION USING GEOMETRICAL FEATURES

This technique involves computation of a set of geometrical features such as nose width and length, mouth position and chin shape, etc. from the picture of the face we want to recognize. This set of features is then matched with the features of known individuals. A suitable metric such as Euclidean distance (finding the closest vector) can be used to find the closest match. Most pioneering work in face recognition was done using geometric features (Kanade, 1973), although Craw et al. (1987) did relatively recent work in this area.

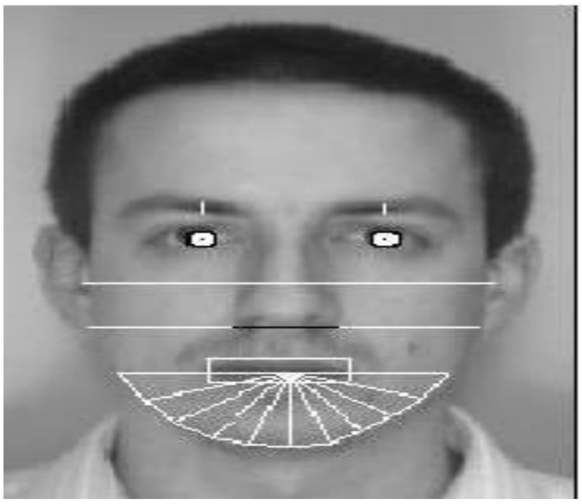


Figure 6.1 Geometrical features (white) which could be used for face recognition

The advantage of using geometrical features as a basis for face recognition is that recognition is possible even at very low resolutions and with noisy images (images with many disorderly pixel intensities). Although the face cannot be viewed in detail its overall geometrical configuration can be extracted for face recognition. The technique's main disadvantage is that automated extraction of the facial geometrical features is very hard. Automated geometrical feature extraction based recognition is also very sensitive to the scaling and rotation of a face in the image plane (Brunelli and Poggio, 1993). This is apparent when we examine Kanade's(1973) results where he reported a recognition rate of between 45-75 % with a database of only 20 people. However if these features are extracted manually as in Goldstein et al. (1971), and Kaya and Kobayashi (1972) satisfactory results may be obtained.

Face recognition using template matching

This is similar the template matching technique used in face detection, except here we are not trying to classify an image as a 'face' or 'non-face' but are trying to recognize a face.

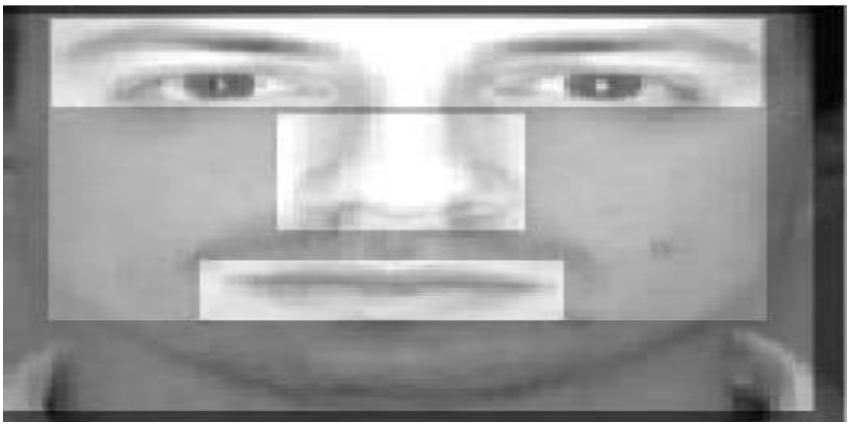


Figure Face recognition using template matching

Whole face, eyes, nose and mouth regions which could be used in a template matching strategy.The basis of the template matching strategy is to extract whole facial regions (matrix of pixels) and compare these with the stored images of known individuals. Once again Euclidean distance can be used to find the closest match. The simple technique of comparing grey-scale intensity values for face recognition was used by Baron (1981). However there are far more sophisticated methods of template matching for face recognition. These involve extensive preprocessing and transformation of the extracted grey-level intensity values. For example, Turk and Pentland (1991a) used Principal Component Analysis, sometimes known as the eigenfaces approach, to pre-process the gray-levels and Wiskott et al. (1997) used Elastic Graphs encoded using Gabor filters to pre-process the extracted regions. An investigation of geometrical features versus template matching for face recognition by Brunelli and Poggio (1993) came to the conclusion that although a feature based strategy may offer higher recognition speed and smaller memory requirements, template based techniques offer superior recognition accuracy.

PROBLEM SCOP AND SYSTEM SPECIFICATION

The following problem scope for this project was arrived at after reviewing the literature on face detection and face recognition, and determining possible real-world situations where such systems would be of use. The following system(s) requirements were identified

1 A system to detect frontal view faces in static images
2 A system to recognize a given frontal view face
3 Only expressionless, frontal view faces will be presented to the face detection&recognition 4 All implemented systems must display a high degree of lighting invariency.
5 All systems must posses near real-time performance.
6 Both fully automated and manual face detection must be supported
7 Frontal view face recognition will be realised using only a single known image
8 Automated face detection and recognition systems should be combined into a fully automated face detection and recognition system. The face recognition sub-system must display a slight degree of invariency to scaling and rotation errors in the segmented image extracted by the face detection sub-system.
9 The frontal view face recognition system should be extended to a pose invariant face recognition system.

Unfortunately although we may specify constricting conditions to our problem domain, it may not be possible to strictly adhere to these conditions when implementing a system in the realworld.

BRIEF OUT LINE OF THE IMPLEMENTED SYSTEM

Fully automated face detection of frontal view faces is implemented using a deformable template algorithm relying on the image invariants of human faces. This was chosen because a similar neural-network based face detection model would have needed far too much training data to be implemented and would have used a great deal of computing time. The main difficulties in implementing a deformable template based technique were the creation of the bright and dark intensity sensitive templates and designing an efficient implementation of the detection algorithm.

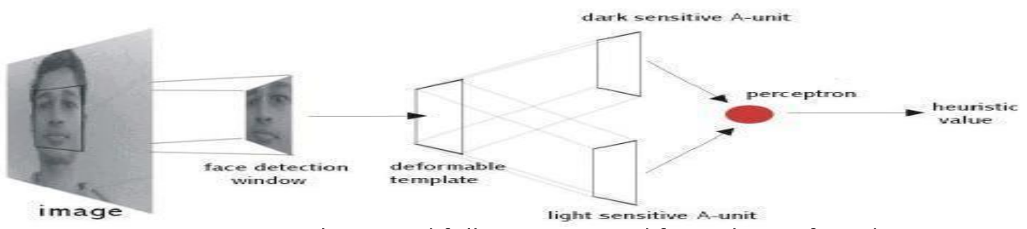


Figure 6.3 Implemented fully automated frontal view face detection model

A manual face detection system was realised by measuring the facial proportions of the average face, calculated from 30 test subjects. To detect a face, a human operator would identify the locations of the subject's eyes in an image and using the proportions of the average face, the system would segment an area from the image

A template matching based technique was implemented for face recognition. This was because of its increased recognition accuracy when compared to geometrical features based techniques and the fact that an automated geometrical features based technique would have required complex feature detection pre-processing.

Of the many possible template matching techniques, Principal Component Analysis was chosen because it has proved to be a highly robust in pattern recognition tasks and because it is relatively simple to implement. The author would also liked to have implemented a technique based on Elastic Graphs but could not find sufficient literature about the model to implement such a system during the limited time available for this project.

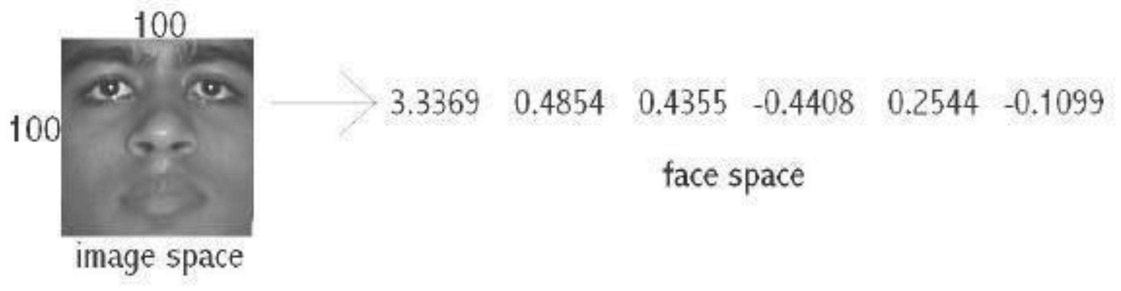


Figure 6.3.1: Principal Component Analysis transform from 'image space' to 'face space'.

Using Principal Component Analysis, the segmented frontal view face image is transformed from what is sometimes called 'image space' to 'face space'. All faces in the face database are transformed into face space. Then face recognition is achieved by transforming any given test image into face space and comparing it with the training set vectors. The closest matching training set vector should belong to the same individual as the test image.Principal

Component Analysis is of special interest because the transformation to face space is based on the variation of human faces (in the training set). The values of the 'face space' vector correspond to the amount certain 'variations' are present in the test image

Face recognition and detection system is a pattern recognition approach for personal identification purposes in addition to other biometric approaches such as fingerprint recognition, signature, retina and so forth. Face is the most common biometric used by humans applications ranges from static, mug-shot verification in a cluttered background.

Fig 6.3.2 Face Recognition

FACE RECOGNITION DIFFICULTIES

1. Identify similar faces (inter-class similarity)

2. Accommodate intra-class variability due to

2.1 head pose

2.2 illumination conditions

2.3 expressions

2.4 facial accessories

2.5 aging effects

3. Cartoon faces

6.4.1 Inter - class similarity:

 Different persons may have very similar appearance

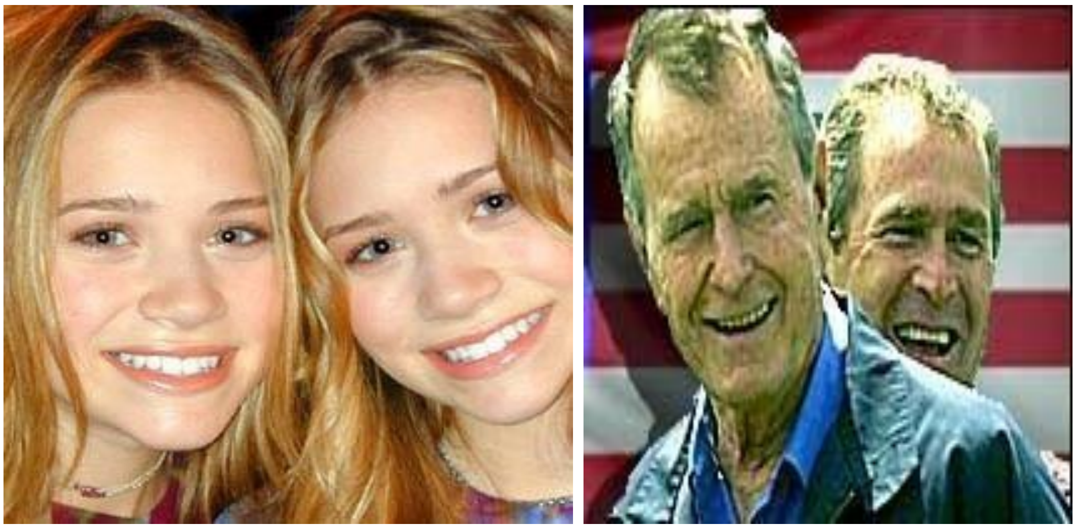


Fig 6.4.1 Face recognition twins and other and Son

Face recognition and detection system is a pattern recognition approach for personal identification purposes in addition to other biometric approaches such as fingerprint recognition, signature, retina and so forth. The variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.

Inter – class variability

Faces with intra-subject variations in pose, illumination, expression, accessories, color, occlusions, and brightnes

PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal Component Analysis (or Karhunen-Loeve expansion) is a suitable strategy for face recognition because it identifies variability between human faces, which may not be immediately obvious. Principal Component Analysis (hereafter PCA) does not attempt to categorise faces using familiar geometrical differences, such as nose length or eyebrow width. Instead, a set of human faces is analysed using PCA to determine which 'variables' account for the variance of faces. In face recognition, these variables are called eigen faces because when plotted they display an eerie resemblance to human faces. Although PCA is used extensively in statistical analysis, the pattern recognition community started to use PCA for classification only relatively recently. As described by Johnson and Wichern (1992), 'principal component analysis is concerned with explaining the variance- covariance structure through a few linear combinations of the original variables.' Perhaps PCA's greatest strengths are in its ability for data reduction and interpretation. For example a 100x100 pixel area containing a face can be very accurately represented by just 40 eigen values. Each eigen value describes the magnitude of each eigen face in each image. Furthermore, all interpretation (i.e. recognition) operations can now be done using just the 40 eigen values to represent a face instead of the manipulating the 10000 values contained in a 100x100 image. Not only is this computationally less demanding but the fact that the recognition information of several thousand.

UNDERSTANDING EIGENFACES

Any grey scale face image I(x,y) consisting of a NxN array of intensity values may also be consider as a vector of N^2. For example, a typical 100x100 image used in this thesis will have to be transformed into a 10000 dimension vector!

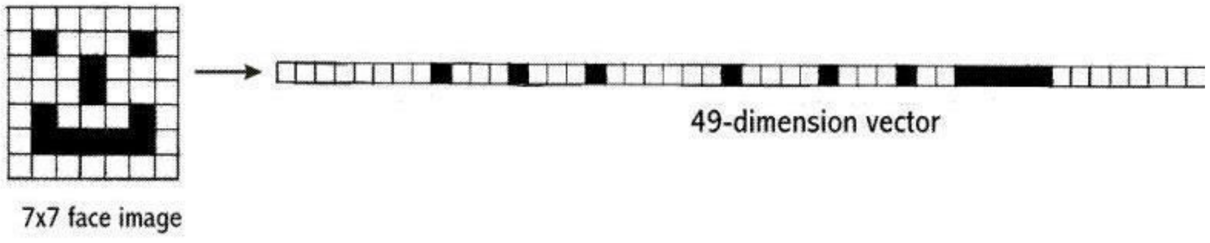
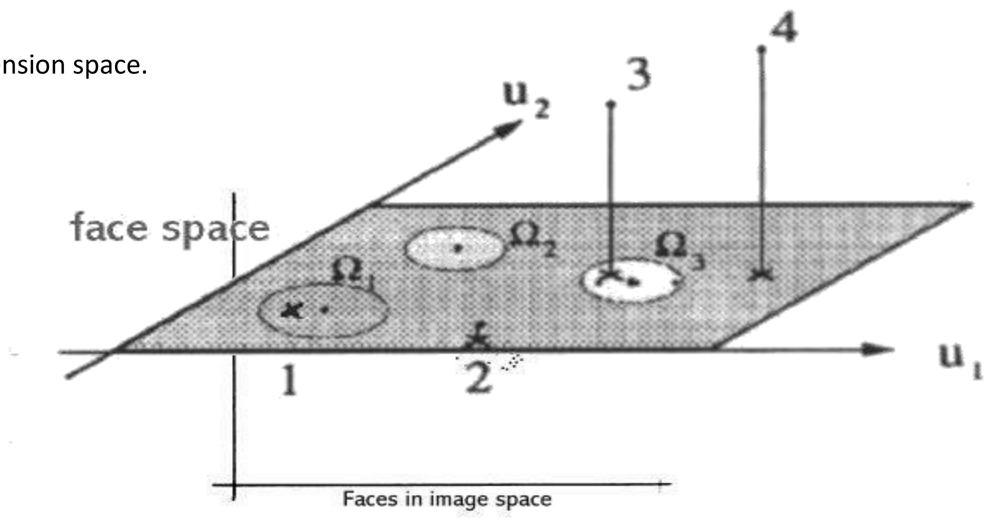


Figure 6.6.0 A 7x7 face image transformed into a 49 dimension vector

This vector can also be regarded as a point in 10000 dimension space. Therefore, all the images of subjects' whose faces are to be recognized can be regarded as points in 10000 dimension space. Face recognition using these images is doomed to failure because all human face images are quite similar to one another so all associated vectors are very close to each other in the 10000-dimension space.



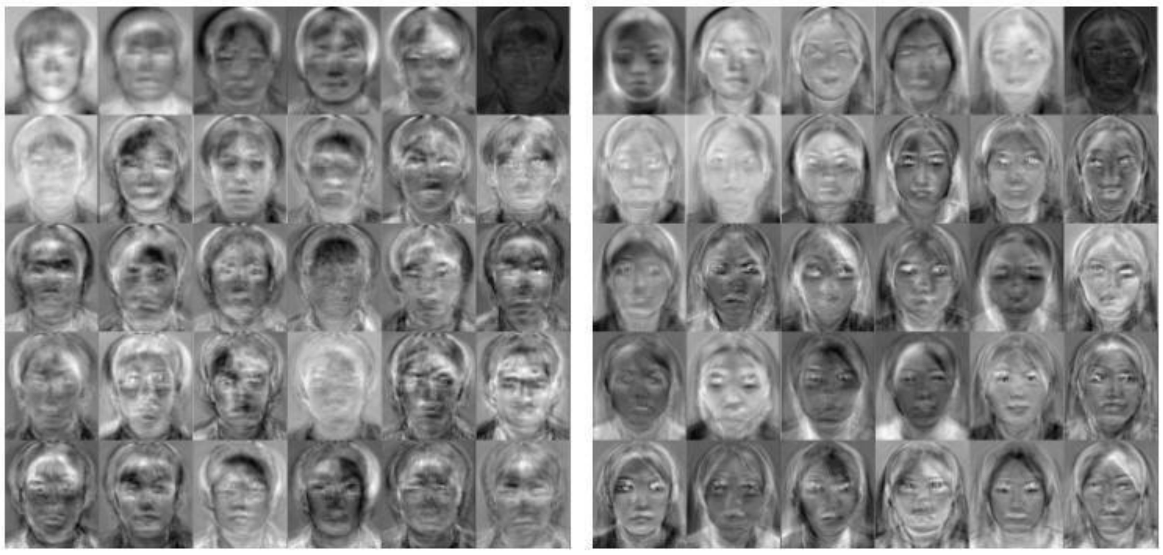


Fig 6.6.1 Eigenfaces

The transformation of a face from image space (I) to face space (f) involves just a simple matrix multiplication. If the average face image is A and U contains the (previously calculated) eigenfaces,

$$f = U * (I - A)$$

This is done to all the face images in the face database (database with known faces) and to the image (face of the subject) which must be recognized. The possible results when projecting a face into face space are given in the following figure.

.

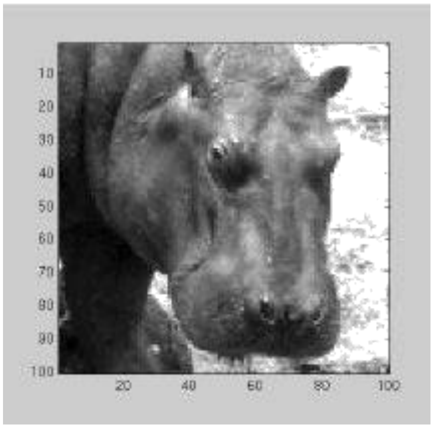
There are four possibilities:

1. Projected image is a face and is transformed near a face in the face database
2. Projected image is a face and is not transformed near a face in the face database
3. Projected image is not a face and is transformed near a face in the face database
4. Projected image is not a face and is not transformed near a face in the face database

While it is possible to find the closest known face to the transformed image face by calculating the Euclidean distance to the other vectors, how does one know whether the image that is being transformed actually contains a face? Since PCA is a many-to-one transform, several vectors in the image space (images) will map to a point in face space (the problem is that even non-face images may transform near a known face image's faces space vector). Turk and Pentland (1991a), described a simple way of checking whether an image is actually of a face. This is by transforming an image into face space and then transforming it back (reconstructing) into image space. Using the previous notation,

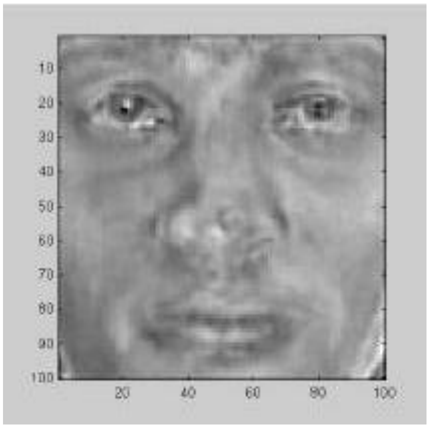
$$I' = U^T * U * (I - A$$

With these calculations it is possible to verify that an image is of a face and recognise that face. O'Toole et al. (1993) did some interesting work on the importance of eigen faces with large and small eigenvalues. They showed that the eigen vectors with larger eigenvalues convey information relative to the basic shape and structure of the faces. This kind of information is most useful in categorising faces according to sex, race etc. Eigen vectors with smaller eigenvalues tend to capture information that is specific to single or small subsets of learned faces and are useful for distinguishing a particular face from any other face. Turk and Pentland (1991a) showed that about 40 eigen faces were sufficient for a very good description of human faces since the reconstructed image have only about 2% RMS. pixel-by-pixel errors.



0.8235

0.0661



- 0.8786

-0.4727

-0.0646

0.6642

-0.4840

-0.4501 -0.2506

0.1591

0.3359

0.0048

0.0745

………..

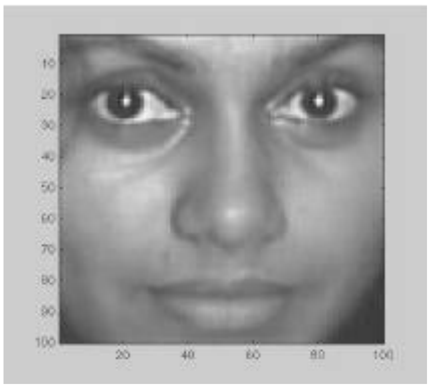
Hippo in image space Hippo in face space Reconstructed hippo in image space

0.7253

-0.0392 0.2896

-0.1725

-0.2642



- 0.0014

- 0.0814

- 0.0054

-0.0623 -

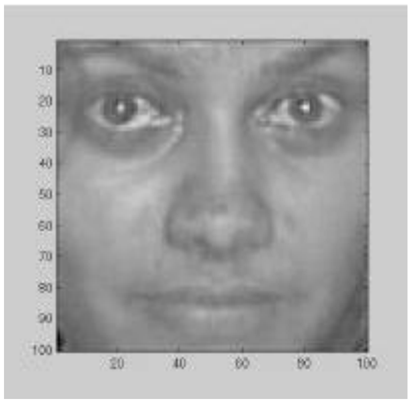
0.0965

- 0.0879

0.0745

-0.0261

…………



Face in image space Face in face space Reconstructed face in image space

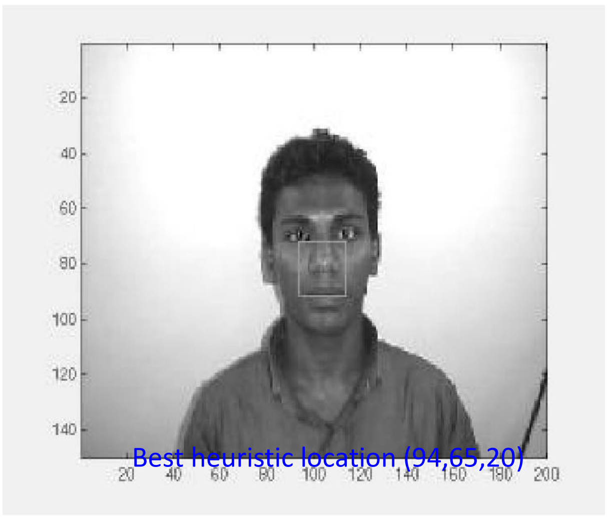
Figure 6.6.3 Images and there reconstruction. The Euclidean distance between a face image and its reconstruction will be lower than that of a non-face image

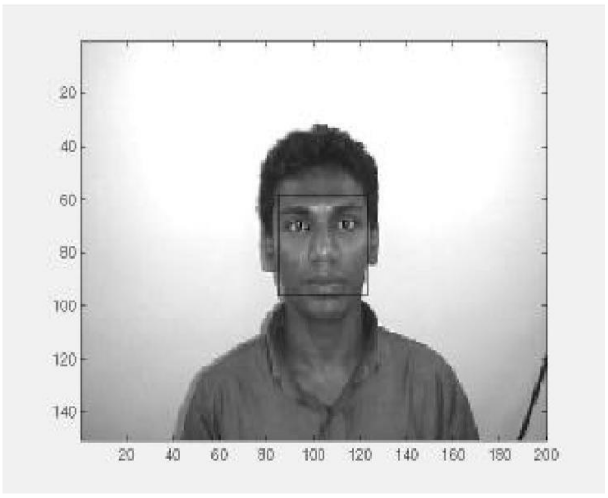
IMPROVING FACE DETECTION USING RECONSTRUCTIN

Reconstruction cannot be used as a means of face detection in images in near real-time since it would involve resizing the face detection *window* area and large matrix multiplication, both of which are computationally expensive. However, reconstruction can be used to verify whether potential face locations identified by the deformable template algorithm actually contain a face. If the reconstructed image differs greatly from the face detection *window* then the *window* probably does not contain a face. Instead of just identifying a single potential face location, the face detection algorithm can be modified to output many high 'faceness' locations which can be verified using reconstruction. This is especially useful because occasionally the best 'faceness' location found by the deformable template algorithm may not contain the ideal frontal view face pixel area.

Output from Face detection system

Heuristic	x	y	width
978	74	31	60





1872	74	33	60
1994	75	32	58
2125	76	32	56
2418	76	34	56
2389	79	32	50
2388	80	33	48
2622	81	33	46
2732	82	32	44
2936	84	33	40
2822	85	58	38
2804	86	60	36
2903	86	62	36
3311	89	62	30
3373	91	63	26
3260	92	64	24
3305	93	64	22
3393	94	65	20

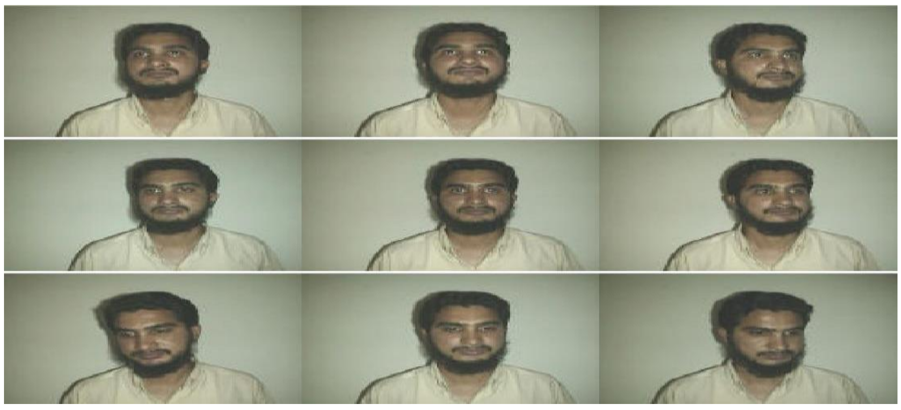
potential face locations that have been identified by the face detection system (the best face locations it found on its search) are checked whether they contain a face. If the threshold level (maximum difference between reconstruction and original for the original to be a face) is set correctly this will be an efficient way to detect a face. The deformable template algorithm is fast and can reduce the search space of potential face locations to a handful of positions. These are then checked using reconstruction. The number of locations found by the face detection system can be changed by getting it to output, not just the best face locations it has found so far but any location, which has a 'faceness' value, which for example is, at least 0.9 times the best heuristic value that has been found so far. Then there will be many more potential face locations to be checked using reconstruction. This and similar speed versus accuracy trade-off decisions have to be made keeping in mind the platform on which the system is implemented.

Similarly, instead of using reconstruction to check the face detection system's output, the output's correlation with the average face can be checked. The segmented areas with a high correlation probably contains a face. Once again a threshold value will have to be established to classify faces from non-faces. Similar to reconstruction, resizing the segmented area and calculating its correlation with the average face is far too expensive to be used alone for face detection but is suitable for verifying the output of the face detection system.

POSE INVARIANT FACE RECOGNITION

Extending the frontal view face recognition system to a pose-invariant recognition system is quite simple if one of the proposed specifications of the face recognition system is relaxed. Successful pose-invariant recognition will be possible if many images of a known individual are in the face database. Nine images from each known individual can be taken as shown below. Then if an image of the same individual is submitted within a 30^{O} angle from the frontal view he or she can be identified.

Nine images in face database from a single known individual



Unknown image from same individual to be identified

Fig: 6.8 Pose invariant face recognition.

Pose invariant face recognition highlights the generalisation ability of PCA. For example, when an individual's frontal view and 30^{O} left view known, even the individual's 15^{O} left view can be recognized

CONCLUSION

The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results confirm that the choices made by the researcher were reliable.The system with manual face detection and automatic face recognition did not have a recognition accuracy over 90%, due to the limited number of eigenfaces that were used for the PCA transform. This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate.The fully automated frontal view face detection system displayed virtually perfect accuracy and in the researcher's opinion further work need not be conducted in this area.

The fully automated face detection and recognition system was not robust enough to achieve a high recognition accuracy. The only reason for this was the face recognition subsystem did not display even a slight degree of invariance to scale, rotation or shift errors of the segmented face image. This was one of the system requirements identified in section 2.3. However, if some sort of further processing, such as an eye detection technique, was implemented to further normalise the segmented face image, performance will increase to levels comparable to the manual face detection and recognition system. Implementing an eye detection technique would be a minor extension to the implemented system and would not require a great deal of additional research.All other implemented systems displayed commendable results and reflect well on the deformable template and Principal Component Analysis strategies.The most suitable real-world applications for face detection and recognition systems are for mugshot matching and surveillance. There are better techniques such as iris or retina recognition and face recognition using the thermal spectrum for user access and user verification applications since these need a very high degree of accuracy.The real-time automated pose invariant face detection and recognition system proposed in chapter seven would be ideal for crowd surveillance applications. If such a system were widely implemented its potential for locating and tracking suspects for law enforcement agencies is immense.

The implemented fully automated face detection and recognition system (with an eye detection system) could be used for simple surveillance applications such as ATM user security, while the implemented manual face detection and automated recognition system is ideal of mugshot matching. Since controlled conditions are present when mugshots are gathered, the frontal view face recognition scheme should display a recognition accuracy far better than the results, which were obtained in this study, which was conducted under adverse conditions.

Furthermore, many of the test subjects did not present an expressionless, frontal view to the system. They would probably be more compliant when a 6'5" policeman is taking their mugshot! In mugshot matching applications, perfect recognition accuracy or an exact match is not a requirement. If a face recognition system can reduce the number of images that a human operator has to search through for a match from 10000 to even a 100, it would be of incredible practical use in law enforcement.

The automated vision systems implemented in this thesis did not even approach the performance, nor were they as robust as a human's innate face recognition system. However, they give an insight into what the future may hold in computer vision.

REFERANCE

- Adelson, E. H., and Bergen, J. R. (1986) The Extraction of Spatio-Temporal Energy in
- Human and Machine Vision, Proceedings of Workshop on Motion: Representation and
- Analysis (pp. 151-155) Charleston, SC; May 7-9
- AAFPRS(1997). A newsletter from the American Academy of Facial Plastic and Reconstructive Surgery. Third Quarter 1997, Vol. 11, No. 3. Page 3.
- Baron, R. J. (1981). Mechanisms of human facial recognition. International Journal of Man Machine Studies, 15:137-178
- Beymer, D. and Poggio, T. (1995) Face Recognition From One Example View, A.I. Memo No. 1536, C.B.C.L. Paper No. 121. MIT
- Bichsel, M. (1991). Strategies of Robust Objects Recognition for Automatic Identification of Human Faces. PhD thesis, , Eidgenossischen Technischen Hochschule, Zurich.
- Brennan, S. E. (1982) The caricature generator. M.S. Thesis. MIT.
- Brunelli, R. and Poggio, T. (1993), Face Recognition: Features versus Templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(10):1042-1052
- Craw, I., Ellis, H., and Lishman, J.R. (1987). Automatic extraction of face features. Pattern Recognition Letters, 5:183-187, February.
- Deffenbacher K.A., Johanson J., and O'Toole A.J. (1998) Facial ageing, attractiveness, and distinctiveness. Perception. 27(10):1233-1243
- Dunteman, G.H. (1989) Principal Component Analysis. Sage Publications.
- Frank, H. and Althoen, S. (1994). Statistics: Concepts and applications. Cambridge University Press. p.110
- Gauthier, I., Behrmann, M. and Tarr, M. (1999). Can face recognition really be dissociated from object recognition? Journal of Cognitive Neuroscience, in press.

- Goldstein, A.J., Harmon, L.D., and Lesk, A.B. (1971). Identification of human faces. In Proc. IEEE, Vol. 59, page 748
- de Haan, M., Johnson, M.H. and Maurer D. (1998) Recognition of individual faces and average face prototypes by 1- and 3- month-old infants. Centre for Brain and Cognitive
- Development, Department of Psychology, Birkbeck College.
- Hadamard, J. (1923) Lectures on the Cauchy Problem in Linear Partial Differential Equations , Yale University Press

- Haralick, R.M. and Shapiro, L.G.. (1992) Computer and Robot Vision, Volume I. Addison-Wesley
- Haxby, J.V., Ungerleider, L.G., Horwitz, B., Maisog, J.M., Rapoport, S.I., and Grady, C.L. (1996). Face encoding and recognition in the human brain. Proc. Nat. Acad. Sci.

  93: 922 - 927.
- Heisele, B. and Poggio, T. (1999) Face Detection. Artificial Intelligence Laboratory.

  MIT.
- Jang., J., Sun, C., and Mizutani, E. (1997) Neuro-Fuzzy and Soft Computing. Prentice Hall.
- Johnson, R.A., and Wichern, D.W. (1992) Applied Multivariate Statistical Analysis. Prentice Hall. p356-395

APPENDIX

FACE DETECTION:

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

cap = cv2.VideoCapture(0)

while 1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]

        eyes = eye_cascade.detectMultiScale(roi_gray)
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

    cv2.imshow('img',img)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

FACE RECOGNITION

```
# coding: utf-8


# Face Recognition with OpenCV


# To detect faces, I will use the code from my previous article on [face detection](https://www.superdatasc
ience.com/opencv-face-
detection/). So if you have not read it, I encourage you to do so to understand how face detection works an
d its Python coding.


# ### Import Required Modules


# Before starting the actual coding we need to import the required modules for coding. So let's import them
 first.
#
# - **cv2:** is _OpenCV_ module for Python which we will use for face detection and face recognition.
# - **os:** We will use this Python module to read our training directories and file names.
# - **numpy:** We will use this module to convert Python lists to numpy arrays as OpenCV face recogniz
ers accept numpy arrays.


# In[1]:


#import OpenCV module
import cv2
#import os module for reading training data directories and paths
import os
#import numpy to convert python lists to numpy arrays as
#it is needed by OpenCV face recognizers
import numpy as np


# ### Training Data


# The more images used in training the better. Normally a lot of images are used for training a face recogni
zer so that it can learn different looks of the same person, for example with glasses, without glasses, laughi
```

```
ng, sad, happy, crying, with beard, without beard etc. To keep our tutorial simple we are going to use only
12 images for each person.

#

# So our training data consists of total 2 persons with 12 images of each person. All training data is inside
_`training-data`_ folder. _`training-
data`_ folder contains one folder for each person and **each folder is named with format `sLabel (e.g. s1, s
2)` where label is actually the integer label assigned to that person**. For example folder named s1 means
that this folder contains images for person 1. The directory structure tree for training data is as follows:

#

# ```

# training-data

# |-------------- s1

# |               |-- 1.jpg

# |               |-- ...

# |               |-- 12.jpg

# |-------------- s2

# |               |-- 1.jpg

# |               |-- ...

# |               |-- 12.jpg

# ```

#

# The _`test-
data`_ folder contains images that we will use to test our face recognizer after it has been successfully train
ed.


# As OpenCV face recognizer accepts labels as integers so we need to define a mapping between integer la
bels and persons actual names so below I am defining a mapping of persons integer labels and their respect
ive names.

#

# **Note:** As we have not assigned `label 0` to any person so **the mapping for label 0 is empty**.


# In[2]:


#there is no label 0 in our training data so subject name for index/label 0 is empty
```

```
subjects = ["", "Ramiz Raja", "Elvis Presley"]


# ### Prepare training data

# You may be wondering why data preparation, right? Well, OpenCV face recognizer accepts data in a specific format. It accepts two vectors, one vector is of faces of all the persons and the second vector is of integer labels for each face so that when processing a face the face recognizer knows which person that particular face belongs too.
#
# For example, if we had 2 persons and 2 images for each person.
#
# ```
# PERSON-1    PERSON-2
#
# img1        img1
#
# img2        img2
# ```
#
# Then the prepare data step will produce following face and label vectors.
#
# ```
# FACES                       LABELS
#
# person1_img1_face           1
#
# person1_img2_face           1
#
# person2_img1_face           2
#
# person2_img2_face           2
# ```
#
#
# Preparing data step can be further divided into following sub-steps.
#
```

```
# 1. Read all the folder names of subjects/persons provided in training data folder. So for example, in this t
utorial we have folder names: `s1, s2`.

# 2. For each subject, extract label number. **Do you remember that our folders have a special naming con
vention?** Folder names follow the format `sLabel` where `Label` is an integer representing the label we h
ave assigned to that subject. So for example, folder name `s1` means that the subject has label 1, s2 means
subject label is 2 and so on. The label extracted in this step is assigned to each face detected in the next ste
p.

# 3. Read all the images of the subject, detect face from each image.

# 4. Add each face to faces vector with corresponding subject label (extracted in above step) added to label
s vector.

#

# **[There should be a visualization for above steps here]**



# Did you read my last article on [face detection](https://www.superdatascience.com/opencv-face-
detection/)? No? Then you better do so right now because to detect faces, I am going to use the code from
my previous article on [face detection](https://www.superdatascience.com/opencv-face-
detection/). So if you have not read it, I encourage you to do so to understand how face detection works an
d its coding. Below is the same code.



# In[3]:



#function to detect face using OpenCV

def detect_face(img):

    #convert the test image to gray image as opencv face detector expects gray images

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)



    #load OpenCV face detector, I am using LBP which is fast

    #there is also a more accurate but slow Haar classifier

    face_cascade = cv2.CascadeClassifier('opencv-files/lbpcascade_frontalface.xml')



    #let's detect multiscale (some images may be closer to camera than others) images

    #result is a list of faces

    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5);



    #if no faces are detected then return original img
    if (len(faces) == 0):

        return None, None


    #under the assumption that there will be only one face,

    #extract the face area

    (x, y, w, h) = faces[0]


    #return only the face part of the image

    return gray[y:y+w, x:x+h], faces[0]


# I am using OpenCV's **LBP face detector**. On _line 4_, I convert the image to grayscale because most
 operations in OpenCV are performed in gray scale, then on _line 8_ I load LBP face detector using `cv2.C
ascadeClassifier` class. After that on _line 12_ I use `cv2.CascadeClassifier` class' `detectMultiScale` meth
od to detect all the faces in the image. on _line 20_, from detected faces I only pick the first face because in
 one image there will be only one face (under the assumption that there will be only one prominent face). A
s faces returned by `detectMultiScale` method are actually rectangles (x, y, width, height) and not actual fa
ces images so we have to extract face image area from the main image. So on _line 23_ I extract face area f
rom gray image and return both the face image area and face rectangle.
#

# Now you have got a face detector and you know the 4 steps to prepare the data, so are you ready to code
the prepare data step? Yes? So let's do it.



# In[4]:



#this function will read all persons' training images, detect face from each image

#and will return two lists of exactly same size, one list

# of faces and another list of labels for each face

def prepare_training_data(data_folder_path):


    #------STEP-1--------

    #get the directories (one directory for each subject) in data folder

    dirs = os.listdir(data_folder_path)


    #list to hold all subject faces
```

```
faces = []

#list to hold labels for all subjects

labels = []


#let's go through each directory and read images within it

for dir_name in dirs:


    #our subject directories start with letter 's' so

    #ignore any non-relevant directories if any

    if not dir_name.startswith("s"):

        continue;


    #------STEP-2--------

    #extract label number of subject from dir_name

    #format of dir name = slabel

    #, so removing letter 's' from dir_name will give us label

    label = int(dir_name.replace("s", ""))


    #build path of directory containin images for current subject subject

    #sample subject_dir_path = "training-data/s1"

    subject_dir_path = data_folder_path + "/" + dir_name


    #get the images names that are inside the given subject directory

    subject_images_names = os.listdir(subject_dir_path)


    #------STEP-3--------

    #go through each image name, read image,

    #detect face and add face to list of faces

    for image_name in subject_images_names:


        #ignore system files like .DS_Store
        if image_name.startswith("."):
          continue;

        #build image path
        #sample image path = training-data/s1/1.pgm
        image_path = subject_dir_path + "/" + image_name

        #read image
        image = cv2.imread(image_path)

        #display an image window to show the image
        cv2.imshow("Training on image...", cv2.resize(image, (400, 500)))
        cv2.waitKey(100)

        #detect face
        face, rect = detect_face(image)

        #------STEP-4--------
        #for the purpose of this tutorial
        #we will ignore faces that are not detected
        if face is not None:
          #add face to list of faces
          faces.append(face)
          #add label for this face
          labels.append(label)

  cv2.destroyAllWindows()
  cv2.waitKey(1)
  cv2.destroyAllWindows()

  return faces, labels
```

```
# I have defined a function that takes the path, where training subjects' folders are stored, as parameter. Thi
s function follows the same 4 prepare data substeps mentioned above.
#

# **(step-
1)** On _line 8_ I am using `os.listdir` method to read names of all folders stored on path passed to functi
on as parameter. On _line 10-13_ I am defining labels and faces vectors.

#

# **(step-
2)** After that I traverse through all subjects' folder names and from each subject's folder name on _line 2
7_ I am extracting the label information. As folder names follow the `sLabel` naming convention so remov
ing the  letter `s` from folder name will give us the label assigned to that subject.

#

# **(step-
3)** On _line 34_, I read all the images names of of the current subject being traversed and on _line 39-
66_ I traverse those images one by one. On _line 53-
54_ I am using OpenCV's `imshow(window_title, image)` along with OpenCV's `waitKey(interval)` metho
d to display the current image being traveresed. The `waitKey(interval)` method pauses the code flow for t
he given interval (milliseconds), I am using it with 100ms interval so that we can view the image window f
or 100ms. On _line 57_, I detect face from the current image being traversed.

#

# **(step-4)** On _line 62-66_, I add the detected face and label to their respective vectors.



# But a function can't do anything unless we call it on some data that it has to prepare, right? Don't worry, I
 have got data of two beautiful and famous celebrities. I am sure you will recognize them!

#

# ![training-data](visualization/tom-shahrukh.png)

#

# Let's call this function on images of these beautiful celebrities to prepare data for training of our Face Re
cognizer. Below is a simple code to do that.



# In[5]:



#let's first prepare our training data

#data will be in two lists of same size

#one list will contain all the faces

#and other list will contain respective labels for each face
```

```
print("Preparing data...")

faces, labels = prepare_training_data("training-data")

print("Data prepared")


#print total faces and labels

print("Total faces: ", len(faces))

print("Total labels: ", len(labels))


# This was probably the boring part, right? Don't worry, the fun stuff is coming up next. It's time to train o
ur own face recognizer so that once trained it can recognize new faces of the persons it was trained on. Rea
d? Ok then let's train our face recognizer.


# ### Train Face Recognizer


# As we know, OpenCV comes equipped with three face recognizers.

#

# 1. EigenFace Recognizer: This can be created with `cv2.face.createEigenFaceRecognizer()`

# 2. FisherFace Recognizer: This can be created with `cv2.face.createFisherFaceRecognizer()`

# 3. Local Binary Patterns Histogram (LBPH): This can be created with `cv2.face.LBPHFisherFaceRecogn
izer()`

#

# I am going to use LBPH face recognizer but you can use any face recognizer of your choice. No matter w
hich of the OpenCV's face recognizer you use the code will remain the same. You just have to change one l
ine, the face recognizer initialization line given below.


# In[6]:


#create our LBPH face recognizer

face_recognizer = cv2.face.LBPHFaceRecognizer_create()


#or use EigenFaceRecognizer by replacing above line with

#face_recognizer = cv2.face.EigenFaceRecognizer_create()
```

```
#or use FisherFaceRecognizer by replacing above line with

#face_recognizer = cv2.face.FisherFaceRecognizer_create()



# Now that we have initialized our face recognizer and we also have prepared our training data, it's time to train the face recognizer. We will do that by calling the `train(faces-vector, labels-vector)` method of face recognizer.


# In[7]:



#train our face recognizer of our training faces

face_recognizer.train(faces, np.array(labels))



# **Did you notice** that instead of passing `labels` vector directly to face recognizer I am first converting it to **numpy** array? This is because OpenCV expects labels vector to be a `numpy` array.
#

# Still not satisfied? Want to see some action? Next step is the real action, I promise!



# ### Prediction



# Now comes my favorite part, the prediction part. This is where we actually get to see if our algorithm is actually recognizing our trained subjects's faces or not. We will take two test images of our celeberities, detect faces from each of them and then pass those faces to our trained face recognizer to see if it recognizes them.

#

# Below are some utility functions that we will use for drawing bounding box (rectangle) around face and putting celeberity name near the face bounding box.



# In[8]:



#function to draw rectangle on image

#according to given (x, y) coordinates and

#given width and heigh

def draw_rectangle(img, rect):

    (x, y, w, h) = rect
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)


#function to draw text on give image starting from

#passed (x, y) coordinates.

def draw_text(img, text, x, y):

    cv2.putText(img, text, (x, y), cv2.FONT_HERSHEY_PLAIN, 1.5, (0, 255, 0), 2)


# First function `draw_rectangle` draws a rectangle on image based on passed rectangle coordinates. It uses
 OpenCV's built in function `cv2.rectangle(img, topLeftPoint, bottomRightPoint, rgbColor, lineWidth)` to
draw rectangle. We will use it to draw a rectangle around the face detected in test image.
#

# Second function `draw_text` uses OpenCV's built in function `cv2.putText(img, text, startPoint, font, fon
tSize, rgbColor, lineWidth)` to draw text on image.

#

# Now that we have the drawing functions, we just need to call the face recognizer's `predict(face)` method
 to test our face recognizer on test images. Following function does the prediction for us.


# In[9]:


#this function recognizes the person in image passed

#and draws a rectangle around detected face with name of the

#subject

def predict(test_img):

    #make a copy of the image as we don't want to chang original image

    img = test_img.copy()

    #detect face from the image

    face, rect = detect_face(img)


    #predict the image using our face recognizer

    label, confidence = face_recognizer.predict(face)

    #get name of respective label returned by face recognizer

    label_text = subjects[label]
    #draw a rectangle around face detected

    draw_rectangle(img, rect)

    #draw name of predicted person

    draw_text(img, label_text, rect[0], rect[1]-5)


    return img


# Now that we have the prediction function well defined, next step is to actually call this function on our te
st images and display those test images to see if our face recognizer correctly recognized them. So let's do i
t. This is what we have been waiting for.


# In[10]:


print("Predicting images...")


#load test images

test_img1 = cv2.imread("test-data/test1.jpg")

test_img2 = cv2.imread("test-data/test2.jpg")


#perform a prediction

predicted_img1 = predict(test_img1)

predicted_img2 = predict(test_img2)

print("Prediction complete")


#display both images

cv2.imshow(subjects[1], cv2.resize(predicted_img1, (400, 500)))

cv2.imshow(subjects[2], cv2.resize(predicted_img2, (400, 500)))

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.waitKey(1)

cv2.destroyAllWindows()
```

Using Frontline Xml. file