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 RECOGNITION

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.

1.1.1 Recognition algorithms can be divided into two main approaches:

1. **Geometric:** Is based on the 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.
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.

1.1.2 Popular recognition algorithms include:

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

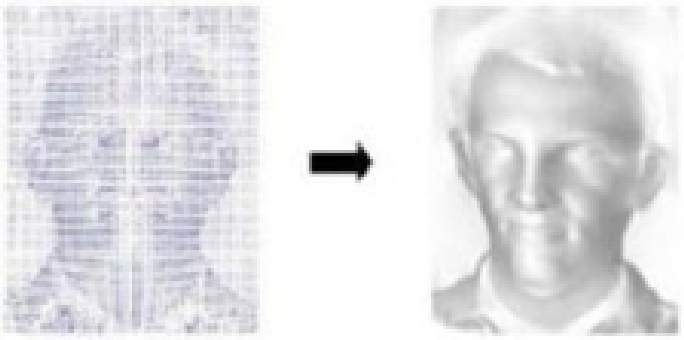


Fig. 1.1: Photometric stereo image

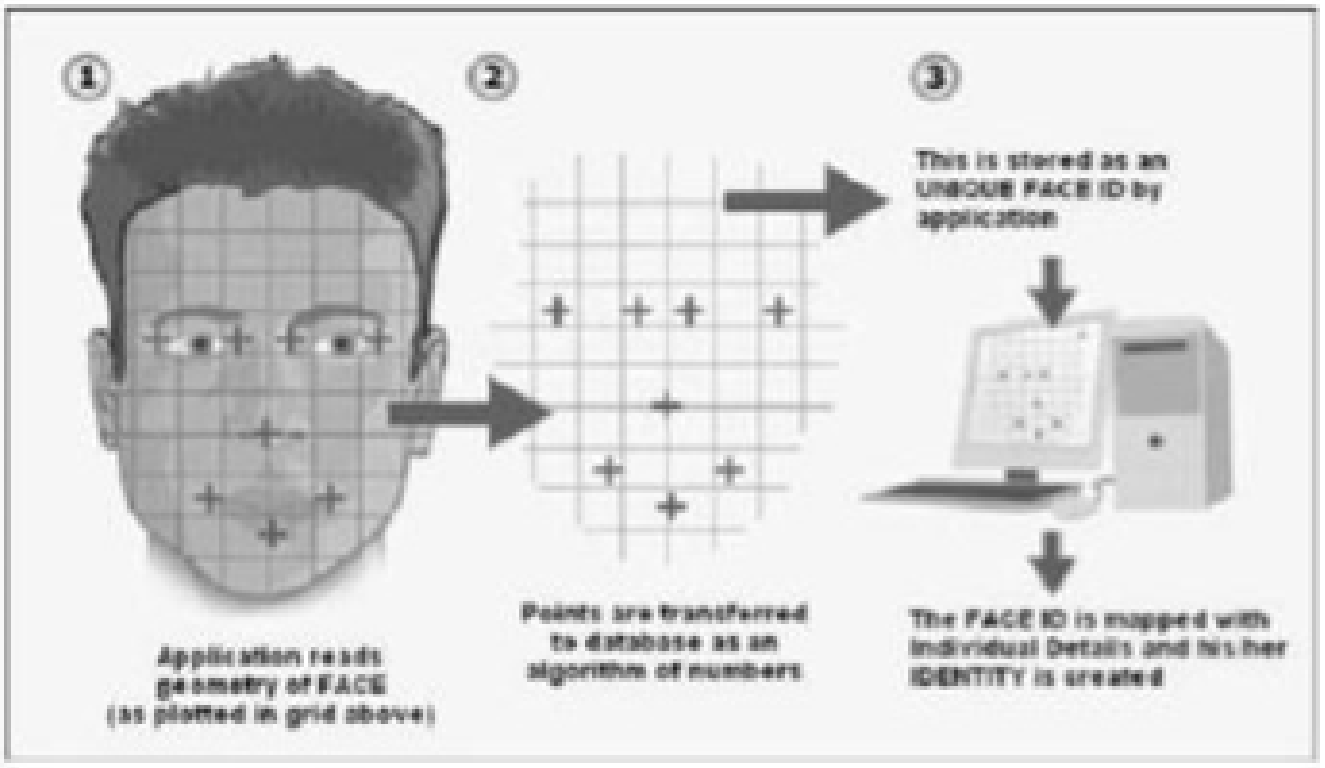


Fig. 1.2: Geometric Facial Recognition

1.2 FACE DETECTION

Face detection involves separating image windows into two classes; one containing faces (turning the background (clutter). It is difficult because

although commonalities exist between faces, they can vary considerably in terms of age, skin color and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

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 non-faces 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 been done on:- Position Scale Orientation Illumination.

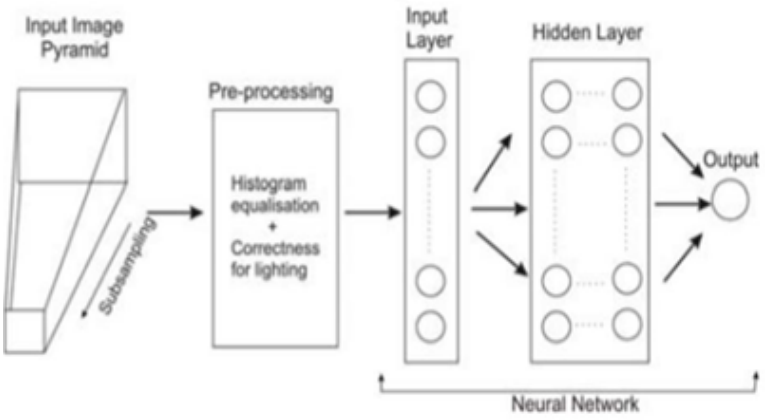


Fig. 1.3: Face Detection algorithm

1.3 DEEP LEARNING

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts.

Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains (especially human brains), which make them incompatible with neuroscience evidence.

Most modern deep learning models are based on an artificial neural network, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines.

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw

input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own. (Of course, this does not completely obviate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.)

The "deep" in "deep learning" refers to the number of layers through which the data is transformed. More precisely, deep learning systems have a substantial credit assignment path (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feedforward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized). For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited. No universally agreed upon threshold of depth divides shallow learning from deep learning, but most researchers agree that deep learning involves CAP depth > 2. CAP of depth 2 has been shown to be a universal approximator in the sense that it can emulate any function. Beyond that more layers do not add to the function approximator ability of the network. Deep models (CAP > 2) are able to extract better features than shallow models and hence, extra layers help in learning features.

Deep learning architectures are often constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance.

For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures that remove redundancy in representation.

Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabeled data is more abundant than labeled data. Examples of deep structures that can be trained in an unsupervised manner are neural history compressors and deep belief networks.

Deep neural networks are generally interpreted in terms of the universal approximation theorem or probabilistic inference.

The classic universal approximation theorem concerns the capacity of feedforward neural networks with a single hidden layer of finite size to approximate continuous functions. In 1989, the first proof was published by George Cybenko for sigmoid activation functions and was generalized to feed-forward multi-layer architectures in 1991 by Kurt Hornik.

The universal approximation theorem for deep neural networks concerns the capacity of networks with bounded width but the depth is allowed to grow. Lu et al. proved that if the width of a deep neural network with ReLU activation is strictly larger than the input dimension, then the network can approximate any Lebesgue integrable function; If the width is smaller or equal to the input dimension, then deep neural network is not a universal approximator.

The probabilistic interpretation derives from the field of machine learning. It features inference, as well as the optimization concepts of training and testing, related to fitting and generalization, respectively. More specifically, the probabilistic interpretation considers the activation nonlinearity as a cumulative distribution function. The probabilistic interpretation led to the introduction of dropout as regularizer in neural networks. The probabilistic interpretation was introduced by researchers including Hopfield, Widrow and Narendra and popularized in surveys such as the one by Bishop.

1.4 CONVOLUTIONAL NEURAL NETWORK

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, RELU layer i.e. activation function, pooling layers, fully connected layers and normalization layers.

Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution (although cross-correlation is a related operation). This only has significance for the indices in the matrix, and thus which weights are placed at which index.

Convolutional: Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.

Each convolutional neural processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of

free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

Pooling: Convolutional networks may include local or global pooling layers. Pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2. Global pooling acts on all the neurons of the convolutional layer. In addition, pooling may compute a max or an average. Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer.

Fully connected: Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

Receptive Field: In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically, the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

Weights: Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous

layer. The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a filter and represent some feature of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

1.5 EMOTION RECOGNITION

Emotion recognition is the process of identifying human emotion, most typically from facial expressions as well as from verbal expressions. This is both something that humans do automatically but computational methodologies have also been developed.

Knowledge-based techniques: Knowledge-based techniques (sometimes referred to as lexicon-based techniques), utilize domain knowledge and the semantic and syntactic characteristics of language in order to detect certain emotion types. In this approach, it is common to use knowledge-based resources during the emotion classification process such as WordNet, SenticNet, ConceptNet, and EmotiNet, to name a few. One of the advantages of this approach is the accessibility and economy brought about by the large availability of such knowledge-based resources. A limitation of this technique on the other hand, is its inability to handle concept nuances and complex linguistic rules.

Knowledge-based techniques can be mainly classified into two categories: dictionary-based and corpus-based approaches. Dictionary-based approaches find opinion or emotion seed words in a dictionary and search for their synonyms and antonyms to expand the initial list of opinions or emotions. Corpus-based approaches on the other hand, start with a seed list of opinion or emotion words, and expand the database by finding other words with context-specific characteristics in a large corpus. While corpus-based approaches take into account context, their performance still varies in different domains since a word in one domain can have a different orientation in another domain.

Statistical methods: Statistical methods commonly involve the use of different supervised machine learning algorithms in which a large set of annotated data is fed into the algorithms for the system to learn and predict the appropriate emotion types. This approach normally involves two sets of data: the training set and the testing set, where the former is used to learn the attributes of the data, while the latter is used to validate the performance of the machine learning algorithm. Machine learning algorithms generally provide more reasonable classification accuracy compared to other approaches, but one of the challenges in achieving good results in the classification process, is the need to have a sufficiently large training set.

Some of the most commonly used machine learning algorithms include Support Vector Machines (SVM), Naive Bayes, and Maximum Entropy. Deep learning, which is under the unsupervised family of machine learning, is also widely employed in emotion recognition. Well-known deep learning algorithms include different architectures of Artificial Neural Network (ANN) such as Convolutional Neural Network (CNN), Long Short-term Memory (LSTM), and Extreme Learning Machine (ELM). The popularity of deep learning approaches in the domain of emotion recognition may be mainly attributed to its success in related applications such as in computer vision, speech recognition, and Natural Language Processing (NLP).

SOFTWARE REQUIREMENTS SPECIFICATION

2.1 INTRODUCTION

The main purpose of the use of PCA on face recognition using Eigenfaces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project's face detection system with face recognition is Image processing.

2.1.1 Purpose

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 recognizes a face as individuals.

2.1.2 Scope

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 recognizes 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 patterns, namely, the Eigenface method and Fisherface method. Facial image recognition Eigenface method is based on the reduction of face- dimensional space using Principal Component Analysis (PCA) for facial features.

2.1.3 Overview

Emotion recognition system recognizes the expression of human depending on their mood like happy, sad, angry, surprise, fear, disgust etc.

2.1.4 Definitions, Acronyms and Abbreviations

Table 2.1: Terms used in SRS

Term	**Definition**
Active User	A valid and authenticated user.
Database	Collection of all the information monitored by this system.
Elastic Engine	The master engine allows extraction of resumes.
Field	A cell within a form.
User Collection	The collection contains information of registered users.
Form	The area where text pertaining to login and register is entered.
Keywords	The specific words the user wants to extract the resumes on the basis of.
Result	The list of resumes which have been extracted from given input.
Input	The resume files selected to sort through.
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document.
Stakeholder	Any person with an interest in the project who is not a developer.
User	Any person using the software to extract required resumes.

2.2 SYSTEM INTERFACE

1. Face Recognition Based Attendance System will operate with OpenCV

2. Face Recognition Based Attendance System uses a webcam for face detection and face recognition.

3. Face Recognition Based Attendance System shall permit user access data from database.

4. Operating System: Linux.

2.3 HARDWARE INTERFACE

Hardware Requirements: 256(minimum)/512(recommended) MB RAM

Hard disc- 10GB depending upon the requirement to store data minimum of 25GB Database

OS – Linux

HDD – Min 10 GB, Recommended 25 GB

RAM – Min 1 GB, Recommended 4 GB

Processor - Pentium Dual Xenon Processor

Camera-WebCam

Application

OS – Linux

Software–OpenCV, Python 3.6+

HDD – Min 5 GB, Recommended 10 GB

RAM – Min 2 GB, Recommended 4 GB

2.4 SOFTWARE INTERFACE

1. Software requirements: SQLite3, OpenCV, Python 3.6+

2. Languages used: - Python, Python libraries, OpenCV.

A requirements specification, which is a document providing a detailed description of the issue or project, and the requirements identified to either fix it, or make it happen.

2.5 PRODUCT FEATURES

A. 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.

While initially a form of computer application, it has seen wider uses in recent times on mobile platforms and in other forms of technology, such as robotics. It is typically used as access control in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. Although the accuracy of facial recognition systems as a biometric technology is lower than iris recognition and fingerprint recognition, it is widely adopted due to its contactless and non-invasive process. Recently, it has also become popular as a commercial identification and marketing tool. Other applications include advanced human-computer interaction, video surveillance, automatic indexing of images, and video database, among others.

A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from a given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape.

B. Face Detection

Face detection involves separating image windows into two classes; one

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

C. Emotion Detection

Emotion recognition systems based on facial gesture enable real-time analysis, tagging, and inference of cognitive affective states from a video recording of the face. It is assumed that facial expressions are triggered for a period of time when an emotion is experienced and so emotion detection can be achieved by detecting the facial expression related to it. From each facial expression, a set of facial action units is extracted, each facial action unit identifying an independent motion of the face. Movements in facial muscles are perceived as changes in the position of the eyes, nose, and mouth. Computer systems implement this approach by capturing images of the user's facial expressions and head movements. Those systems detect changes in the position of the eyes, nose, and mouth as changes in the position of dots in a coordinate system. Then, by analyzing those changes, the occurrence of a facial action unit can be determined. The Facial Action Coding System (FACS) documents 46 possible facial action units (Ekman et al., 1980). For instance, happiness is associated with the occurrence of action units 6 and 12 (cheek riser and lip corner puller), and sadness is associated with the occurrence of action units 1, 4, and 15 (inner brow raiser, brow lowerer, and lip corner depressor).

2.6 SOFTWARE SYSTEM ATTRIBUTES

The requirements in this section specify the required reliability, availability, security and maintainability of the software system.

A. Reliability

- Title: System Reliability
- Description: The reliability of the system.
- Scale: The reliability that the system gives the right result on an operation.
- Meter: Measurements obtained from 1000 searches during testing.
- Must:More than 98% of the searches.
- Plan: More than 99% of the searches.
- Wish: 100% of the searches.

B. Availability

- Title: System Availability
- Scale: The average system availability (not considering network failing).
- Meter: Measurements obtained from 1000 hours of usage during testing.
- Must: More than 98% of the time.
- Plan: More than 99% of the time.
- Wish: 100% of the time.
- Title: Internet Connection
- Description: The application should be connected to the Internet in order for the application to communicate with the database.

C. Extendibility

- Title: Application extensibility
- Description: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions

D. Testability

- Title: Application testability
- Description: Test environments should be built for the application to allow testing of the applications different functions.

E. Portability

- Title: Application portability
- Description: The application should be portable with OS and should be able to run smoothly with minimum requirements.

F. Security

- The computer that runs the program will have its own security. Only the System Admin will log in to the system with his/her username and password. The person whose face and the iris are recognized will access to view the output. The PC that runs the program will have its very own security. Just the System Admin will sign in to the framework with his/her username and secret key. The individual whose face and the iris are perceived will access to see the yield.

2.7 NON FUNCTIONAL REQUIREMENTS

In non-functional requirement limits are provided, which means restriction is attached with the requirement and one has to fulfill or satisfy that limit.

- **Time:** The project has to be completed within two months.
- **Input:** Facial expression of human.
- **General Requirements:**
    - Physical Environment: Physical environment requirements such as where is the equipment to function located, are there any environmental restrictions like temperature, humidity, magnetic interference, etc are to be gathered prior to the development phase of the system. No special physical requirements are needed in our project.
    - Interface:Interface requirements are such as is the input coming from one or more systems, is output going to one or more systems, is there any prescribed medium that the data must use should be gathered. In our project the input comes from a webcam.
    - User and Human Factors: User and Human factors consist of requirements as who will use the system, will there be different types of users, what is the skill levels of each type of user, what kind of training is required for each user, and how easy will it be for a user to

understand and use the system are required. In this project no special training is required to use the project.

- Documentation: Documentation requirements like how much documentation is required, should it be online or book format or both are required.
- Data: Data requirements like what should be the format of both the input and output, how accurate they must be, should any data be retained for any period of time should be known.
- Error handling: Application shall handle expected and unexpected errors in ways that prevent loss in information.

2.8 RISK ANALYSIS

These steps are performed in risk analysis for designing the system because:
The future of the system is our concern. We identifying what risks might create problem in the life of the system. We also identified that what change in the user requirements, technologies, hardware and all other entries connected to the system will affects the system. Risk analysis is a technique used to identify and assess factors that may jeopardize the success of a project or achieving a goal.

This technique also helps to define preventive measures to reduce the probability of these factors from occurring and identify counter measures to successfully deal with these constraints when they develop to avert possible negative effects on the competitiveness of the company.

One of the more popular methods to perform a risk analysis in the computer field is called facilitated risk analysis process (FRAP). FRAP analyzes one system, application or segment of business processes at a time.

2.8.1 Risk Identification

We were able to identifying the risk under the following categories:

- Project risk
- Technical risk
- Business risk

Following list was identifying under the categories mentioned above:

- Enough number of people was available, as estimated, to complete the system.
- All staff involved in the system was not fully trained on the platform to be used for development. We also had to study various things about the platform and the system.
- The staff involved in the system was committed for the entire decision of the project. The entire member worked full time on the system.

2.8.2 Probability of Risk

The probability for the project risks such as schedule, resources, customer, requirement problems and their impact on the system was negligible. There was a risk on the technical grounds because the system was developed with a new technology hence the experience on the tools was taking which faced the management to think whether the choice made was right or wrong. But a survey done on the use of a new platform gave us the confidence of continuing on this decision. As we know system design is a “how to” approach to the creation of a new system. This important phase is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study.

SYSTEM DESIGN

3.1 Client Server Architecture

Client/server architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or internet connection. This system shares computing resources.

Client/server architecture is also known as a networking computing model or client/server network because all the requests and services are delivered over a network.

Advantages

- Centralization of control: access, resources and integrity of the data are controlled by the dedicated server so that a program or unauthorized client cannot damage the system. This centralization also facilitates the task of updating data or other resources (better than the networks P2P).
- Scalability: You can increase the capacity of clients and servers separately. Any element can be increased (or enhanced) at any time, or you can add new nodes to the network (clients or servers).
- Easy maintenance: distribute the roles and responsibilities to several standalone computers, you can replace, repair, upgrade, or even move a server, while customers will not be affected by that change (or minimally affected). This independence of the changes is also known as encapsulation.

There are technologies sufficiently developed, designed for the paradigm of client/server to ensure security in transactions, interface friendliness, and ease of use.

3.2 Software Architectural Design

Our system follows the three tier architecture. First tier consists of GUI, Processing block and the Database.

GUI:

The GUI (Graphical User Interface) in our project deals with the interface for the user where the user will login and submit his resume in any format (pdf, doc, docx,

etc.) and social profiles links. The GUI provides a platform for the user to communicate with the database. It acts as a connector as well as communicator which connects the database and helps in transfer of data between the GUI and the database.

Processing block:

Processing block is the block where the actual processing of our project is done. This block connects the GUI to the database i.e. it acts as a connector as well as communicator which connects the database and helps in transfer of data between the GUI and the database. Its main function is to take input from resumes and social profiles of the candidate and parse it to store the information and store it in the structured format(json), and database. After storing this information this system will give output using a web application.

Database:

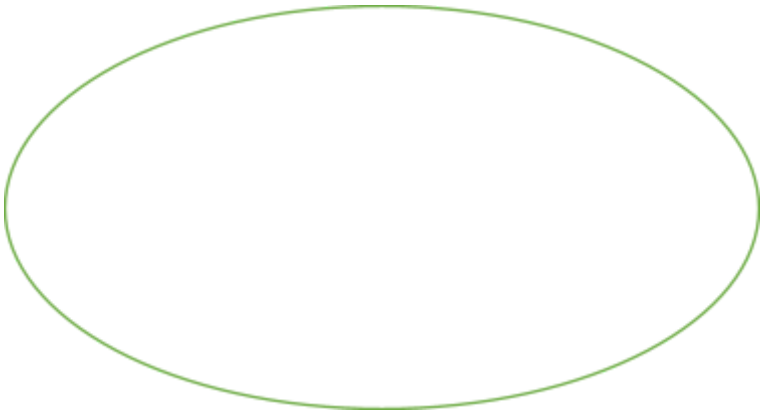
Database tier is the tier used for the storage of data. This tier contains all the data that is needed for the processing of the whole project. The data in this tier is related to the student information gathered from his/her resumes and social profiles.

3.3 Flowchart

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence. They can range from simple, hand-drawn charts to comprehensive computer-drawn diagrams depicting multiple steps and routes. If we consider all the various forms of flowcharts, they are one of the most common diagrams on the planet, used by both technical and non-technical people in numerous fields. Flowcharts are sometimes called by more specialized names such as Process Flowchart, Process Map, Functional Flowchart, Business Process Mapping, Business Process Modeling and Notation (BPMN), or Process Flow Diagram (PFD). They are

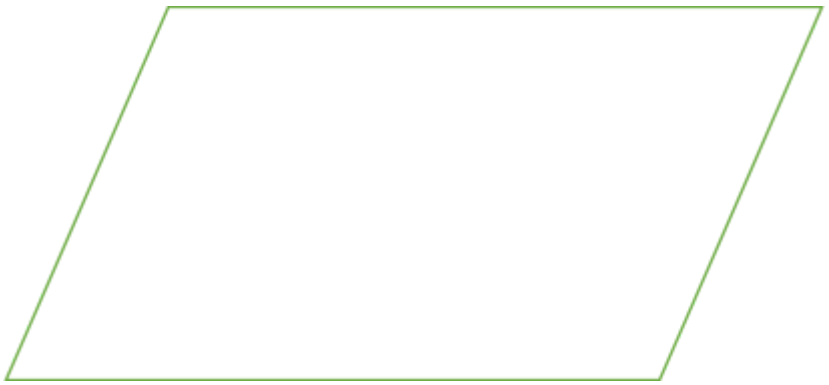
related to other popular diagrams, such as Data Flow Diagrams (DFDs) and Unified Modeling Language (UML) Activity Diagrams.

3.3.1 Flowchart Symbols

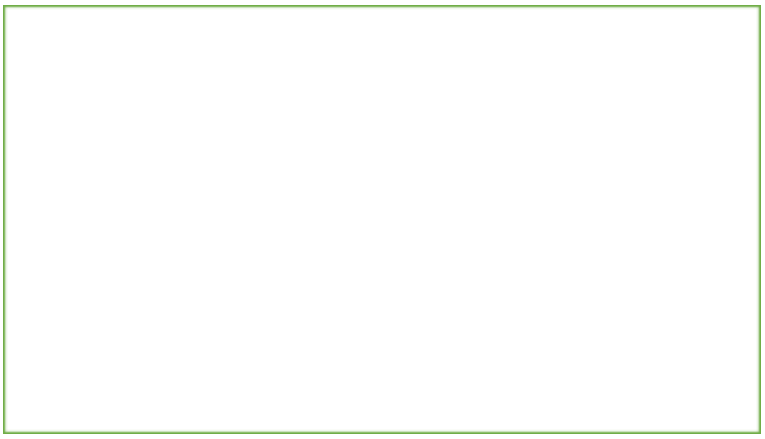


Start/End

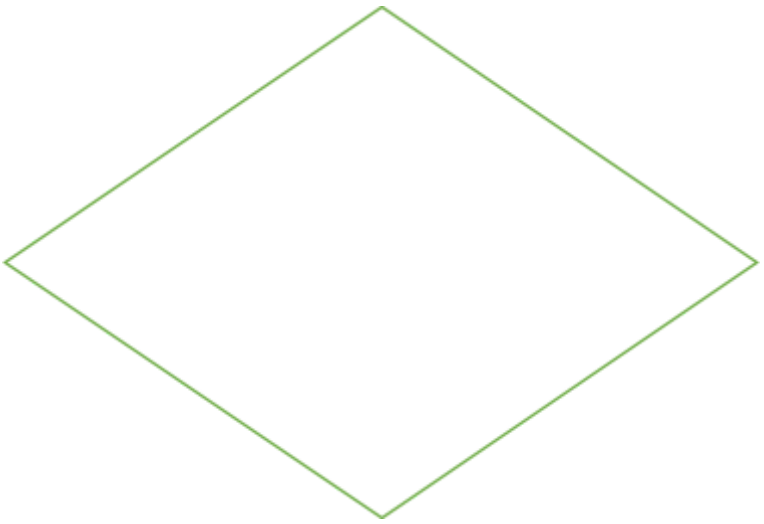
Arrows



Input/Output



Process



Decision

Figure 3.1: Flowchart Symbols

3.4 Entity Relationship Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.
ER diagrams are used to sketch out the design of a database.

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. It does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities. An ER model can also be expressed in a verbal form, for example: one building may be divided into zero or more apartments, but one apartment can only be located in one building.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity–relationship models.

3.4.1 Common Entity Relationship Diagram Symbols

An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

- **Entities**, which are represented by rectangles. An entity is an object or concept about which you want to store information.

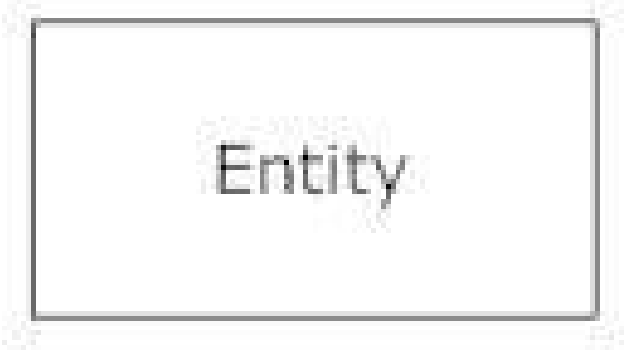


Figure 3.2: Entity

- **Actions**, which are represented by diamond shapes, show how two entities share information in the database.

Figure 3.3: Relationship

- **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.

Figure 3.4: Attribute

- **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.
- **Cardinality** specifies the maximum number of relationships and **Ordinality** specifies the absolute minimum number of relationships.

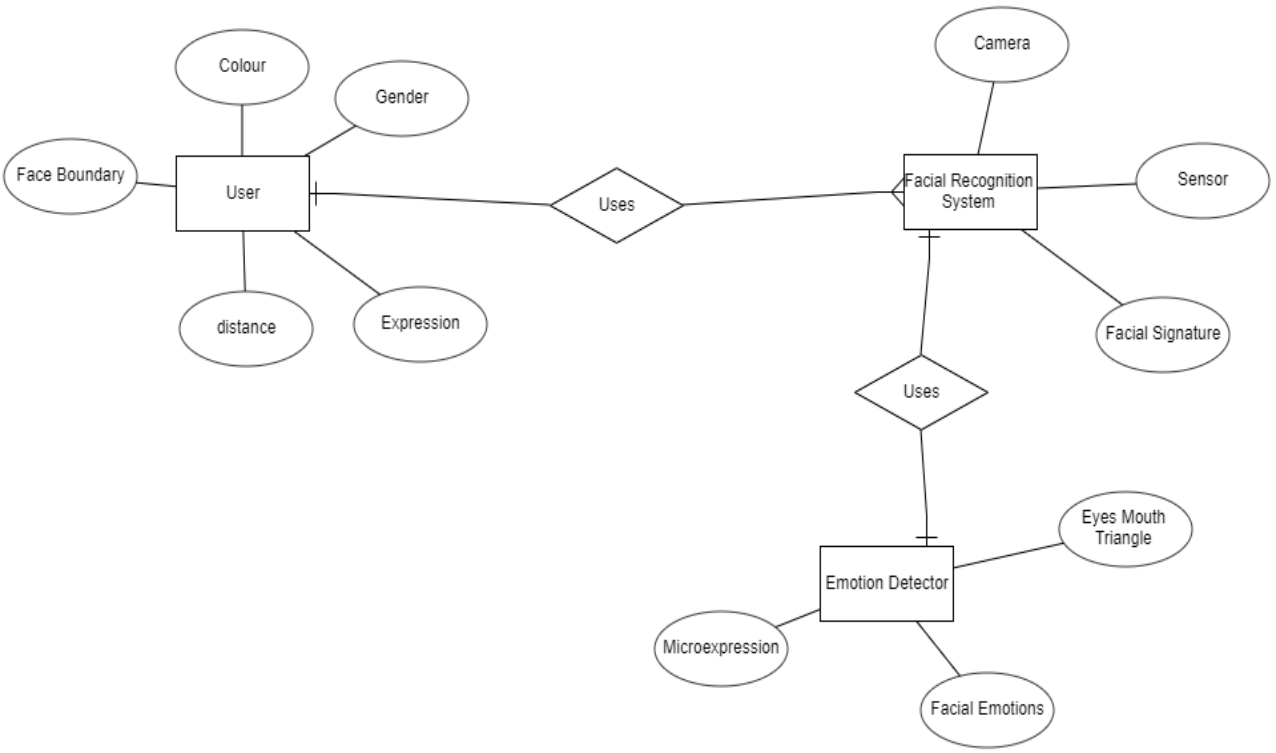


Figure 3.5 ER Diagram of Emotion Recognition

3.5 Data Flow Diagrams

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored. It maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

3.5.1 Data Flow Diagrams Symbols

There are essentially two different types of notations for data flow diagrams (Yourdon & Coad or Gane & Sarson) defining different visual representations for processes, data stores, data flow and external entities.

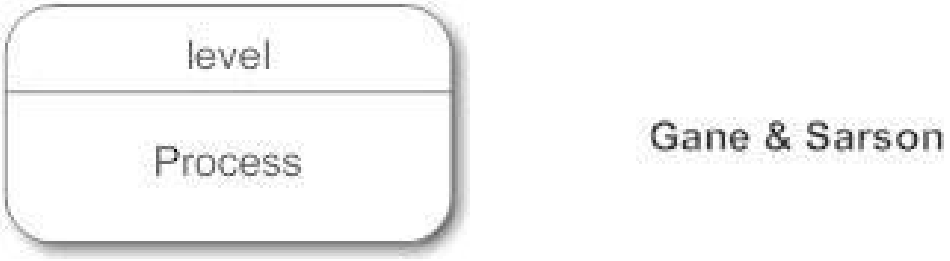


Figure 3.6:
Process Notation

Process Notations: A process transforms incoming data flow into outgoing data flow.

Figure 3.7: Data store Notations

Data store Notations: Data stores are repositories of data in the system. They are sometimes also referred to as files.

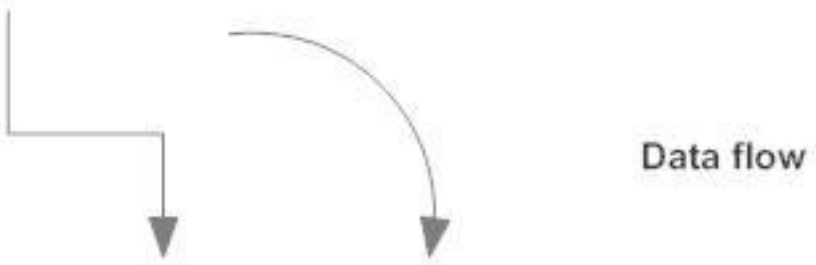


Figure 3.8: Data flow Notations

Data flow Notations: Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.

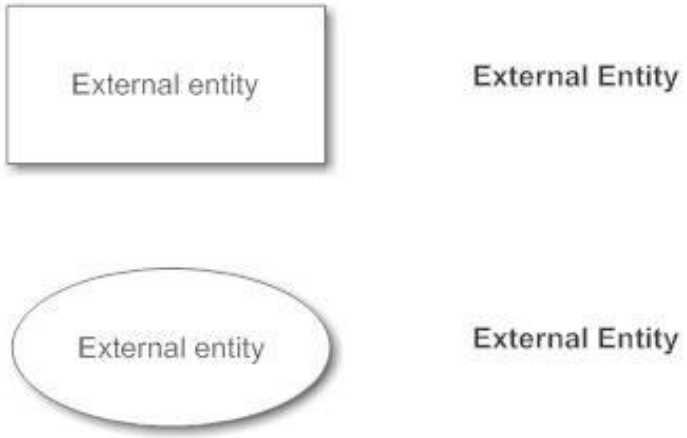


Figure 3.9: External Entity Notations

External Entity Notations: External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system's inputs and outputs.

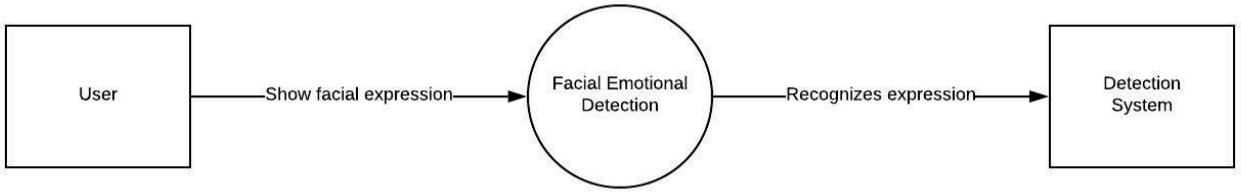


Figure 3.10 Data Flow Diagram Level 0 of Emotion Recognition

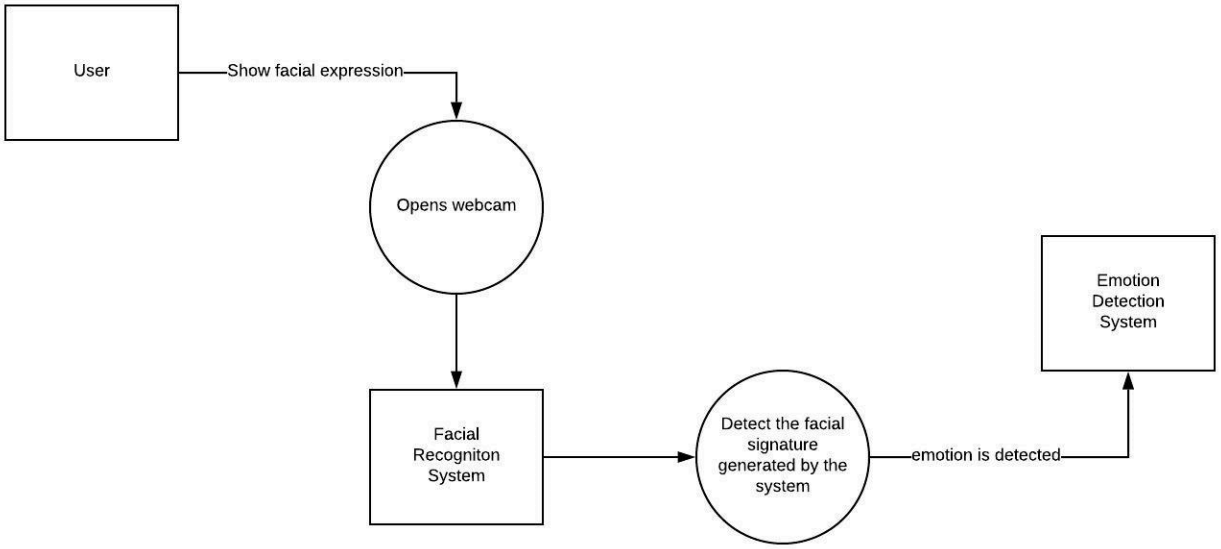


Figure 3.11 Data Flow Diagram Level 1 of Emotion Recognition

3.6 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide a simplified and graphical representation of what the system must actually do.

Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed. Siau and Lee conducted research to determine if there was a valid situation for use case diagrams at all or if they were unnecessary. What was found was that the use case diagrams conveyed the intent of the system in a more simplified manner to stakeholders and that they were "interpreted more completely than class diagrams".

The purpose of the use case diagrams is simply to provide the high level view of the system and convey the requirements in laypeople's terms for the stakeholders. Additional diagrams and documentation can be used to provide a complete functional and technical view of the system.

Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities. They also help identify any internal or external factors that may influence the system and should be taken into consideration. They provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

To understand the dynamics of a system, we need to use different types of diagrams. Use case diagrams are one of them and its specific purpose is to gather system requirements and actors.

Use case diagrams specify the events of a system and their flows. But use case diagrams never describe how they are implemented. Use case diagrams can be imagined as a black box where only the input, output, and the function of the black box is known.

These diagrams are used at a very high level of design. This high level design is refined again and again to get a complete and practical picture of the system. A well-structured use case also describes the pre-condition, post condition, and exceptions. These extra elements are used to make test cases when performing the testing.

In forward engineering, use case diagrams are used to make test cases and in reverse engineering use cases are used to prepare the requirement details from the existing application.

Use case diagrams can be used for −

- Requirement analysis and high level design.
- Model the context of a system.
- Reverse engineering.
- Forward engineering.

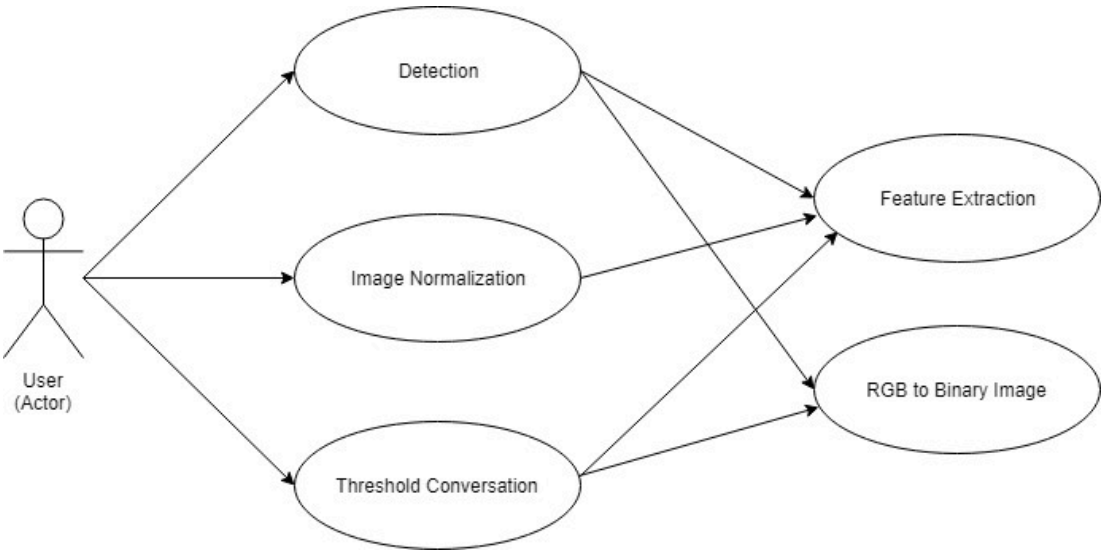


Figure 3.12 Use Case Diagram of Emotion Recognition

TEST PLAN

4.1 Introduction

The Test Plan is designed to describe the scope, approach, resources, and schedule of all testing activities of the project Emotion Detection System. The plan identifies the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the schedule and resources required to complete testing, and the risks associated with the plan. The objective is to guarantee the operation of the Emotion Detection System is in alignment of requirements.

4.1.1 Objective

The objective of testing is as follows:

The objective of testing is as follows:

1. Verify that the solution meets the requirements.
2. Ensure the solution is optimized for the environment.
3. Ensure overall quality.

Meeting Requirement

The first testing objective is to prove that the Emotion Detection System module addresses the problem and satisfies the requirement. Making sure to serve the basic as well as very important needs.

Optimized of Business Environment

The second testing objective is to ensure it works in a real-world environment. This means providing tests that simulate real life conditions and situations.

Ensure Quality

The third goal of our testing is to help ensure a high-level of quality of product. Since software changes become exponentially more expensive as a project advances through its lifecycle, the goal is to identify defects as early as possible.

4.1.2 Features to be tested

Registration

- User is able to register with valid credentials
- Error message is displayed when the user registers with invalid credentials

Login

- User is able to Login with valid credentials and is taken to Home Page
- Error message is displayed when the user registers with invalid credentials

Camera

- User is able to access the camera.
- If the user is using a desktop, a computer accessory, webcam is required.
- The camera should be good enough to recognize the user's facial expression from a distance.

Face Detection

- On opening the application, the user must be able to see his/her face on the screen.
- The sensor must be able to recognize the face of the user.
- The sensors should be able to detect the facial gestures made by the user.

Emotion Detector

- The sensor should be able to detect the emotions of the user like happy, sad, surprise, angry, fear and more.

4.1.3 Features not to be tested

- Data present in the table.
- Comparison of user's expressions with previous inputs.

4.1.4 Testing Approaches

Testing process will be focused on two major portions. First, the face detector, which detects the user. Second is emotion detection which focuses on the emotions of the user.

4.1.5 Testing Methodology

In order to ensure that all testing objectives are met, our testing philosophy is based on industry accepted principles of testing and validation.

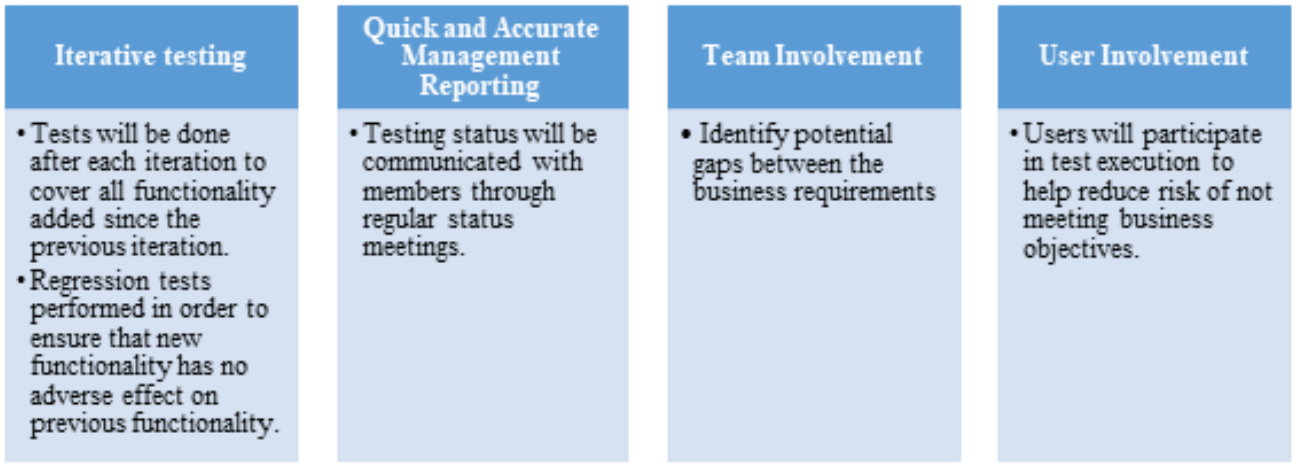


Figure 4.1 Testing Methodology

4.1.6 Test Execution

Test execution involves actually running the tests against the system. Actual results should be confirmed that they match the expected results. Any actual results that do not match expected results will go through defect management processes.

4.2 Testing Scope

Smoke Testing

- Test the major functionalities of bugs.
- If 'must have' functionalities have defects, the testing phase will not start.
- Product would be sent back to the developer.

Integration Testing

- Scope of integration testing is to ensure that specific functionalities work in sync with each other.
- Prepare test cases for all the functionality where there is a linkage between modules.

- Test the flow of data among different modules for Application under test.

System Testing

- The objectives of system tests are to test major business functionality, ensure functional and quality requirements are met.
- Ensure the system supports the use cases that have been defined

User Interface Testing

- Ensure the presence and alignment of elements as defined in requirements.
- Identify the elements of which User Interface testing has to be performed.
- Prepare test cases for the presence and look of elements.
- Execute the test cases designed.

Progression Testing

- Objective should be to validate the functionality according to client requirements.
- Identify progression scenarios.
- Prepare test cases for progression.
- Execute newly created test cases for the added requirement

Regression Testing

- Test that no new defects are introduced after the addition of requirements.
- Identify regression scenarios.
- Prepare test cases.
- Automate regression test cases in the previously present functionalities.

User Acceptance Testing

- Receive sign off from project manager, test lead and development lead.
- Set the pre-production environment.
- Train the user on how to use the application.
- Record the result and responses while the user performs testing

4.3 Testing Entry/Exit Criteria and environment

Smoke Testing

Entry Criteria: Major functionalities are added and working.

Exit Criteria: No Bugs found in testing

Environment: Test

Integration Testing

Entry Criteria: No bugs in smoke testing

Exit Criteria:

1. All the integration test cases have been executed.
2. No critical and Priority P1 & P2 defects are opened.

Environment: Test

System Testing

Entry Criteria: No critical or P1, P2 Priority bug is in open state.

Exit Criteria: No critical or Priority bugs should be in an open state

Environment: Test

User Interface Testing

Entry Criteria:

1. The system should have passed the exit criteria of system testing.
2. No critical or Priority P1, P2 bug in an open state.

Exit Criteria: No critical or Priority bugs should be in an open state

Environment: Test

Regression Testing

Entry Criteria: After progression testing and no P1, P2 open bugs found.

Exit Criteria:

1. All previous bugs are tested.
2. No critical or Priority P1, P2 bug in an open state

Environment: Test

Progression Testing

Entry Criteria:

1. After the exit criteria for the smoke test are met.
2. No open P1, P2 defects open.

Exit Criteria:

1. All test cases are executed.
2. No critical or Priority P1, P2 bug in an open

Environment: Test

User Acceptance Testing

Entry Criteria: Sign off from Team Member.

Exit Criteria:

1. No critical defects open
2. Business process works satisfactorily

Environment: Pre-Production

4.4 Defect management

Testing team recognizes that the bug reporting process is a critical communication tool within the testing process. Without effective communication of bug information and other issues, the development and release process will be negatively impacted.

When logging a Bug, the tester should provide a priority to defect to assist the developer in prioritizing the order in which they review and fix defects. Priority 1 and priority 2 defects must be addressed for acceptance (see definitions below).

Bug Priority: Critical

Priority = 1-Critical

Priority Definition: System down or a critical defect inhibiting the majority of users from performing key tasks *with no workarounds*. These incidents must be resolved prior to implementation.

Example: Form Tool defect preventing submission.

Bug Priority: Very Important

Priority = 2-High

Priority Definition: Important defect that significantly impacts performance of a large number of users, but a workaround exists*;* a critical defect inhibiting a small number of users from performing key tasks with no workarounds. Incident requires resolution.

Example: Round-robin assignment logic defect that causes a manual assignment workaround.

Bug Priority: Important

Priority = 3-Medium

Priority Definition: Enhancement that improves usability for the majority of users or a non-critical defect, with workarounds.

Example: Form does not format properly across less common devices/browsers.

Bug Priority: Nice to Have

Priority = 4-Low

Priority Definition: Enhancement that improves usability for a small set of users or is cosmetic in nature; features that are not critical to the functionality of the solution.

Example: Incorrect capitalization, punctuation or font; tabbing order; size of the text box does not correspond to max text length.

Bug Triage

The Test candidate and Development candidate should all be involved in these triage meetings. The Test Lead will provide required documentation and reports on bugs for all attendees. The purpose of the triage is to determine the type of resolution for each bug and to prioritize and determine a schedule for all "To Be Fixed Bugs'. Development will then assign the bugs to the appropriate person for fixing and report the resolution of each bug back into the Enrich. The Test Lead will be responsible for tracking and reporting on the status of all bug resolutions.

Bug Regression will be a central tenant throughout all testing phases.

All bugs that are resolved as "Fixed, Needs Re-Testing" will be regressed when the testing team is notified of the new drop containing the fixes. When a bug passes regression, it will be considered "Closed, Fixed". If a bug fails regression, the adopters testing team will notify the development team by entering notes into notes. When a Priority 1 bug fails regression, the testing team should also put out an immediate email to development. The Test Lead will be responsible for tracking and reporting.

Fixed, Needs retesting

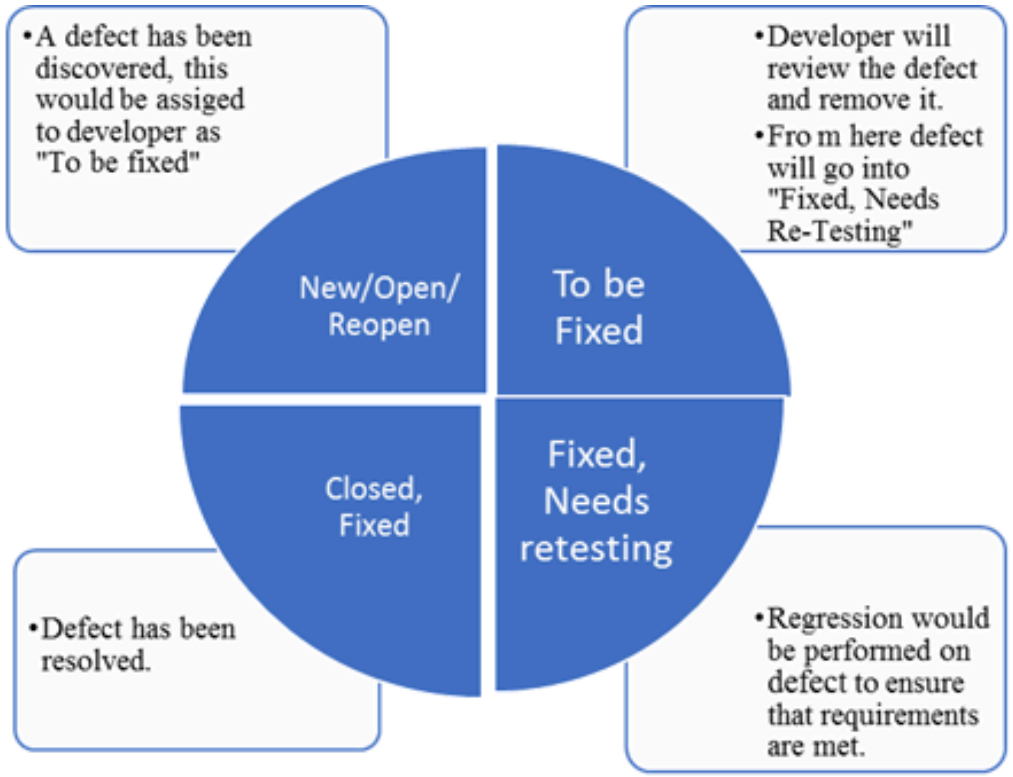


Figure 4.2 Testing Approach

4.5 Criteria

4.5.1 Entry Criteria

Testing would begin when Unit testing of developed modules/components is complete. And no bugs are found in the smoke test.

4.5.2 Suspension Criteria

- Testing will be suspended when Smoke Test or Critical test case bugs are discovered
- Testing will be suspended if there is critical scope change that impacts the requirement
- Suspend testing until the Development team fixes all the failed cases.

4.5.3 Exit Criteria

- When the app is stable, the Team agrees that the application meets functional requirements.
- Script execution of all test cases in all areas have passed.
- All priority 1 and 2 bugs have been resolved and closed.

Each test area has been signed off as completed by the Test Lead.

4.6 Test Case

Table 4.1: Test case for emotion detection (Happy)

Test	Detection of user's emotion
Type	Integration
Pre-condition	Webcam must be on
Steps	1. Turn on the camera 2. Click on the application icon and open the application. 3. Maintain a suitable distance from the camera.
Input	User portrays a happy face
Expected Result	Emotion : Happy
Actual Result	Emotion : Happy
Error	-

Table 4.2: Test case for emotion detection (Sad)

Test	Detection of user's emotion
Type	Integration
Pre-condition	Webcam must be on
Steps	1. Turn on the camera 2. Click on the application icon and open the application. 3. Maintain a suitable distance from the camera.
Input	User portrays a sad face

Expected Result	Emotion : Sad
Actual Result	Emotion : Sad
Error	-

Table 4.3: Test case for emotion detection (Surprise)

Test	Detection of user's emotion
Type	Integration
Pre-condition	Webcam must be on
Steps	1. Turn on the camera 2. Click on the application icon and open the application. 3. Maintain a suitable distance from the camera.
Input	User portrays a surprising face
Expected Result	Emotion : Surprise
Actual Result	Emotion : Surprise
Error	-

Table 4.4: Test case for emotion detection (Disgust)

Test	Detection of user's emotion
Type	Integration
Pre-condition	Webcam must be on
Steps	1. Turn on the camera 2. Click on the application icon and open the application. 3. Maintain a suitable distance from the camera.
Input	User portrays a disgust face
Expected Result	Emotion : Disgust
Actual Result	Emotion : Disgust
Error	-

Table 4.5: Test case for emotion detection (Angry)

Test	Detection of user's emotion
Type	Integration
Pre-condition	Webcam must be on
Steps	1. Turn on the camera 2. Click on the application icon and open the application. 3. Maintain a suitable distance from the camera.
Input	User portrays a angry face
Expected Result	Emotion : Angry
Actual Result	Emotion : Angry
Error	-

Table 4.6: Test case for emotion detection (Fear)

Test	Detection of user's emotion
Type	Integration
Pre-condition	Webcam must be on
Steps	1. Turn on the camera 2. Click on the application icon and open the application. 3. Maintain a suitable distance from the camera.
Input	User portrays a fearful face
Expected Result	Emotion : Fear
Actual Result	Emotion : Fear
Error	-

Table 4.7: Test case for emotion detection (Neutral)

Test	Detection of user's emotion
Type	Integration
Pre-condition	Webcam must be on
Steps	1. Turn on the camera 2. Click on the application icon and open the application. 3. Maintain a suitable distance from the camera.
Input	User portrays a neutral face
Expected Result	Emotion : Neutral
Actual Result	Emotion : Neutral
Error	-

TECHNOLOGIES USED

5.1 Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

1. web development (server-side),
2. software development,
3. mathematics,
4. system scripting.

Python was developed by Guido van Rossum in the early 1990's and its latest version is 3.8.0, we can simply call it Python3. Python 3.0 was released in 2008. and is interpreted language i.e it's not compiled and the interpreter will check the code line by line. This article can be used to learn the very basics of the Python programming language.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off-line as well.

For a description of standard objects and modules, see The Python Standard Library. The Python Language Reference gives a more formal definition of the language. To write extensions in C or C++, read Extending and Embedding the Python Interpreter

and Python/C API Reference Manual. There are also several books covering Python in depth.

This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most noteworthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in The Python Standard Library.

If you are on Windows OS download Python from its official site and now install from the setup and in the start menu type IDLE.IDLE, you can think of it as a Python's IDE to run the Python Scripts.

If you are on Linux/Unix-like just open the terminal and on 99% linux OS Python comes preinstalled with the OS. Just type 'python3' in the terminal and you are ready to go.

5.2. TensorFlow

Tensorflow is a computational framework for building machine learning models. TensorFlow provides a variety of different toolkits that allow you to construct models at your preferred level of abstraction. You can use lower-level APIs to build models by defining a series of mathematical operations. Alternatively, you can use higher-level APIs (like tf.estimator) to specify predefined architectures, such as linear regressors or neural networks.

The following figure shows the current hierarchy of TensorFlow toolkits:

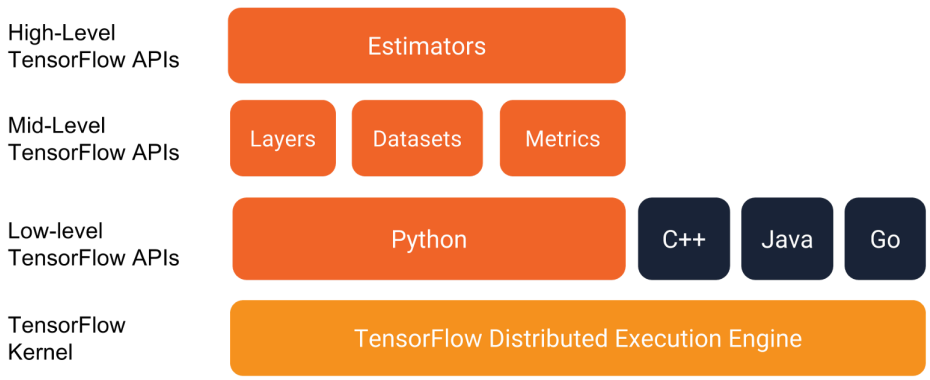


Figure 5.3: TensorFlow Hierarchy [27]

The following table summarizes the purposes of the different layers:

Table 5.1: TenserFlow toolkits

Toolkit(s)	Description
Estimator (tf.estimator)	High-level, OOP API.
tf.layers/tf.losses/tf.metrics	Libraries of common model components.
TensorFlow	Lower-level APIs

TensorFlow consists of the following two components:

1. a graph protocol buffer.
2. a runtime that executes the (distributed) graph.

These two components are analogous to Python code and the Python interpreter. Just as the Python interpreter is implemented on multiple hardware platforms to run Python code, TensorFlow can run the graph on multiple hardware platforms, including CPU, GPU, and TPU.

Which API(s) should you use? You should use the highest level of abstraction that solves the problem. The higher levels of abstraction are easier to use, but are also (by design) less flexible. We recommend you start with the highest-level API first and get everything working. If you need additional flexibility for some special modeling concerns, move one level lower. Note that each level is built using the APIs in lower levels, so dropping down the hierarchy should be reasonably straightforward.

5.3. IDLE

IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

5. coded in 100% pure Python, using the tkinter GUI toolkit.
6. cross-platform: works mostly the same on Windows, Unix, and macOS
7. Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
8. multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features

9. search within any window, replace within editor windows, and search through multiple files (grep)
10. debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
11. configuration, browsers, and other dialogs

5.3.1 Menus

IDLE has two main window types, the Shell window and the Editor window. It is possible to have multiple editor windows simultaneously. On Windows and Linux, each has its own top menu. Each menu documented below indicates which window type it is associated with.

Output windows, such as used for Edit => Find in Files, are a subtype of editor window. They currently have the same top menu but a different default title and context menu.

On macOS, there is one application menu. It dynamically changes according to the window currently selected. It has an IDLE menu, and some entries described below are moved around to conform to Apple guidelines.

- **Edit menu (Shell and Editor)**

1. Undo-Undo the last change in the current window. A maximum of 1000 changes may be undone.
2. Redo-Redo the last undone change in the current window.
3. Cut-Copy selection into the system-wide clipboard; then delete the selection.
4. Copy-Copy selection into the system-wide clipboard.
5. Paste-Insert contents of the system-wide clipboard into the current window.

- **Format menu (Editor window only)**

1. Indent Region-Shift selected lines right by the indent width (default 4 spaces).
2. Dedent Region-Shift selected lines left by the indent width (default 4 spaces).
3. Comment Out Region-Insert ## in front of selected lines.
4. Uncomment Region-Remove leading # or ## from selected lines.
5. Tabify Region-Turn *leading* stretches in spaces into tabs. (Note: We recommend using 4 space blocks to Python code.)

6. Untabify Region-Turn *all* tabs into the correct number of spaces.
7. Toggle Tabs-Open a dialog to switch between indenting with spaces and tabs.

- **Run menu (Editor window only)**

1. Python Shell-Open or wake up the Python Shell window.
2. Check Module-Check the syntax of the module currently open in the Editor window. If the module has not been saved, IDLE will either prompt the user to save or autosave, as selected in the General tab of the Idle Settings dialog. If there is a syntax error, the approximate location is indicated in the Editor window.
3. Run Module-Do Check Module (above). If there is no error, restart the shell to clean the environment, then execute the module. Output is displayed in the Shell window. Note that output requires use of print or write. When execution is complete, the Shell retains focus and displays a prompt. At this point, one may interactively explore the result of execution. This is similar to executing a file with a python -i file at a command line.

- **Shell menu (Shell window only)**

1. View Last Restart-Scroll the shell window to the last Shell restart.
2. Restart Shell-Restart the shell to clean the environment.
3. Previous History-Cycle through earlier commands in history which match the current entry.
4. Next History-Cycle through later commands in history which match the current entry.
5. Interrupt Execution-Stop a running program.

5.3.2 Key bindings

In this section, ‘C’ refers to the Control key on Windows and Unix and the Command key on macOS.

1. Backspace deletes to the left; Del deletes to the right
2. C-Backspace delete word left; C-Del delete word to the right
3. Arrow keys and Page Up/Page Down to move around
4. C-LeftArrow and C-RightArrow moves by words
5. Home/End go to begin/end of line
6. C-Home/C-End go to begin/end of file

Some useful Emacs bindings are inherited from Tcl/Tk:

1. C-a beginning of line
2. C-e end of line
3. C-k kill line (but doesn't put it in clipboard)
4. C-l center window around the insertion point
5. C-b go backward one character without deleting (usually you can also use the cursor key for this)
6. C-f go forward one character without deleting (usually you can also use the cursor key for this)
7. C-p go up one line (usually you can also use the cursor key for this)
8. C-d delete next character

Code Context

Within an editor window containing Python code, code context can be toggled in order to show or hide a pane at the top of the window. When shown, this pane freezes the opening lines for block code, such as those beginning with class, def, or if keywords that would have otherwise scrolled out of view. The size of the pane will be expanded and contracted as needed to show the all current levels of context, up to the maximum number of lines defined in the Configure IDLE dialog (which defaults to 15). If there are no current context lines and the feature is toggled on, a single blank line will display. Clicking on a line in the context pane will move that line to the top of the editor.

The text and background colors of the context pane can be configured under the Highlights tab in the Configure IDLE dialog.

5.3.3 Python Shell window

With IDLE's Shell, one enters, edits, and recalls complete statements. Most consoles and terminals only work with a single physical line at a time.

When one pastes code into Shell, it is not compiled and possibly executed until one hits Return. One may edit pasted code first. If one pastes more that one statement into Shell, the result will be a SyntaxError when multiple statements are compiled as if they were one.

The editing features described in previous subsections work when entering code interactively. IDLE's Shell window also responds to the following keys.

1. C-c interrupts executing command
2. C-d sends end-of-file; closes window if typed at a >>> prompt
3. Alt-/ (Expand word) is also useful to reduce typing
4. Command history
5. Alt-p retrieves previous commands matching what you have typed. On macOS use C-p.
6. Alt-n retrieves next. On macOS use C-n.
7. Return while on any previous command retrieves that command

5.3.4 Startup and code execution

Upon startup with the -s option, IDLE will execute the file referenced by the environment variables IDLESTARTUP or PYTHONSTARTUP. IDLE first checks for IDLESTARTUP; if IDLESTARTUP is present the file referenced is run. If IDLESTARTUP is not present, IDLE checks for PYTHONSTARTUP. Files referenced by these environment variables are convenient places to store functions that are used frequently from the IDLE shell, or for executing import statements to import common modules.

In addition, Tk also loads a startup file if it is present. Note that the Tk file is loaded unconditionally. This additional file is.Idle.py and is looked for in the user's home directory. Statements in this file will be executed in the Tk namespace, so this file is not useful for importing functions to be used from IDLE's Python shell.

5.4 KERAS

Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.

It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

1. Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles:

2. Modularity: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
3. Minimalism: The library provides just enough to achieve an outcome, no frills and maximizing readability.
4. Extensibility: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
5. Python: No separate model files with custom file formats. Everything is native Python.

The core data structure of Keras is a model, a way to organize layers. The simplest type of model is the Sequential model, a linear stack of layers. For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs in layers.

The focus of Keras is the idea of a model.

The main type of model is called a Sequence which is a linear stack of layers.

You create a sequence and add layers to it in the order that you wish for the computation to be performed.

Once defined, you compile the model which makes use of the underlying framework to optimize the computation to be performed by your model. In this you can specify the loss function and the optimizer to be used.

Once compiled, the model must be fit to data. This can be done one batch of data at a time or by firing off the entire model training regime. This is where all the compute happens.

Once trained, you can use your model to make predictions on new data.

Keras can be installed easily using PyPI, as follows:

- sudo pip install keras

You can check your version of Keras on the command line using the following snippet:

- python -c "import keras; print keras.__version__"

We can summarize the construction of deep learning models for Keras as follows:

1. Define your model. Create a sequence and add layers.
2. Compile your model. Specify loss functions and optimizers.
3. Fit your model. Execute the model using data.
4. Make predictions. Use the model to generate predictions on new data.

LITERATURE SURVEY

Face detection is a computer technology that determines the location and size of a human face in an 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 parts in the given image i.e. feature base and image base approach. Feature based approach tries to extract features of the image and match it against the knowledge of the face features. The image base approach tries to get the best match between training and testing images.

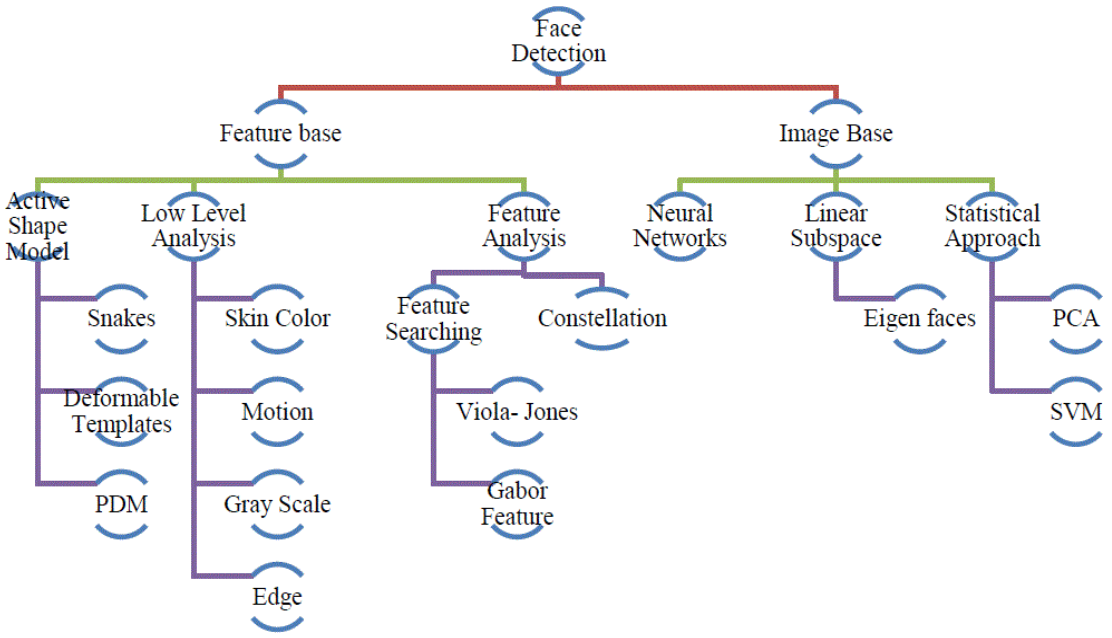


Figure 6.1: Detection methods

6.1 FEATURE BASED APPROACH

Active Shape Model Active 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 modeled object in an image. When 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) 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. 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 assumes the shape of the head. The evolution of a snake is achieved by minimizing an energy function, Esnake (analogy with physical systems), denoted as Esnake = Einternal + EExternal Where Einternal and EExternal 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 equilibrium. Two main considerations for forming snakes i.e. selection of energy terms and energy minimization. Elastic energy is used commonly as internal energy. Internal energy varies 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 the other side external energy relies on image features. Energy minimization process is done by optimization techniques such as the steepest gradient descent. Which needs the 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.

6.1.1 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 et 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 template 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.E = Ev + Ee + Ep + Ei + Einternal; where Ev , Ee , Ep , Ei , Einternal are external energy due to valley, edges, peak and image brightness and internal energy.

6.1.2 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 examples and 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 brooches. Ideal Point Distribution Models can only deform in ways that are characteristic of the object. Cootes 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 modeling 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 Cootes and Taylor in.

6.2 LOW LEVEL ANALYSIS:

Based on low level visual features like color, intensity, edges, motion etc. Skin Color or Base Color is a vital 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 factors.

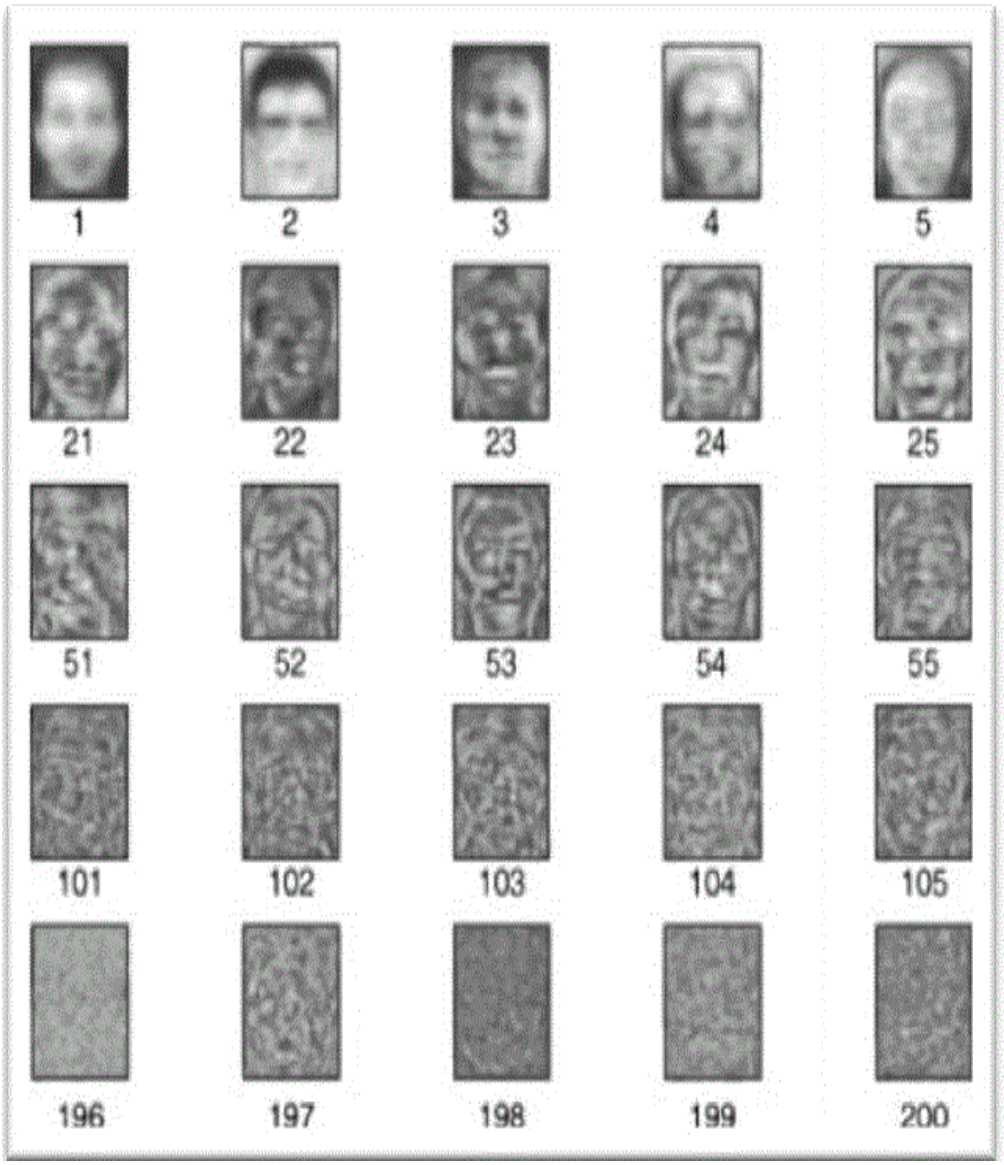


Figure 6.2: Face detection

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 a bounding box to extract the face image.

Crowley and Coutaz 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 regions like legs, arms, etc. Cahi and Ngan suggested a skin color classification algorithm with YCbCr color space. Research found that pixels belonging to skin regions having similar Cb and Cr values. So that the thresholds are 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. Kjeldson and Kender defined a color predicate in HSV color space to separate skin regions from background. Skin color classification in HSI 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 the above two algorithms this algorithm is also having the same constraint.

6.3 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 features can be located by frame differences.

6.3.1 Gray Scale Base:

Gray information within a face can also be treat as important features. Facial features such as eyebrows, pupils, and lips appear generally darker than their surrounding facial regions. Various recent feature extraction algorithms search for local gray minima within segmented facial regions. In these algorithms, the input images are first enhanced by contrast-stretching and gray-scale morphological routines to improve the quality of local dark patches and thereby make detection easier. The extraction of dark

patches is achieved by low-level gray-scale thresholding. Based method and consist of three levels. Yang and Huang presented a new approach i.e. faces gray scale behavior in pyramid (mosaic) images. This system utilizes hierarchical Face location consisting of three levels. Higher two levels based on mosaic images at different resolutions. In the lower level, an edge detection method is proposed. Moreover this algorithm gives a fine response in a complex background where the size of the face is unknown.

6.3.2 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.'s work to trace a human head outline. Then after remarkable works were carried out by many researchers in this specific area. Method suggested by Anila and Devarajan was very simple and fast. They proposed a framework which consists of 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 sobel 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.

6.4 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

6.4.1 Feature Searching

6.4.1.1 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 over complete 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.

6.4.2 Gabor Feature Method:

Sharif et al proposed an Elastic Bunch Graph Map (EBGM) algorithm that successfully implements face detection using Gabor filters. The proposed system applies 40 different Gabor filters on an image. As a result of which 40 images with different angles and orientation are received. Next, maximum intensity points in each filtered image are calculated and mark them as fiducial points. The System reduces these points in accordance with the distance between them. The next step is calculating the distances between the reduced points using the distance formula. At last, the distances are compared with the database. If match occurs, it means that the faces in the image are detected. Equation of the Gabor filter is shown below.

$$\psi_{u,v}(z) = \frac{\| k_{u,v} \|^2}{\sigma^2} e^{\left(\frac{\|k_{u,v}\|^2 \|z\|^2}{2\sigma^2} \right)} \left[e^{i\vec{k}_{u,v} z} - e^{-\frac{\sigma^2}{2}} \right]$$

Where

$\phi_u = \frac{u\pi}{8}$, $\phi_u \in [0, \pi)$ gives the frequency,

Figure 6.3: Formula

6.5 CONSTELLATION METHOD

All methods discussed so far are able to track faces but still some issues like locating faces of various poses in complex backgrounds is truly difficult. To reduce this difficulty investigators form a group of facial features in face-like constellations using more robust modeling 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 multi-scale 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.

6.5.1 Neural Network

Neural networks are 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 networks were proposed by Agui et al. The first stage has two parallel sub-networks 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 Samal developed one of the earliest neural networks for face detection. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units. Feraud and Bernier presented a detection method using auto associative neural networks. The idea is based on 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 networks (PDBNN). The architecture of PDBNN is similar to a radial basis function (RBF) network with modified learning rules and probabilistic interpretation.

6.6 LINEAR SUBSPACE METHOD

6.6.1 Eigenfaces Method:

An early example of employing eigenvectors in face recognition was done by Kohonen in which a simple neural network is demonstrated to perform face recognition for aligned and normalized face images. Kirby and Sirovich 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 by HOTELLING 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 Eigenvectors 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 the variance)

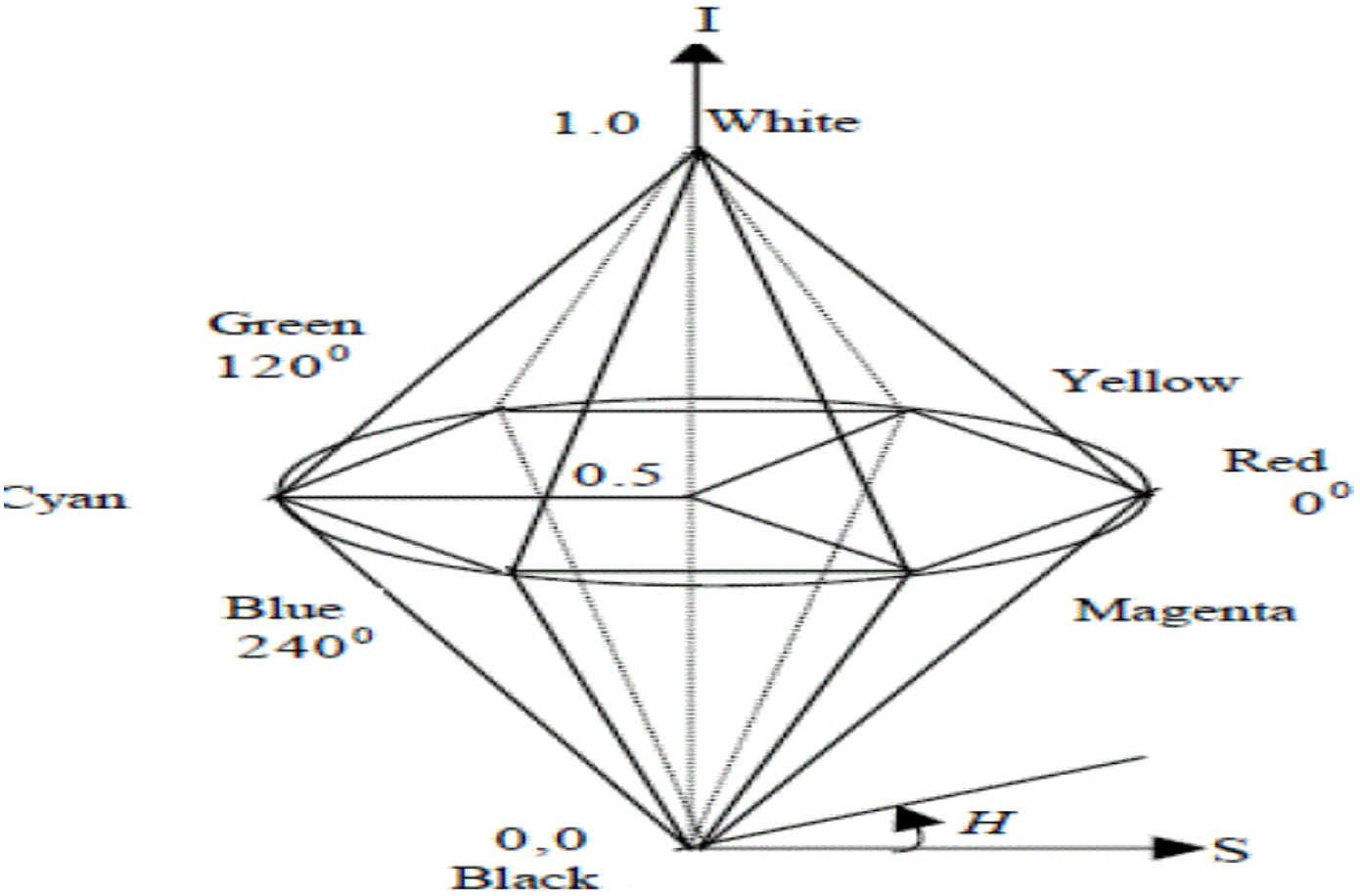


Figure 4.4: Eigen Faces

6.7 STATISTICAL APPROACH

6.7.1 Support Vector Machine (SVM):

SVMs were first introduced by 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 work on an 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 hyperplane is chosen to minimize the expected classification error of the unseen test patterns. In Osuna Et 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 Poggio. SVMs have also been used to detect faces and pedestrians in the wavelet domain.

7.1 OVERVIEW

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.

2. Real-time face detection.

7.2 FACE DETECTION IN IMAGES

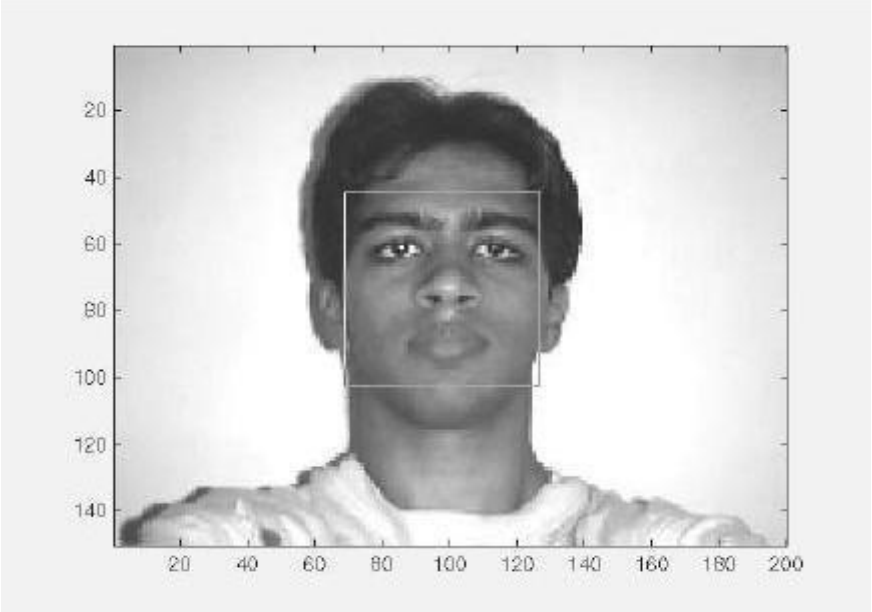


Figure 7.1: Successful face detection in an image with a frontal view of a human face.

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 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 'non-face' 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 'non-face' 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 individual’s 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).

7.3 REAL-TIME FACE DETECTION

Real-time face detection involves detection of a face from a series of frames from a video-capturing device. While the hardware requirements for such a system are far more stringent, from a computer vision standpoint, real-time face detection is actually a far simpler process than detecting 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 7.2: Frame 1 from camera

Figure 7.3: Frame 2 from camera

Figure 7.4: 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-temporal 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).Furthermore 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.

7.4 FACE DETECTION PROCESS

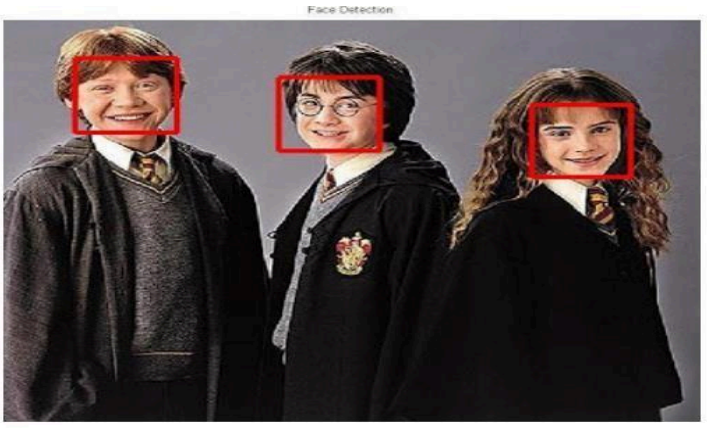
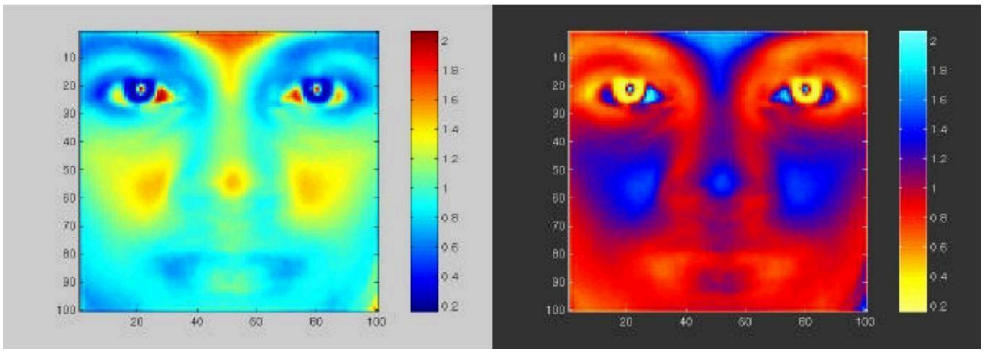


Figure 7.5: Face detection

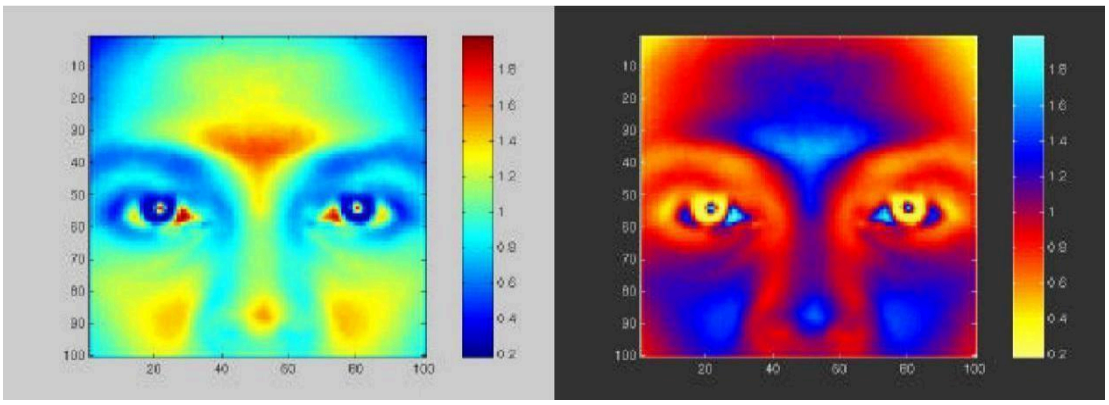
It is the process of identifying different parts of human faces like eyes, nose, mouth, etc. This process can be achieved by using MATLAB code. In this project the author will attempt to detect faces in still images by using image invariants. To do this it would be useful to study the gray-scale intensity 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 color map has been used to highlight gray-scale intensity differences.



Scaled color map **scaled color map (negative)**

Figure 7.6: Average human face in gray-scale

The gray-scale differences, which are invariant across all the sample faces, are strikingly apparent. The eye-eyebrow area seems to always contain dark intensity (low) gray-levels while nose forehead and cheeks contain bright intensity (high) gray-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 color map **Scaled color map (negative)**

Figure 7.7 Area chosen for face detection (indicated on average human face in grayscale)

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.

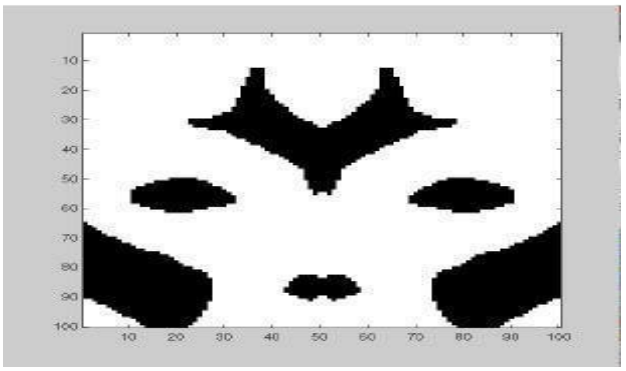


Figure 7.8: Basis for a bright intensity invariant sensitive template.

Note the slight differences which were made to the bright intensity invariant sensitive template which were needed because of the pre-processing done by the system to overcome irregular lighting. 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.

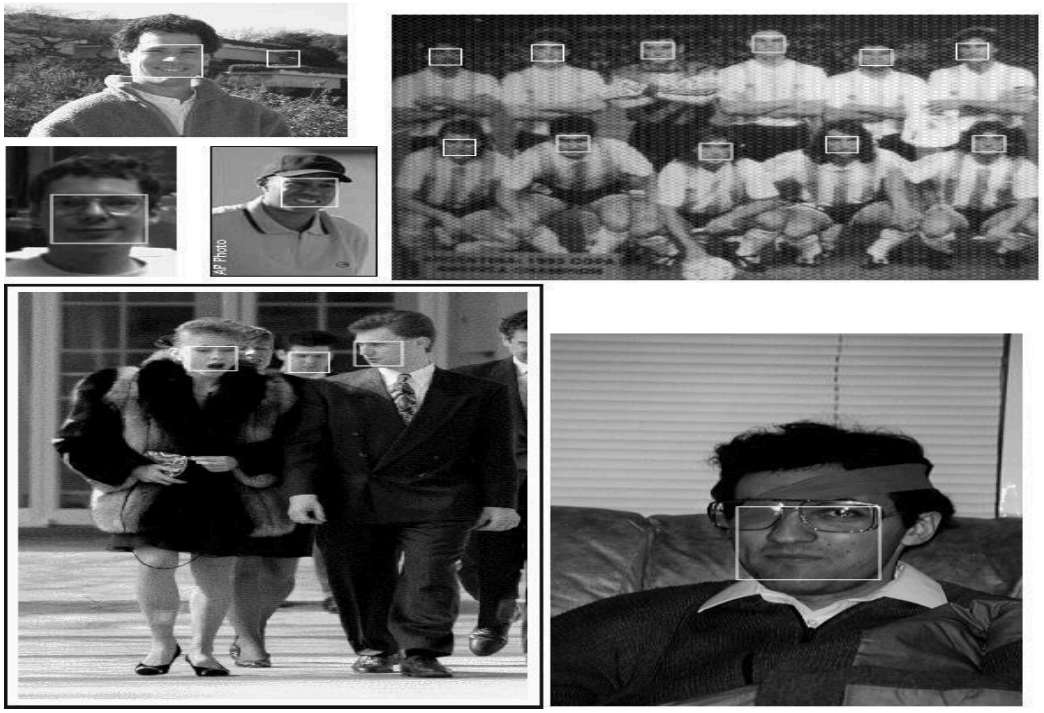


Figure 7.9: Scanned image detection

- San window over image
- Classify window as either

7.5 FACE DETECTION ALGORITHM

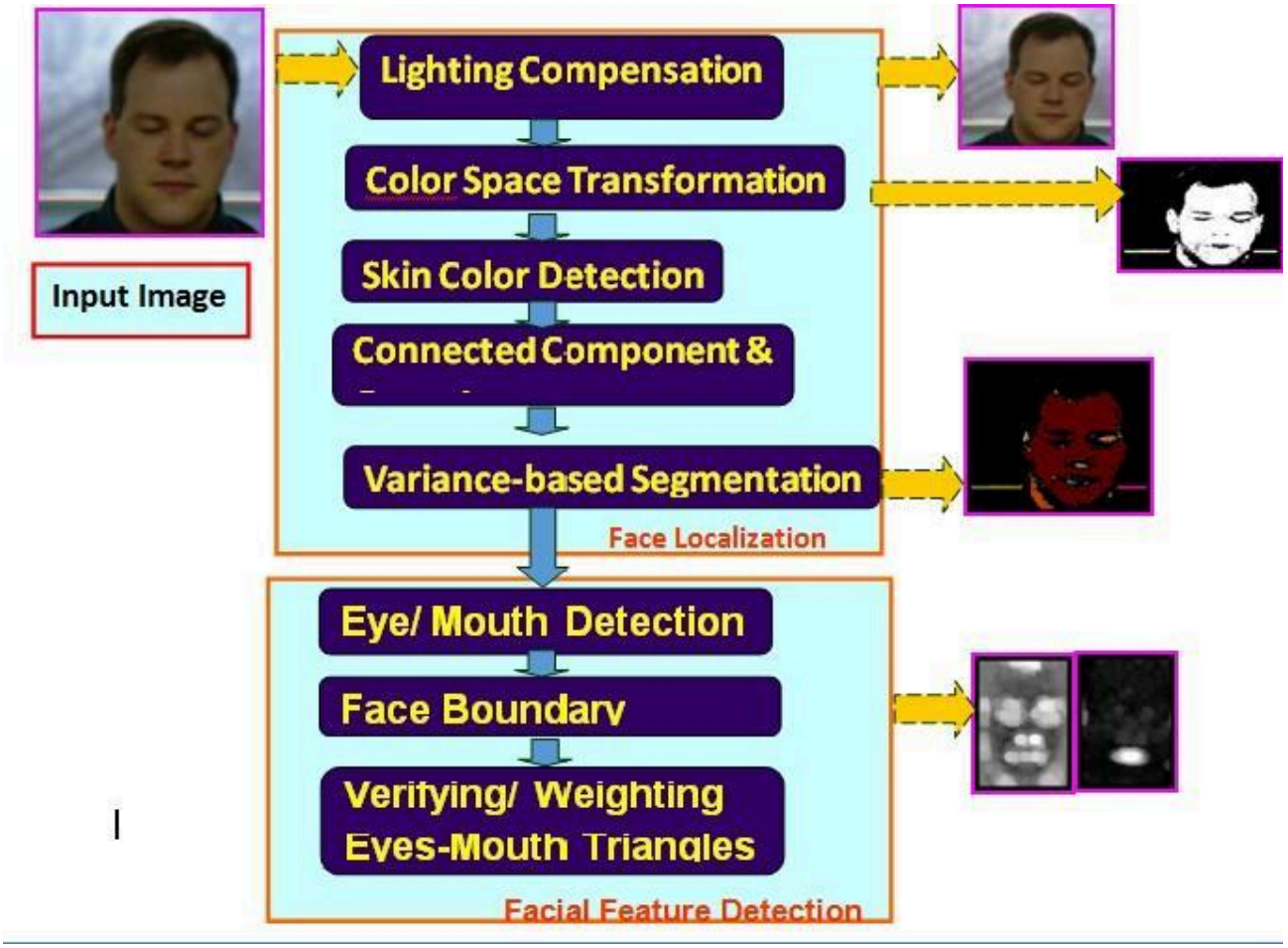


Figure 7.10 Face detection algorithm

Figure 7.11: Mouth detection

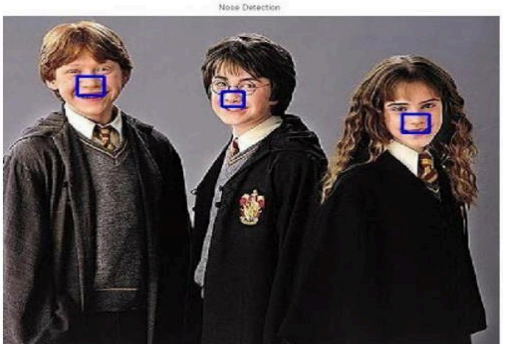


Figure 7.12: Noise detection

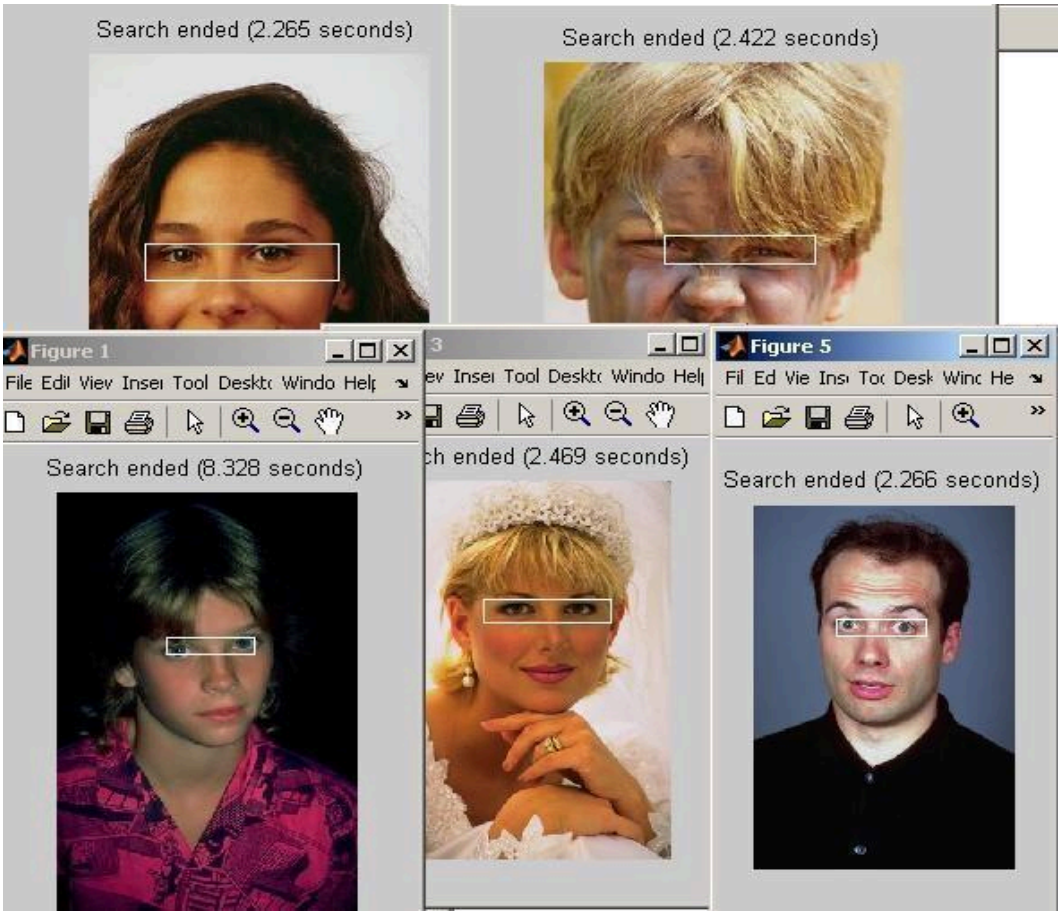


Figure 7.13 Eye detection

8.1 OVERVIEW

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.

8.2 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.

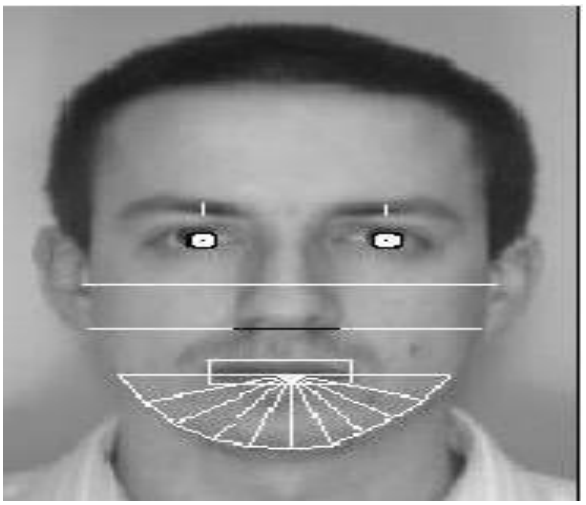


Figure 8.1 Geometrical features (white) which could be used for face recognition

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.

8.2.1 Face recognition using template matching

This is similar to 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.

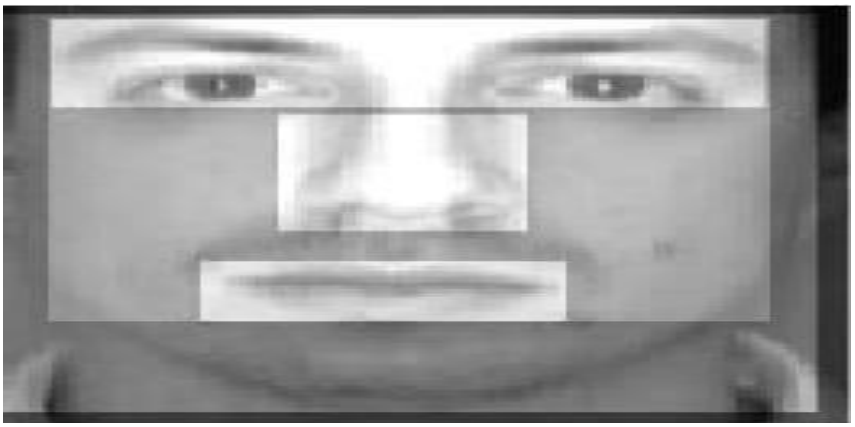


Figure 8.2 Face recognition using template matching

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 gray-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 pre- processing and transformation of the extracted gray-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.

8.3 PROBLEM SCOPE 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) 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 variency.
5) All systems must possess near real-time performance.
6) Both fully automated and manual face detection must be supported
7) Frontal view face recognition will be realized 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 invariant 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 real- world.

8.4 BRIEF OUTLINE 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.

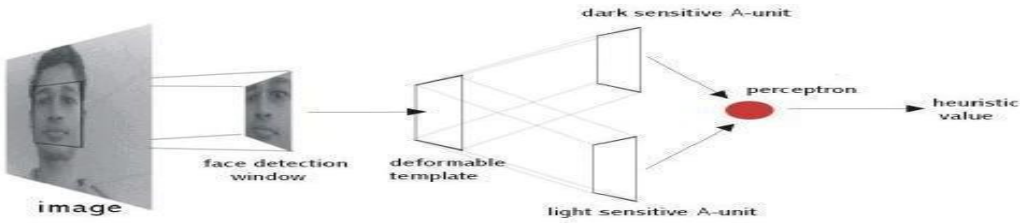


Figure 8.3: Implemented fully automated frontal view face detection mode

A manual face detection system was realized 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 feature 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 highly robust in pattern recognition tasks and because it is relatively simple to implement. The author would also like 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.

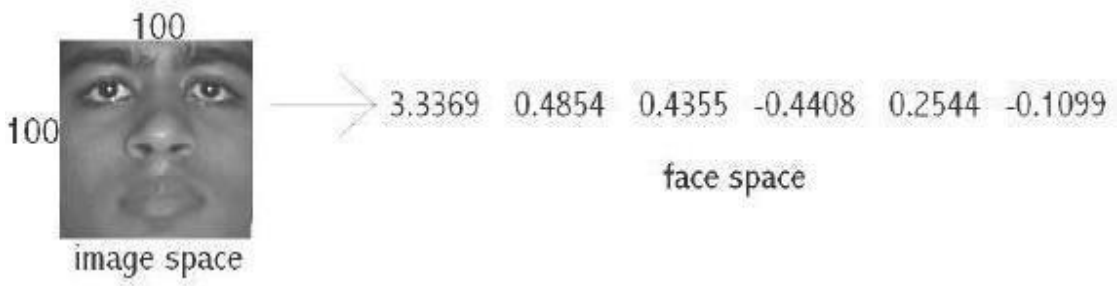


Figure 8.4 Principal Component Analysis transform from 'image space' to 'face space'.

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 human applications ranging from static, mug-shot verification in a cluttered background.

Figure 8.5 Face Recognition

8.5 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

8.5.1 Inter - class similarity:

Different persons may have very similar appearance

Figure 8.6 Face recognition twins and other is father 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.

8.5.2 Inter – Class Variability

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

8.6 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 categorize faces using familiar geometrical differences, such as nose length or eyebrow width. Instead, a set of human faces is analyzed using PCA to determine which 'variables' account for the variance of faces. In face recognition, these variables are called eigenfaces 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 eigenvalues. Each eigenvalue describes the magnitude of each eigenface in each image. Furthermore, all interpretation (i.e. recognition) operations can now be done using just the 40 eigenvalues to represent a face instead of 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.

8.7 UNDERSTANDING EIGEN FACES

Any grayscale 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.

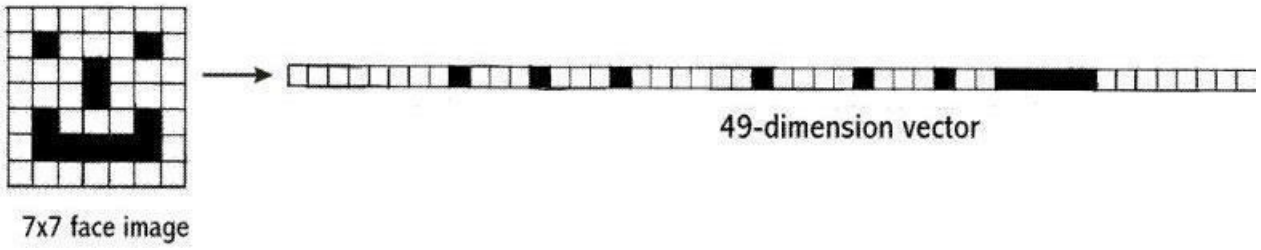


Figure 8.7: A 7x7 face image transformed into a 49 dimension vector

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.

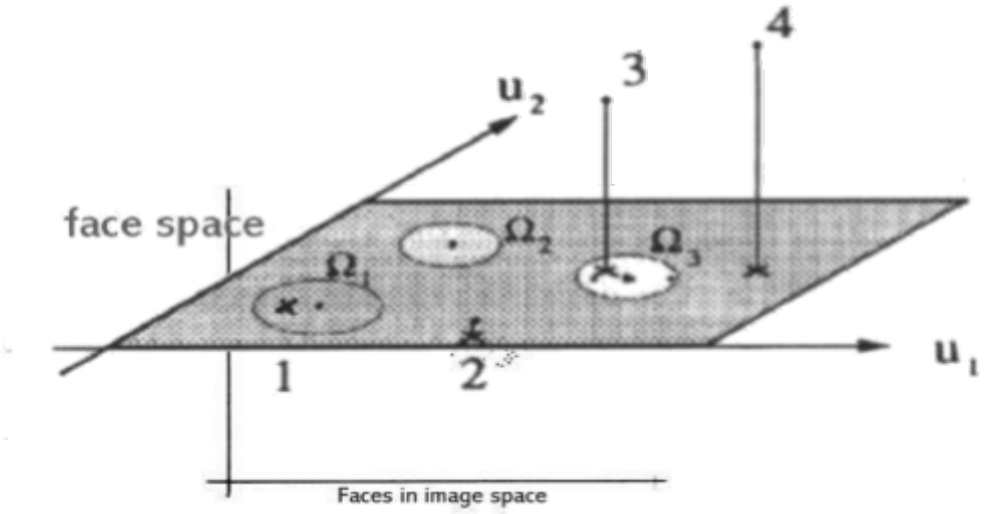


Figure 8.8: Eigen plane

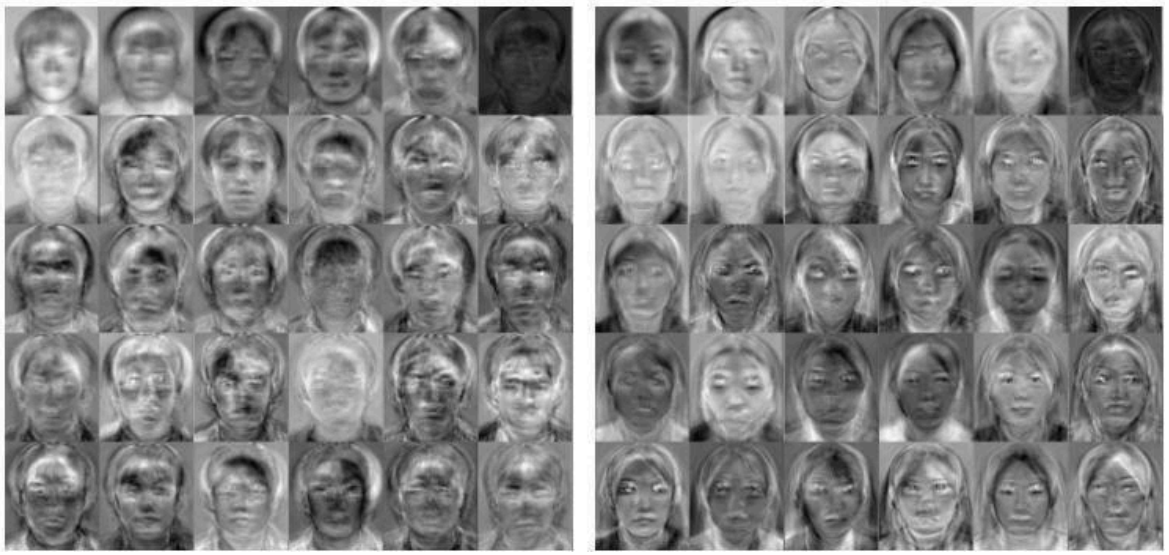


Figure 8.9: Eigen faces

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 recognize that face. Toole et al. (1993) did some interesting work on the importance of eigenfaces with large and small eigenvalues. They showed that the eigenvectors with larger eigenvalues convey information relative to the basic shape and structure of the faces. This kind of information is most useful in categorizing faces according to sex, race etc. Eigenvectors 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 eigenfaces were sufficient for a very good description of human faces since the reconstructed image has only about 2% RMS. pixel-by-pixel errors.

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 high correlation probably contain 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.

8.8 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 the face database from a single known individual. Unknown image from the same individual to be identified.

Figure 8.10 Pose invariant face recognition.

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

RESULT

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 recognizes 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 patterns namely, the Eigenface method and the Fisherface 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 Eigenfaces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project's face detection system with face recognition is Image processing. The software requirements for this project is matlab software.

Extension: There are a vast number of applications from this face detection project, this project can be extended so that the various parts in the face can be detected which are in various directions and shapes. The application uses OpenCV, deep learning, convolutional neural networks, keras and python to detect human emotions like fear, anger, happiness etc.

The application successfully detected the emotion which is being expressed in the camera like fear, happy, sad, disgust, surprise, angry etc.

SUMMARY AND 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 normalize 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 it’s 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 for 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.

REFERENCES

1. Adelson, E. H., and Bergen, J. R. (1986) The Extraction of Spatio-Temporal Energy in
2. Human and Machine Vision, Proceedings of Workshop on Motion: Representation and
3. Analysis (pp. 151-155) Charleston, SC; May 7-9
4. AAFPRS(1997). A newsletter from the American Academy of Facial Plastic and Reconstructive Surgery. Third Quarter 1997, Vol. 11, No. 3. Page 3.
5. Baron, R. J. (1981). Mechanisms of human facial recognition. International Journal of Man Machine Studies, 15:137-178
6. 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
7. Bichsel, M. (1991). Strategies of Robust Objects Recognition for Automatic Identification of Human Faces. PhD thesis, , Eidgenossischen Technischen Hochschule, Zurich.
8. Brennan, S. E. (1982) The caricature generator. M.S. Thesis. MIT.
9. Brunelli, R. and Poggio, T. (1993), Face Recognition: Features versus Templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(10):1042-1052
10. Craw, I., Ellis, H., and Lishman, J.R. (1987). Automatic extraction of face features. Pattern Recognition Letters, 5:183-187, February.
11. Deffenbacher K.A., Johanson J., and O'Toole A.J. (1998) Facial ageing, attractiveness, and distinctiveness. Perception. 27(10):1233-1243
12. Dunteman, G.H. (1989) Principal Component Analysis. Sage Publications.
13. Frank, H. and Althoen, S. (1994). Statistics: Concepts and applications. Cambridge University Press. p.110
14. Gauthier, I., Behrmann, M. and Tarr, M. (1999). Can face recognition really be dissociated from object recognition? Journal of Cognitive Neuroscience, in press.
15. Goldstein, A.J., Harmon, L.D., and Lesk, A.B. (1971). Identification of human faces. In Proc. IEEE, Vol. 59, page 748
16. 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
17. Development, Department of Psychology, Birkbeck College.
18. Haralick, R.M. and Shapiro, L.G.. (1992) Computer and Robot Vision, Volume I. Addison-Wesley
19. 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.
20. Heisele, B. and Poggio, T. (1999) Face Detection. Artificial Intelligence Laboratory. MIT.
21. Jang. J., Sun, C., and Mizutani, E. (1997) Neuro-Fuzzy and Soft Computing. Prentice Hall.
22. Johnson, R.A., and Wichern, D.W. (1992) Applied Multivariate Statistical Analysis. Prentice Hall. p356-357.
23. https://github.com/Abhishek-dev2/OpenCV-Face-Recognition?files=1
24. http://www.tannerhelland.com/4660/dithering-eleven-algorithms-source-code/
25. https://stackoverflow.com/questions/7584489/python-print-statement-syntax-error-invalid-syntax
26. Hadamard, J. (1923) Lectures on the Cauchy Problem in Linear Partial Differential Equations, Yale University.
27. https://developers.google.com/machine-learning