On Algorithms for Facial Expression Recognition and their Hardware Implementation

by

Rajesh A. Patil

ID 2008REC106

Submitted
in fulfillment of the requirements
of
the degree of

DOCTOR OF PHILOSOPHY

to the

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR, INDIA

October 2016

Certificate

This is to certify that the thesis entitled ***'On Algorithm for Facial Expression Recognition and their Hardware Implementation,*** being submitted by **Rajesh A. Patil** to the Department of Electronics and Communication Engineering, Malaviya National Institute of Technology, Jaipur, for the award of the degree of Doctor of Philosophy, is a bonafide research work carried out by him under my supervision and guidance. The results obtained in this thesis have not been submitted to any other university or institute for the award of any other Degree or Diploma.

Dr. Vineet Sahula
Professor
Department of ECE
MNIT, Jaipur
Jaipur - 302017, India

Dr. A. S. Mandal
Chief Scientist
IC Design Group
CEERI, Pilani
Rajasthan, India

Acknowledgement

I would like to dedicate this work to my mother Late Smt. Shakuntala who left me during my Ph.D. journey. Her love provided me inspiration and was my driving force.

Completing my PhD degree is probably the most challenging activity of my life. The best and worst moments of my doctoral journey have been shared with many people. It has been a great privilege to spend several years in the Department of Electronics and Communication Engineering at MNIT Jaipur and its members will always remain dear to me.

At the outset I would like to express my appreciation to my supervisor Dr. Vineet Sahula, Professor and Head of the Department for his advice during my doctoral research endeavor for the past six years. As my supervisor, he constantly forced me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction of research and to move forward with the investigation in depth. My sincere thanks and gratitude, to him for his invaluable guidance, support, and encouragement.

I would like to thank my co supervisor Dr. A. S. Mandal, Principal Scientist CEERI Pilani, who helped me to start my doctoral research with a smile. It has been my privilege to work closely with him. I have enjoyed the opportunity to watch and learn from his knowledge and experience. His frequent insights and patience with me are always appreciated. I am very grateful for his motivation, enthusiasm, and immense knowledge in machine intelligence.

Special thanks to my committee members, Dr. D. Bhoolchandani, Dr. Vijay Janyani, Dr. Md. Salim, Dr. C. Periasamy, Dr. Lava Bhargava, and Dr. S. J. Nanda for their support, guidance and helpful suggestions. Their guidance has served me well and I owe them my heartfelt appreciation.

During my time as a Ph. D. student, I have met many nice and inspiring colleagues. This made my, sometimes a bit isolated, working life at MNIT Jaipur quite enjoyable. Thank you all! A special thanks go to Renu Kumawat, Lokesh Garg, Sapna Khandelwal and Arjun. I am very thankful to Anshul, Sandeep, Mayank, Monika, Gauri and Rahul for their kind support during my Ph.D. work.

I am thankful to Naatyashastra Institute of fine arts, Navi Mumbai for their help in preparing the database of navras, “Bharatnatyam” a classical dance style of south India.

I deeply appreciate and acknowledge Dr. Hemant Taskar, Principal Govt. polytechnic Mumbai for his support and valuable suggestions.

I am thankful to my colleagues Mrs. Usha Khake, Ms. Pooja Chelani and Ms. Sadaf Shaikh for their cooperation and encouragement.

My hard-working parents have sacrificed their lives for me and provided unconditional love and care. I love them so much, and I would not have made it this far without them.

Above and beyond all, my heartfelt gratitude to my Sister Sucheta and brother in Law Sanjay for their much needed support, patience, understanding, and encouragement in every possible way.

I also want to thank to my Father-in-law and Mother in law for their unconditional support. Special thanks to Ratnakar, Vidya, Yogesh and Deepa. Their endless love, priceless, perpetual, indispensable help, support and everything made all this possible.

This last word of acknowledgment I have saved for my dear wife Vaishali, who has been constantly with me during all these years and has made them the best years of my life. Her love and encouragement made this journey of PhD comfortable. She already has my heart, so I will just give her a heartfelt "thanks." To my beloved daughter Ritika and Son Rushikesh I would like to express my thanks for being such a good kids always cheering me up.

Finally I thank my God, for letting me through all the difficulties. I have experienced Your guidance day by day. You are the one who let me finish my degree. I will keep on trusting You for my future. Thank you, Lord.

Rajesh Patil

Abstract

Machine vision has been defined as "the automatic acquisition and analysis of images to obtain desired data for interpreting a scene or controlling an activity" [1]. Machine vision is a complex task and it is infinitely complex for computers to perform. However, it seems relatively trivial to humans. The machine vision systems need significant advancements to be able to deal with real world applications. Some of these applications are navigation, target recognition, remote sensing, photo interpretation, etc. Now a days, interest has been growing in improving all aspects of the interaction between humans and computers. There is a need for the computer to interact naturally with the user (HCI), similar to the way human-human interaction takes place. In day to day life, facial expressions are commonly used for human-to-human communication, such as one smiles to show greetings or happiness, frowns to express sadness. Facial expressions are important, since they carry much information about human's feelings, emotions and so on. To create human like robots and machines, automatic facial expression recognition system with high accuracy and performance is required. Besides, humanoid robots and human computer interaction, expression recognition systems can be used in other domains. Telecommunications, Behavioral Science, Video Games, Animations, Automobile Safety, Psychiatry, Televisions, Educational Software, etc. to name a few. For a human being, detection and interpretation of faces and facial expressions in a scene is a simple and natural task, but for a machine this task is equally difficult. Several related problems during facial expression recognition are face detection, feature extraction, tracking and most significantly classification. Though much progress has been made, recognition of facial expression with a high accuracy still remains difficult due to large variety in the types of faces and facial expressions. We can see an extremely large variety in pose, resolution, orientation and lighting conditions.

In this thesis, we have proposed two algorithms. One is detecting facial expressions from image sequence and other is detecting facial expressions from still image. For facial expression recognition in image sequences, we have proposed algorithm which uses Candide wire frame model. Face is detected using Viola Jones algorithm. Facial features are detected using image normalization, and thresholding techniques. Wire frame model is automatically fitted on the first frame of the face image sequence. In the subsequent frames of the image sequence, facial features are tracked using Active Appearance Model (AAM). Once the model fits on the first frame, animation parameters of the model are set to zero, in order to obtain shape of the model for neutral facial expression of the same face. In the

subsequent frames, the model changes the shape as per facial expressions. The last frame of the image sequence corresponds to the greatest facial expression intensity. The geometrical displacement of wire frame nodes, between the neutral expression frame and the last frame, is used as an input to the multiclass support vector machine (SVM). SVM classifies facial expression into one of the class such as happy, surprise, sad, anger, disgust, fear and neutral. This method is applicable for frontal as well as tilted faces with an angle 30, 45 and 60 degrees with respect to Y axis. The proposed classifier works on geometrical deformation of feature vectors. That is, it does not use texture information.

Many tasks of a vision system are trivial low level algorithms, where the same instructions are applied to each pixel in the frame. This increases the processing time. Hence, in this thesis we have implemented these low level algorithms more efficiently on a parallel structure such as array of processing elements (PEs) mapped on to the Field Programmable Gate Array (FPGA). Hardware implementation of facial expression recognition is done using systolic array architecture. Systolic array architecture provides efficient data transfer and memory management mechanisms, and reduced complexity. Partial reconfiguration scheme is used to obtain power optimal mapping of the design.

Initially the algorithm is trained and tested on Cohn Kanade database for seven facial expressions such as happy, anger, disgust, fear, sad, surprise and neutral. Later on we have designed our own database for nine facial expressions based on Indian classical dance style "Bharatnatyam". These nine facial expressions are Shringar, Virya, Roudra, Karuna, Bibhtsa, Shanta, Adbhuta, Bhayanak, Hasya. Here SVM classifier is giving the lower accuracy, so we have designed a sparse representation classifier with multiple kernel and showed that it outperforms the other classifiers.

For facial expression recognition in still images, we have proposed another algorithm. Topographical maps begin to be recognized as one of the major computational structures underlying neural computations in the brain. They provide dimension reducing feature spaces that seem to be established and maintained under the participation of self organizing adaptive processes. The structure of these maps can be replicated by simple adaptive processes and can be realized by the use of a mathematical model. We have proposed a mathematical model to replicate the neural computation occurring in the brain for different facial expressions. Using this model, we obtained different patterns for different facial expressions of many persons. These patterns are then classified to one of the seven basic facial expressions such as happy, anger, disgust, fear, sad, surprise and neutral. We have used artificial neural network to classify these patterns.

Contents

List of Figures

List of Tables

List of Abbreviations

AAA	Active Appearance Algorithm
AAM	Active Appearance Model
AFA	Automatic Face Analysis
ANN	Artificial Neural Network
ASM	Active Shape Model
AU	Action Unit
CLBs	Configurable Logic Blocks
CPU	Central Processing Unit
DBN	Dynamic Bayesian Network
FA	Facial Animation
FACS	Facial Action Coding System
FAP	Facial Animation Parameter
FER	Facial Expression Recognition
FFP	Facial Feature Point
FFT	Fast Fourier Transform
FP	Feature Points
FPGA	Field Programmable Gate Array

HMM	Hidden Markov Model
ICAP	Internal Configuration Access Port
KCCA	Kernel Canonical Correlation Analysis
LBP	Local Binary Pattern
LG	Labelled Graph
LOIO	Leave One Image Out
LOSO	Leave One Subject Out
LUT	Look Up Table
MPEG	Moving Pictures Experts Group
MU	Motion Unit
NB	Nave Bayesian
NN	Neural Network
PBVD	Piecewise Bezier Volume Deformation
PC	Personal Computer
PCA	Principal Component Analysis
PEs	Processing Elements
PLBP	Pyramidal Local Binary Pattern
PR	Partial Reconfiguration
RBF	Radial Basis Function
SOFM	Self Organizing Feature Map
SRC	Sparse representation classifier
SSE	Summed Square Error
SVM	Support Vector Machine

Chapter 1

Introduction

Machine vision is a system in which, machine performs functions similar to those performed by human intelligence. These functions are one or more of learning, reasoning, self correcting, and adapting. It is possible to develop systems that mimic and surpass some human capabilities, such as sensing, correlating, speed of calculations, and deducing using computer technology. However, intelligence is not involved in such systems. The goal of machine vision research is to design a machine with perception capabilities like human, so that they can sense the environment, understand the sensed data. After sensing the data, machine is supposed to take appropriate decisions and sometimes actions. Machine should learn from this experience in order to enhance future performance. The field of machine vision has arised from the application of classical pattern recognition and image processing methods to advanced techniques in image understanding, like model based and knowledge based vision. Significant advancements have been happening in machine vision to deal with real world applications, such as navigation, target recognition, photo interpretation, remote sensing, etc. A variety of practical applications of the machine learning research are emerging, due to recent advances in hardware and software.

Now a days, interest has been growing in improving all aspects of the interaction between humans and computers. With the rapid advance of technology in recent years computers are becoming cheaper and more powerful. The use of microphones and personnel computer (PC) cameras are affordable and easily available. The microphones and cameras enable the computer to hear and see, and to use this information further to act. To achieve effective human computer interaction, computer should be able to interact naturally with the user, similar to human to human interaction. Human beings possess and express emotions in everyday interactions with others. Emotions are often reflected on the face, in the form of facial expressions. Facial expressions are commonly used in day to day life for human to human communication, as one smiles to show greeting,

frowns when confused. According to Mehrabian, when people are speaking, 55% of communication happens via expression whereas, only 7% happens via spoken words and vocal part contributes for 38 percent [4]. This implies that the facial expressions play a major role in human to human communication. For a Human Computer Interface, expression is a great potential input. Todays computers are emotionally challenged because they neither recognize the user's emotions nor possess emotions of their own. Psychologists and engineers have tried to analyze facial expressions in an attempt to understand and categorize these expressions. This knowledge is used to teach computers to recognize facial expressions from video images acquired from built in cameras. Computers in the future may be able to offer advice in response to the mood of the users. Various applications using automatic facial expression analysis can be developed in the near future, creating further interest in doing research in different areas, including face image compression and synthetic face animation, image understanding, video indexing, psychological studies, robotics as well as virtual reality. Ekman and Friesen [5] has defined six basic emotions. These are happiness, sadness, fear, disgust, surprise and anger. These expressions in addition to neutral expression are considered to be universal by all researchers.

1.1 Motivation

Since last decade there has been a tremendous development in ubiquitous computing environment, where powerful and low cost computing systems are being integrated into medical instruments, cars, mobile phones, and almost every aspect of our lives. This has developed interest, in automatic processing of digital images and videos in a number of applications, including surveillance, biometric authentication, and human computer interaction, etc. To build a friendlier Human Computer Interface, facial expression recognition is essential. Facial expression is an efficient way of communication, as it is natural, non intrusive, and conveys more information than spoken words and voice tone. Facial expression recognition has been a research interest for scientists from several different tracks, i.e., computer science, engineering, psychology, and neuroscience. Their studies focus not only on improving computer interfaces, but also on improving the actions that the computer takes, based on feedback from the user. The best known facial expression analyzer is a human visual system. In our day to day life, we detect any facial pattern by simply inspecting the scene, irrespective of the distance, orientation or lightning conditions. It is a difficult task to incorporate all these features of the human visual system into an automated system. In an ideal automated facial expression analyzer, all of the stages of the facial expression analysis

namely face detection, facial feature extraction and classification should be performed automatically. The main aim is to achieve a real time performance. The system should be able to recognize facial expressions irrespective of changes in lightning conditions and distractions like glasses, changes in hair style, and facial hair like moustache, beard and grown together eyebrows. In a classroom teaching, many times teacher is not able to get proper and real feedback. In such a cases, a facial expression recognition system can play a great role. By analyzing facial expressions of students, teacher can get real feedback. Not only that, level of difficulty of posed questions could also be obtained from expressions. Facial expression estimated from real images can be used to animate synthetic characters. This technique is useful in video telephony, where bandwidth is limited. Instead of transmitting the complete video, we can just transmit the facial expression sequence, using which the original video can be reconstructed at the receiving end. In patient monitoring system facial expression recognition can be used for pain assessment. Facial expression of a customer can be collected by service providers as implicit user feedback to improve their service. Compared to a conventional questionnaire based method, this will be more reliable, quick and has virtually no cost.

1.2 Scope of the work

Recognizing facial expression with a high accuracy is a very difficult task due to large variety of faces and facial expressions. Normally, a human can detect face and interpret facial expressions without much effort and delay. But development of an automated system that perform this task is difficult. There are several problems such as detection of a face in an image, extraction of facial expression information and classification of the expression. Getting 100% accurate detection is still an ideal task for an algorithm or computer system. Most of the proposed methods are not real time, they are applicable only for frontal faces, tilted faces are not allowed. Most of the methods are not fully automatic, they need manual fitting or labeling at the initial stage. Most of the developed systems have some limitations on the settings of their use, which make them unsuitable for real life applications. The limitations in automatic expression recognition are to a large extent the result of high variability that can be found in images that contains a face. We will see an extremely large variety in types of faces, resolution, pose, lighting conditions, and orientation. In order to analyze all these variations in images correctly, it is desirable to design a real time facial expression recognition system.

Now a days more and more automation is being introduced everywhere. In

many of these automation systems computer vision play a great role. Normally these computer vision systems are implemented using a typical PC. Designing the system using a PC is easy. The technology used in PC is widely known and is used from many years. But from the performance point of view, the PC is not a right choice. In Central Processing Unit (CPU) of a PC, operations are performed in a sequential manner. With the increase in the complexity of a vision system, the frame rate at which the PC is capable of processing images in real time decreases. This increase in processing time is because of the sequential operation mode of the CPU. Many of the demanding tasks of a vision system are trivial low level algorithms, where the same instructions are applied to each pixel in the frame. This thing motivated us to implement our facial expression recognition (FER) algorithm efficiently on a parallel structure such as the Field Programmable Gate Array (FPGA).

1.3 Our Contribution

Algorithm 1 : Facial expression recognition using Active shape model and Sparse representation classifier (for image sequence)

Limitations of existing systems:

- The methods proposed in literature are not real time, they are applicable only for frontal faces, tilted faces are not allowed.
- The methods are not fully automatic, they need manual fitting or labeling at the initial stage.
- While detecting facial expressions in image sequences, most of the methods assumed that first frame corresponds to a neutral facial expression.
- The majority of the methods are person dependents and can not handle spontaneous facial expressions.
- Most of the methods are using training and testing images from same reference database.
- Almost all the methods are working on only Six basic facial expressions Anger, Sad, smile, fear, disgust and surprise.
- The methods use either of Bayesian, HMM, or SVM classifiers.

Solutions through our algorithm:

- Our method can be applied to tilted faces as well.

- Our method employs fully automated fitting and does not require manual fitting or labeling at the initial stage.

- In our proposed method, for image sequences, it is not mandatory that first frame correspond to neutral facial expression.

- Our method is person independent and can handle spontaneous facial expressions.

- We have used one reference data set for training and another one for testing. Then we perform a hardware implementation of this algorithm on FPGA.

- We have created our own database of nine facial expressions based on navras "Bharatnatyam" a classical dance style of south India. We have tested our algorithm on this database.

- We have explored different classifiers such as Bayesian, ANN, SVM. We have designed Sparse representation classifier with multiple kernels and proved that it outperforms the other classifier.

Algorithm 2: Facial expression recognition using mathematical modeling of the brain functioning (for still image)

Problem formulation : The limitations in automatic expression recognition are to a large extent the result of high variability that can be found in images that contains a face. We will see an extremely large variety in types of faces, resolution, pose, lighting conditions, and orientation. In order to address this problem, it becomes necessary to follow how a human brain detects facial expression. What changes occur inside a brain, when the human being watches a scene or a face?

Proposed Approach: We studied the functioning of the brain. We studied how a brain understands an image. While going through literature, we also studied the following paper

K. Obermayer and H. Ritter, "A model for the development of the spatial structure of retinotopic maps and orientation columns," IEICE Transactions Fundamentals, vol. 75, pp. 537–545, May 1992. Based on this, we propose a novel way of recognizing facial emotion expressions by using mathematical modeling of the brain functioning, i.e. finding neuronal structures that takes place in the brain, while it learns to recognize various facial expressions. To implement this we have designed our own algorithm using Gabor filter and Self organizing feature map technique. For various expressions, different network folding patterns are obtained. During the training phase unique folding patterns or structures are generated for various expressions. During the testing phase, pattern having the

maximum similarity with the stored pattern will be the winner. And the recognized expression will be the expression corresponding to the stored pattern. A multilayer perceptron (ANN) has been used for the classification. This algorithm is successfully applied to still images.

1.4 Outline of the thesis

We have organized the thesis in eight Chapters. In Chapter 2, we present the state of the art of facial expression recognition. It would really be difficult to consider and to cover all the approaches published in a thesis, still we we have attempted and considered about more than 20 papers from 1996 to 2014, which we realized as important and distinct from each other. We have presented their methods in brief. The papers are presented in chronological order starting from 1996. Comparative study is also summarized in a tabular form for quick access & comparison. In Chapter 3, we propose an approach of facial expression recognition in image sequence based on geometrical deformation of feature vectors. Method of feature point extraction is described. Wire frame model fitting and tracking using Active Appearance Model (AAM) is described. Results of facial feature point detection and wire frame fitting are presented. Chapter 4 elaborates classification techniques. Bayesian classifier and different methods of Support Vector Machine are implemented. Confusion matrices and accuracy of each methos is presented. A comparison of these methods is presented. Chapter 5 describes hardware implementation of our proposed algorithm onto Xilinx FPGA. Hardware implementation approach for wire frame fitting, tracking and multiclasss SVM is described in detail. Chapter 6 describes facial expression recognition using sparse representation classifier. We have designed multiple kernel sparse representation classifier. We apply this technique for recognition of nine facial expressions of classical dance style of south India "Bharatnatyam". Chapter 7 describes the mathematical model, which we have developed for facial expression recognition in still images. Model is based on neural computations underlying in brain. In Chapter 8, we summarize the contributions of this thesis and highlight the focus of future research.

Chapter 2

Facial expression recognition system : State of the art

2.1 Introduction

The modern day science of automatic facial expression recognition has a direct relationship with the work done by Charles Darwin in 1872 on facial expression analysis. Darwin wrote a report that established the general principles of expression and the means of expressions in both animals and humans. In 1970s psychologist Paul Ekman [5] worked on emotions and facial expressions. His work has a large influence on the development of modern day automatic facial expression recognizers. According to him, basic facial expressions are Happy, Sad, Anger, Fear, Surprise, Disgust. In 1978, Suwa et al. [6] has started working on automatic recognition of facial expressions which was the first reported step towards FER. They designed a system for facial expression recognition from an image sequence, by using 20 tracking points. Research on the analysis of automatic facial expression was not actively pursued till the early 1990s [6]. The reason for this may be, the automatic recognition of facial expressions requires robust face detection and face tracking systems. In 1990s, computing power became available at a lower cost, that led to the development of robust face detection and face tracking algorithms. In [6], Iyenger presented a survey of facial expression recognition systems reported till year to 1992. In 1990s, Human-Computer Interaction and Affective Computing were the emerging fields. People working in these fields realized that the computers will unreceptive and remain cold to the users' emotional state without automatic expression recognition systems. This created an interest in the development of automatic FER systems and then this field become very active.

Detailed surveys of facial expression recognition are available in [4] and [1] that focus on in depth study of the work published between years 1990 and 2001. In

this chapter, we present a review and discussion of various available approaches for facial expression recognition. Although, it would be very difficult to comprehensively cover all the published work in a thesis; however, we have considered more than 20 papers between years 1996 and 2014, which we realized to be important and were relatively distinct from each other. The various works reported are discussed in chronological order, starting from 1996. We have summarized these methods in tabular form also. We also present a review on hardware implementation of support vector machine, which is used as a classifier in many facial expression recognition systems to classify facial expressions.

2.2 Applications

For a human beings, facial expression is one of the most powerful, immediate and natural means for non-verbal communication i.e. to communicate their emotions and intentions. Facial expression contains a great deal of information, hence, the need to automatically extract this information has been felt for long. Automatic facial expression recognition systems find important applications in many areas such as human computer interaction, and data driven animation. Various applications using automatic facial expression analysis can be designed in the near future, developing further interest in doing research in different areas, including facial image compression and synthetic face animation, image understanding, video indexing, robotics, psychological studies, as well as virtual reality. Many applications, such as video conferencing, neurology, pain assessment, customer satisfaction studies for broadcast and web services, lie detection, intelligent environments, clinical psychology, surveillance and multimodal human computer interface require efficient facial expression recognition in order to achieve the desired results. Therefore, the scope of facial expression recognition is constantly growing in the above mentioned application areas. It can be widely applied as a part of an effort to develop basic techniques for space teleconferencing in which the machine can recognize human facial expressions and then reproduce the human facial images with realistic expressions in a remote location. Computers in the future will be able to offer advice in response to the mood of the users. The reaction of the people in the test panels could be automatically monitored and forensic investigation could benefit from a method to automatically detect signs of extreme emotions, fear or aggression as an early warning system.

In humanoid robots, as robot interact more and more with humans, they should understand the human moods and emotions. To create such intelligent interface between the man and the machine, expression recognition system is required. Other than robotics and human computer interaction, expression recognition sys-

tems find uses in other domains like Animations, Psychiatry, Behavioral Science, Video Games, Automobile Safety, Telecommunications, and Educational Software, etc.

Facial expressions of customers can be collected by service providers as a user feedback to improve their service. Compared to a conventional questionnaire based method, it will be more reliable and has very low effective cost. Facial expression estimated from real images can be used to animate synthetic characters. This technique is useful in video telephony, where bandwidth is limited. Instead of transmitting the video, we can just send the facial expression sequence, using which the original video can be reconstructed. It can be used in clinical psychology, psychiatry and neurology. It can also be used in pain assessment, image and video database management and searching, lie detection and so on

To develop an animated character, which mirrors the users expressions, M. Bartlett et al [7] have used their face expression recognition system. Another application called the 'EmotiChat' has been developed by Anderson and McOwen [8]. It was a chat room application, used for chatting by user. The facial expression recognition system automatically inserts emoticons based on the user's facial expressions.

2.3 Facial action coding system

Facial Action Coding is a technique used for facial expression recognition. It is a muscle based approach. It identifies various facial muscles that cause changes in facial behaviors. These changes in the face and the underlying muscles are called Action Units (AUs). The facial action coding system (FACS) [5] is made up of combinations of such several action units, such as the action of raising the Inner Brow is indicated by AU 1, while the action of raising the Outer Brow is indicated by AU 2, and AU 26 represents the action of dropping the Jaw, and so on. Following are some AUs which are not caused by facial muscles,

1. AU 19 'Tongue Out'
2. AU 33 'Cheek Blow'
3. AU 66 'Cross-Eye', and so on

AUs are of two types, additive and non additive. It is said to be additive, if their appearance is independent. If their appearance depends on other AUs or if they modify other's appearance, then they are said to be non additive. Using AUs, representation of facial expressions becomes an easy job. The combination of one or more additive or non additive AUs, represents facial expressions, e.g.

combination of AUs 1, 2 and 26 represents fear, combination of AUs 1, 2, 5 and 27 represents surprise etc.

2.4 Facial animation parameters

In 1990s and prior to that, every animation system was having their own set of parameters. There was no common standard. Instead of putting the efforts to choose the best set of parameters, more focus was given on facial movements caused by the parameters, which made the systems unusable across domains. To address these issues, the Moving Pictures Experts Group (MPEG) [2] has introduced the Facial Animation (FA) specifications in the MPEG-4 standard. In the last few years, researchers have started using these standard to model the facial expressions. The MPEG-4 standard provides Facial Animation Parameters (FAPs), and focuses mainly on facial expression synthesis and animation [9]. This standard also defines 84 key feature points (FPs) on a neutral face. To understand and recognize facial movements and to animate the faces, the movement of the FPs can be used

2.5 Generic facial expression analysis framework

A large variety of faces due to different age, ethnicity, gender, facial hair make the facial expression analysis more difficult. Further, pose and lighting conditions make it more crucial. Generic facial expression analysis framework is shown in Figure 2.1.

1. Face Acquisition: It is the first and important step in facial expression recognition system. In face acquisition stage, an automatic face detector is used to locate faces in a complex scene. Some method needs the exact location of the face while some needs coarse location of the face. Some methods, processes whole face while some need only facial features [1]. Face detector must be able to detect faces from any angle, frontal as well as profile view.

2. Face Normalization: The appearance of facial expression depends on the distance and angle at which a given face is being observed. So face analysis becomes complicated due to face appearance changes caused by pose, scale and illumination. Therefore, it is good idea to normalize it [1]. It is similar to signal conditioning. It is necessary for noise removal, and normalization against the variation of pixel position or brightness, together with segmentation, location, or tracking of the face or its parts. Expression representation can be sensitive to translation, scaling, and rotation of the head in an image.

To combine the effect of these unwanted transformations, the facial image is geometrically standardized prior to classification.

3. Face Segmentation: It is used to separate faces of interest from the background. It also allows to isolate transient and intransient features within a face. Intransient features are eyes, eyebrows and mouth. Transient features are different kind of bulges and wrinkles [1].

4. Deformation extraction: These methods have to rely on neutral face images or face model to extract useful facial features that are not caused by wrinkles due to old age. It can be applied to single image as well as to the image sequence. Without relying on extensive knowledge about the object of interest, image based methods extract features from images. These methods are fast and simple. When there are many different views of the same object, then these methods become unreliable. The facial structure can also be described with 2D or 3D face models. They allows to model facial features and faces based on their appearance, without attempting to recover the geometry of the scene. There are two types of 3D models, namely muscle and motion models. But heavy computations are required for mapping 3D models. In addition, accurate head and face models have to be constructed manually, which is a tedious undertaking [1]. The techniques used for image based deformation extraction are Neural network, Gabor wavelets, Principal component analysis with neural network, and intensity profiles. The techniques used for Model based deformation extraction are Active appearance model, Point distribution model, Labeled graphs, and Geometric face model

5. Motion extraction: These approaches directly focus on facial changes occurring due to facial expressions. These methods do not need neutral face images. Difference images are mostly created by subtracting a given facial image from a previously registered reference image, that contains a neutral face of the same subject. In comparison to optical flow approaches, only differences of image intensities are extracted, no flow direction are extracted. In addition, accurate face normalization procedures are necessary in order to align reference faces onto the test faces. In feature point tracking, motion estimates are obtained only for a selected set of prominent features such as intransient facial features. The automatic initialization of feature points is difficult and is often done manually. Region-based or whole face based dense optical flow is used in order to estimate the activity of facial muscles. For each muscle, a window in the face image is defined as well as an axis along which each muscle expands and contracts. Dense optical flow mo-

tion is quantified into eight directions and allowed for a coarse estimation of muscle activity. In pattern tracking, it is possible to determine facial actions by measuring deformation in areas, where underlying muscles interact. However, these are mostly skin regions with relatively poor texture. Here, highlighting becomes necessary and can be done by either affixing colored plastic dots to predefined locations on the subject's face or by applying color to the salient facial features and skin. In motion models, 3D face models are used to specify shape, texture and motion [1]. The techniques used for motion extraction are dense optical flow, 3D motion models, 3D deformable model, parametric motion model, feature point tracking, dot markers, highlighted facial features and region based difference images.

6. Facial feature representation: After extracting the features they are in the form of either muscle based, model parameters or component projection. These features are then given to the classifier for classification.

7. Recognition: Traditional approaches for modeling characteristics of facial motion and deformation have relied on hand-crafted rules and symbolic mid-level representations for emotional states, which have been introduced by computer scientists in the course of their investigations on facial expressions. To map these symbolic representations into emotions, human expertise is necessary. But facial signals consist of numerous distinct expressions, each with specific facial action intensity evolutions. Individual realizations of facial expressions differ only in subtle ways. So the task of manually creating facial expression classes becomes difficult [1]. Designing a suitable classifier to classify extracted facial features is a difficult task. Mostly used classifiers are Neural network, Hidden Markov model, support vector machines etc.

8. Interpretation: Many automatic facial expression analysis systems attempt to directly interpret observed facial expressions and mostly in terms of basic emotions. Only a few systems use rules or facial expression dictionaries in order to translate coded facial actions into emotion categories. The latter approaches have not only the advantage of accurately describing facial expressions without resorting to interpretation, but allow also to animate synthetic faces, e.g. within the FACS coding framework. This is of interest, as animated synthetic faces make a direct inspection of automatically recognized facial expressions possible [1].

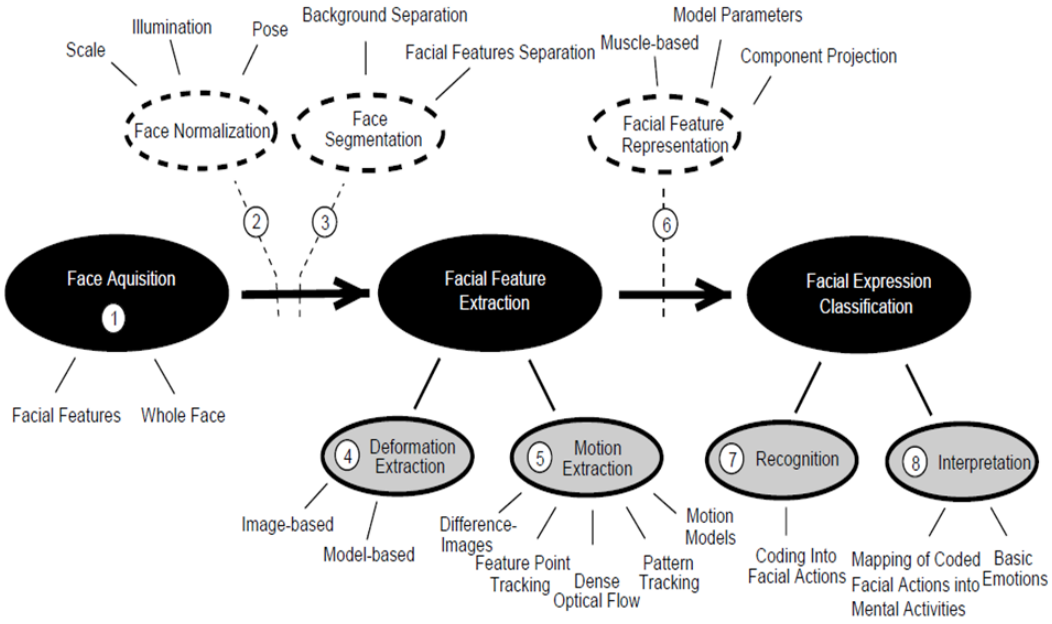

Figure 2.1: Generic facial expression analysis framework [1]

2.5.1 Characterstics of Ideal Facial expression recognition system

A good facial expression recognition system must be

- fully automatic and able to work with videos and images
- real time and recognize spontaneous expressions
- robust to lighting variations and must work with occlusions
- unobtrusive and person independent
- able to work on faces with different cultures as well as different skin colors.
- able to recognize expressions of a person of any age.
- able to recognize expressions with facial hair, glasses, and makeup etc.
- able to recognize expressions from frontal, profile as well as other intermediate angles.

2.6 Related work

The approaches of facial expression recognition can be broadly classified in two classes

1. template based approaches
2. model based approaches

In template based systems, a template is used, which is a pixel image or a feature vector that is obtained after processing the face image. In feature based system, major face components or feature points are detected from the face image. The distances between feature points and the relative sizes of the major face components are computed, which will form a feature vector. Thereafter, low dimensional representation of feature vectors is obtained using principal component analysis or multilayer neural networks. The feature points can also form a geometric graph representation of the faces. Feature based techniques are computationally more expensive as compared to template based techniques. However, they are more robust to variations in size, scale, location, and head orientation of the face in an image.

Facial expression recognition problem in image sequences can be divided into three sub problems.

- Face detection- before a facial expression can be analyzed, the face must be detected in a scene. Different methods used for face detection are eigenface, Canny edge detector, brightness distribution, skin color detection etc [4].
- Feature extraction and tracking- to develop a mechanism for the extraction of the facial expression information from the observed facial image sequence and then track these features in subsequent frames. Prominent facial features of the face constitute eyebrows, eyes, nose and mouth. For feature extraction, researchers have explored many techniques like labeled graph, point distribution model, brightness distribution, optical flow computation, potential net fitting, Gabor wavelets. While for tracking they have used Kalman filters, active appearance algorithm and optical flow computation methods [4].
- Classification- to develop a mechanism to classify facial expressions into one of the basic facial expressions. Different methods used by researchers for classifying facial expressions are hidden Markov model, neural networks, principal component analysis, linear discriminant analysis, and support vector machines [4].

We have attempted to cover as large and varied group of reported works as we can, and have considered some papers between years 1996 and 2014. The papers are presented in chronological order, starting from 1996, however, we do not claim the coverage as comprehensive.

Y. Yacoob and L. S. Davis [10] proposed an approach, which is based on a statistical characterization of the motion patterns in specified regions of the face. They have developed a region tracker for rectangles enclosing the face features. Each rectangle encloses one feature of interest, so the flow computation within the region was not contaminated by the motions of other facial features. To simplify the modeling of the eyebrows, they define the rectangles to include the eyes, and then subtract the rectangle of the eye from the combined rectangle. The tracking algorithm integrates spatial and temporal information at each frame. In order to enhance the tracking, the statistics of the motion directions within a rectangle are used to verify a translation of rectangles upward and downward and scaling of the rectangles. With the help of universal expression descriptions proposed by Ekman and Friesen [5], and motion patterns of expression proposed by Bassili [4], they prepare a dictionary of facial feature actions (motion based feature description of facial actions). The dictionary is divided into components, basic actions of these components, and motion cues. The components are defined qualitatively and relative to the rectangles surrounding the facial regions. Using component's visible deformations, and using optical flow within these regions, the basic actions are determined. They designed a rule based system that combines certain expression descriptions. They have proposed rules for identifying the onsets of the beginning and the ending of each facial expression, e.g. for Anger, beginning is inward lowering brows and mouth compaction and the ending is outward raising brows and mouth expansion. These rules apply to the mid level representation to create a complete temporal map describing the evolving facial expression.

Black and Yacoob [11] have used local parametrized models of image motion for facial expression analysis. The location of the face, eyes, eyebrows, and mouth are assumed to be known. They estimate the rigid motion of the face region between two frames using a planar motion model. This estimation was performed using a robust statistical approach to cope with violations of the rigid plane assumption. The motion of the face was used to register the images via warping and subsequently the relative motion of the feature regions was estimated in the coordinate frame of the face using exactly the same robust estimation procedure. The motion estimates of the face and features were used to predict their locations in the next frame and the process was repeated. The estimated motion parameters provide a simple abstraction of the underlying facial motions and can be used to classify the type of rigid head motion and the facial expression. The motion parameters,

e.g. translation and divergence were used to derive the mid level predicates that describe the motion of the facial features. For each of the six basic emotional expressions, they developed a model represented by a set of rules for detecting the beginning and ending of an expression. The rules were applied to the predicates of the mid level representation [4].

Kimura and Yachida [12] made use of integral projection method proposed in [13] to detect facial features. They normalize the input image using the center of the eyes and the center of the mouth. A potential net is then fitted on the normalized image to model the face and its movement. To do that, they first compute edge image by applying differential filter. Then, in order to extract the external force, they apply the Gaussian filter. The filtered image is called potential field and an elastic net model is placed over it. They fit a potential net to each frame of the facial image sequence under consideration. The pattern of the deformed net is compared to the pattern extracted from an expressionless face (usually the first frame of the sequence), and the variation in the position of the net nodes is used for further processing. They built an emotion space by applying PCA on six image sequences carrying three expressions anger, happiness, and surprise shown by a single person gradually, from expressionless to a maximum intensity of expression. The eigenspace spanned by the first three principal components has been used as the emotion space, onto which an input image is projected for classification [4].

Essa and Pentland [14] made use of eigenspace method proposed by Pentland et al. [15] to detect faces in an image sequence. They have applied principal component analysis (PCA) on a sample of 128 facial images and created face space in order to detect facial features. They calculate the distance of the observed image from the face space to detect the presence of face. To detect the location of the facial feature in a given image, the distance of each feature image from the relevant feature space was computed using an FFT. They normalize the input image using the extracted position of facial feature. A two-dimensional (2D) spatio-temporal motion energy representation of facial motion between two subsequent normalized frames was used as a dynamic face model. They employ an optical flow computation method proposed by Simoncelli [16], which uses a multiscale coarse to fine Kalman filter to compute motion estimates. The spatio-temporal templates were generated for six different expressions, and two facial actions (smile and raised eyebrows) and four emotional expressions (surprise, sadness, anger, and disgust) by learning ideal 2D motion views for each expression category. The Euclidean norm of the difference between the motion energy template and the observed image motion energy was used as a metric for measuring similarity (dissimilarity) [4]. However, the method is useful only for frontal view face image sequences.

Cohn et al. [17] proposed a method in which key feature points were manually marked with a computer mouse around facial landmarks on the first frame of the image sequence. Each point was the center of a 13×13 flow window that includes horizontal and vertical flows. A hierarchical optical flow method proposed by Lucas and Kanade [18] was used to automatically track feature points in the image sequence. The displacement of each feature point was calculated by subtracting its normalized position in the first frame from its current normalized position. The resulting flow vectors- 6 horizontal and vertical dimensions in the brow region, 8 horizontal and vertical dimensions in the eye region, 6 horizontal and vertical dimensions in the nose region, and 10 horizontal and vertical dimensions in the mouth region are concatenated to produce a 12 dimensional displacement vector in the brow region, a 16-dimensional displacement vector in the eye region, a 12 dimensional displacement vector in the nose region, and a 20 dimensional vector in the mouth region [17]. Separate group variance-covariance matrices were used for classification. They used two discriminant functions for three facial actions of the eyebrow region, two discriminant functions for three facial actions of the eye region, and five discriminant functions for nine facial actions of the nose and mouth region. [4].

Wang et al. [19] utilized 19 facial feature points (FFPs) - seven FFPs to preserve the local topology and 12 FFPs for facial expression recognition. The FFPs are treated as nodes of a labeled graph that are interconnected with links representing the Euclidean distance between the nodes. The initial location of the FFPs in the first frame of an input image sequence is assumed to be known. The FFPs are tracked in the rest of the frames. The correspondence between the FFPs tracked in two consecutive frames is treated as a labeled graph matching problem proposed by Buhmann et al.[20]. For three emotion categories viz. anger, happy and surprise, they use 12 B-spline curves corresponding to facial feature points, one each for one FFP, in order to construct the expression model. Each curve gives the relationship between expression change and the displacement of the corresponding FFP. The expression is determined by the minimal distance between the actual FFPs and FFPs of model. The degree of expression change is determined based on the displacement of the FFPs in the consecutive frames [4].

Tian et al. [21] developed an Automatic Face Analysis (AFA) system to analyze facial expressions based on both permanent facial features (brows, eyes, mouth) and transient facial features (deepening of facial furrows) in a nearly frontal-view face image sequence. This system recognizes fine-grained changes in facial expression into action units (AUs) of the Facial Action Coding System (FACS). They propose Multistate face and facial component models for tracking and modeling the various facial features, including eyes, lips, brows, furrows and cheeks. During

tracking, they have extracted detailed parametric descriptions of the facial features. With these parameters as the inputs, a group of action units (six upper face AUs and 10 lower face AUs) has been recognized whether they occur alone or in combinations. Instead of one HMM for each AU or AU combination, the current system employs two Artificial Neural Networks (one for the upper face and one for the lower face) for AU recognition. It recognizes 16 of the 30 AUs that have a specific anatomic basis and occur frequently in emotion and paralinguistic communication. They use Cohn Kanade and Ekman Hager database. They have achieved average recognition rates of 96.4 percent (95.4 percent if neutral expressions are excluded) for upper face AUs and 96.7 percent (95.6 percent with neutral expressions excluded) for lower face AUs.

Cohen et al. [22] used Piecewise Bezier Volume Deformation (PBVD) tracker proposed by Tao and Huang [23]. This face tracker uses a model based approach, where an explicit 3D wire frame model of the face was constructed. They have selected interactively landmark facial features such as the eye corners and mouth corners on the first frame of the image sequence. Then they warp the generic face model to fit the selected facial features. The face model consists of 16 surface patches embedded in Bezier volumes. The surface patches defined this way are continuous and smooth. By changing the locations of the control points in the Bezier volume, the shape of the mesh can be changed. Head motion and local deformations of the facial features such as the eyebrows, eyelids, and mouth are tracked, once the model fits.. First they measure 2D image motions using template matching between frames at different resolutions. For more robust tracking, they have used Image templates from the previous frame and from the very first frame. Then they model measured 2D image motions as projections of the true 3D motions onto the image plane. They represent recovered motions in terms of magnitudes of some predefined motion of various facial features. Each feature motion corresponds to a simple deformation on the face. These motion vectors are referred as Motion Units (MU's). They are similar but not equivalent to Ekman's AU's. They are numeric in nature, and representing not only the activation of a facial region, but also the direction and intensity of the motion. The MU's are used as the basic features for the classification. Bayesian classifier is used for classification.

M. Bartlett et al. [7] have designed a system, in which face is detected using Viola Jones algorithm. They have rescaled automatically located faces to 48×48 pixels. They made a comparison at double resolution (96×96) also. The typical distance, they have considered between the centers of the eyes is roughly 24 pixels. They convert the images into a Gabor magnitude representation, using a bank of Gabor filters at 8 orientations and 5 spatial frequencies. For expression

classification they use SVM and Adaboost. There were $48 \times 48 \times 40 = 92160$ possible features. A subset of these filters was chosen using Adaboost. They used Cohn Kanade database.

Michel P. et al. [24] use a real time facial feature tracker to deal with the problems of face localization and feature extraction in spontaneous expressions. The tracker extracts the position of 22 facial features from the video stream. The tracker uses a face template to initially locate the position of the 22 facial features of face model in the video stream and uses a filter to track their position over subsequent frames. Then they calculate displacements for each feature between a neutral and a representative frame of an expression. These are used together with the label of the expression as input to the training stage of an SVM classifier. The trained SVM model is subsequently used to classify unseen feature displacements in real time. They used Cohn Kanade database. For person dependent classification they got 83.7% accuracy, for testing they used only 10 to 12 samples. For Person-dependent training and test data supplied by six users during ad-hoc interaction they got 60.7% accuracy.

M. Valstar et al. [25] have proposed a system that performs action units (AU) recognition using temporal templates as input data. Temporal templates have also been used by Bobick and Davis [26]. These templates are 2D images constructed from image sequences, effectively reducing a 3D spatio temporal space to a 2D representation. To achieve this they first select 9 facial points from the first frame of the image sequence manually. These points are then tracked in all subsequent frames using a condensation based template tracking technique proposed by Isard and Blake [27]. They have used Neural Network as a classifier. They have used just a simple k-level neural network (kNN) based learning machine to classify an input image sequence into one of m facial expression classes, each of which corresponds either to an individual AU or to an AU-combination. The employed algorithm is straightforward for a test sample. It uses a distance metric to compute which k-labeled training samples are nearest to the sample in question and then casts a majority vote on the labels of the nearest neighbors to decide the class of the test sample [25].

Pantic and Patras [28] designed a system for recognition of 27 AUs and their temporal segments in input frontal-view face videos. They start by initializing 20 fiducial points on the first frame of the input face image sequence. They use particle filtering to track these 20 points automatically for the rest of the sequence. Depending on the changes in the position of the fiducial points, they measure changes in facial expression. These changes are then transformed first into a set of mid-level parameters for AU recognition. Then a rule-based method encodes temporal segments (onset, apex, offset) of 27 AUs occurring alone or

in a combination in the input face videos. They used Cohn-Kanade and MMI facial expression database. They achieved an average recognition rate of 90% for encoding of 27 AU codes and their combinations in 135 test samples.

Zheng et al. [29] address the facial expression recognition problem using kernel canonical correlation analysis (KCCA). They locate 34 points manually from each facial image as the landmark locations. Then they convert these geometric locations into a labeled graph vector using Gabor wavelet transformation method to represent the facial image. A semantic expression vector consisting of the semantic ratings of each facial image have been used as the semantic expression representation. For learning the correlation between the LG vector and the semantic expression vector, they used KCCA. According to this correlation, they estimate the associated semantic expression vector of a given test image and then perform the expression classification according to this semantic expression vector. Using Semantic Information On JAFFE database with Leave one image out (LOIO) cross validation they got 85.79% accuracy while with Leave one subject out (LOSO) cross validation they got 74.32% accuracy. On Ekman's database, it is 81.25%. Using Class Label Information On JAFFE database with LOIO it is 98.36%, and with LOSO it is 77.05%, On Ekman's database, it is 78.13%.

Irene Kotsia and Pitas [30] have proposed a method which is based on mapping and tracking the facial model Candide onto the video frames. They have used Candide wire frame model. Their proposed system is semi automatic. The user has to manually place some of the Candide grid nodes on face landmarks depicted at the first frame of the image sequence. They have used a popular Kanade Lucas Tomasi tracker [31] for tracking facial features in subsequent frames. While the tracking system tracks feature points in subsequent frames, it also allows the grid to follow the evolution of the facial expression till it reaches its highest intensity. This produces the deformed Candide grid at each video frame. They have selected a subset of the Candide grid nodes, that predominantly contribute to the facial deformations described by the facial action coding system (FACS). The geometrical displacement of these nodes, i.e., the difference of coordinates of each node at the first and the last frame of the facial image sequence, have been used as an input to a support vector machine classifier. Support vector machine classifies the test sample into one of the six basic facial expressions.

Kotsia et al. [32] have performed an analysis of the effect of partial occlusion on facial expression recognition. They have classified partially occluded face images in to one of the six basic facial expressions using a method based on Gabor wavelets texture information extraction, and a supervised image decomposition method based on Discriminant Non-negative Matrix Factorization, and a shape-based method that exploits the geometrical displacement of certain facial features.

They demonstrate how partial occlusion affects the above mentioned methods in the classification of the six basic facial expressions. They have shown how partial occlusion affects human observers when recognizing facial expressions. Their results show that left/right facial region occlusion does not affect the recognition accuracy rate, indicating that both facial regions possess similar discriminate information. The results indicate that mouth occlusion, in general, causes a greater decrease in facial expression recognition than the equivalent eyes one. Mouth occlusion affects more anger, fear, happiness and sadness, while eyes occlusion the remaining disgust and surprise.

In [33] Lajevardi and Margaret have proposed a method which is fully automatic. They have used Viola Jones method and Adaboost algorithm [34] for face detection. For feature extraction they have made use of log Gabor filters. Five scales and eight orientations were used to extract features from face images. This leads to 40 filter transfer functions representing different scales and orientations. For each training image a set of 4 log-Gabor filters with the smallest value of the spectral difference was selected. This reduces feature dimensions from 40 to only 4 arrays of size 60×60 for each image. Further reduction of the feature data was achieved by down-sampling the feature vectors by the factor of 4 to vectors of length 3600 samples per image. As a classifier they have made use of Naive Bayesian (NB) Classifier.

Ligang and Dian [35] have developed a system for improving the performance of facial expression recognition by automatically capturing facial movement features in static images, based on distance features. They have obtained distances by extracting 'salient' patch-based Gabor features and then they perform patch matching operations. To start with they take the nose as the center and crop facial regions manually from database images. Then they scaled it to a resolution of 48×48 pixels. Then they got multi-resolution Gabor images by convolving eight-scale, four-orientation Gabor filters with the scaled facial regions. During the training stage, a whole set of patches was extracted by moving a series of patches with different sizes across the training Gabor images. Then they perform patch matching operation to convert the extracted patches to distance features. To capture facial movement features, they define the matching area and matching scale to increase the matching space, whereas the minimum rule was used to find the best matching feature in this space. Using Adaboost, a set of 'salient' patches was selected based on the converted distance features. At the test stage, they have performed the same patch matching operation on a new image using the 'salient' patches. The resulting distance features were fed into a multi-class support vector machine (SVM) to recognize six basic emotions. They have used JAFFE and Cohn Kanade database. They got accuracy 92.92% and 93.14% for

these two databases.

M. Valstar [36] uses Viola Jones algorithm for face detection. They perform face registration based on the location of the eyes. To detect the eyes, they have used the Open CV implementation of a Haar-cascade object detector, trained for either a left or a right eye. After the left-eye location and right-eye location are determined, they have rotated the image, so that the angle between the line connecting the eyes and the horizontal axis of the image, is 0^0. Then they scale the image to make the distance between two eyes 100 pixels and the face box is then cropped to be 200 by 200 pixels. They have extracted the local appearance descriptors subsequently from such registered images. As dense local appearance descriptors, they chose to use uniform LBPs. As a classifier, they use standard SVMs with a radial basis function kernel. They reduced the dimensionality of using PCA.

In contrast to the mainstream approaches Qiang et al [37] build a probabilistic model based on the Dynamic Bayesian Network (DBN) to capture the facial interactions at different levels. The flow of information is two way, not only bottom up, but also top down. In particular, not only the facial feature tracking can contribute to the expression/AUs recognition, but also the expression/AU recognition helps to further improve the facial feature tracking performance. With the proposed model, they have recovered all the three levels of facial activities simultaneously, through a probabilistic inference by systematically combining the measurements from multiple sources at different levels of abstraction. The proposed facial activity recognition system consists of two main stages: offline facial activity model construction and on line facial motion measurement and inference. Specifically, using training data and subjective domain knowledge, they have constructed the facial activity model, offline. During the online recognition, various computer vision techniques have been used to track the facial feature points, and to get the measurements of facial motions, i.e., AUs. They have used these measurements as evidence to infer the true states of the three level facial activities simultaneously. They use Cohn Kanade and MMI database. They got average recognition rate 87.43%.

Khan et al. [38] first detects faces using Viola Jones algorithm. Then they extract Pyramidal LBP features from the mouth region. They have used feature vector of 295 dimensions. The classification is carried out on the basis of extracted features in order to make two groups of facial expressions. First group comprises those expressions that have one perceptual salient region, i.e. happiness, sadness and surprise while the second group is composed of those expressions that have two or more perceptual salient regions i.e. anger, fear and disgust. To reduce feature extraction computational time, they made two groups of expressions. If

the stimuli are classified in the first group, then it is classified either as happiness, sadness or surprise by the Classifier using already extracted PLBP features from the mouth region. If the stimuli are classified in the second group, then the framework extracts PLBP features from the eyes region and concatenates them with the already extracted PLBP features from the mouth region, feature. Their feature vectors are of 590 dimensions. Then, they fed the concatenated feature vector to the classifier for the final classification. They used Cohn Kanade and IMM database. They achieved average recognition rate of 91%.

Fang et al. [39] detects face using Viola Jones algorithm. After locating the face, they extract the facial features. One common approach in this respect they used is to landmark key facial points (e.g.,eyes, lips, etc.) and use these to obtain the features. These landmarks can then be used to align the faces in static or dynamic data and thus eliminate the effects of scaling and rotation. By tracking these points throughout a video sequence, they capture the deformations i.e. motion features and use them for the task of expression analysis. They have used MMI database, and achieved the recognition rate of 71.56%.

Lijun Yin [40] proposed a method for detecting and tracking landmark facial features on purely geometric 3D and 4D range models. The method involves fitting a new multi-frame constrained 3D temporal deformable shape model (TDSM) to range data sequences. They consider this a temporal based deformable model as they concatenate consecutive deformable shape models into a single model driven by the appearance of facial expressions. This allows them to simultaneously fit multiple models over a sequence of time with one TDSM. They evaluate the accuracy of the tracking results by comparing the detected landmarks to the ground truth. The efficacy of the 3D feature detection and tracking over range model sequences has also been validated through an application in 3D geometric based face and expression analysis and expression sequence segmentation. They tested the method on the publicly available databases, BU-3DFE, BU-4DFE, and FRGC 2.0. They got average recognition rate of 87%.

Bartlett et al. [41] explore Gabor motion energy filters (GME) as a biologically inspired representation for dynamic facial expressions. Spatial Gabor energy filters (GE) are one of the most successful approaches to represent facial expressions in computer vision applications, including face recognition and expression analysis. It is well known that these filters approximate the response of complex cells in primary visual cortex. However these neurons are modulated by the temporal, not just spatial, properties of the visual signal. This suggests that spatio-temporal Gabor filters may provide useful representations for applications that involve video sequences. They have detected the faces by Viola and Jones detector and then these faces were normalized to 96 x 96 patches based on the location of the eyes.

They pre-processed the training and testing data into following 2 conditions, onset– the first 6 frames – low intensity expressions apex –the last 6 frames – extreme intensity expressions. Each video clip was first convolved with all the filters in a filter bank. Then the responses from all filters were concatenated into a long feature vector. Only the magnitude (energy) of the responses was used. They have aggregated the result from different time frames via statistical operators such as min, max, and mean. They have used linear SVM as a classifier. They have showed that GE outperforms GME. They have used Cohn Kanade database for training as well as testing. For onset classification they got average accuracy 78.56% and for apex classification they got 97.8% accuracy.

Bartlett et al. [42] have used the idea of facial expression for automated feedback in teaching. They have showed how automatic real time facial expression recognition can be effectively used to estimate the difficulty level, as perceived by an individual student, of a delivered lecture. On a video lecture viewing task, training on less thantwo minutes of recorded facial expression data and testing on a separate validation set, their system predicted the subjects' self-reported difficulty scores with mean accuracy of 0:42 (Pearson R) and their preferred viewing speeds with mean accuracy of 0:29.

Sebe et al.[43] proposed a system that uses geometric features to detect emotions. They have used Piecewise Bézier volume deformation tracking after manually locating a number of facial points. They experimented with a large number of machine learning techniques. They got the best result with a simple k-nearest neighbor technique that attained a 93% classification rate on the Cohn–Kanade database [44].

Sung and Kim [45] have used AAMs to track facial points in 3-D videos. They introduce Stereo Active Appearance Models (STAAM), which improves the fitting and tracking of standard AAMs by using multiple cameras to model the 3-D shape and rigid motion parameters. A layered generalized discriminant analysis classifier, which is based on linear discriminant analysis, is then used to combine the 3-D shape and registered 2-D appearance. Unfortunately, although the approach appears to be promising, it was evaluated for only three expressions, and no results on a benchmark database (such as the Cohn–Kanade or MMI Facial Expression database) were presented.

Littlewort et al. [46] have select the best set of Gabor filters using GentleBoost and train support vector machines (SVMs) to classify AU activation. Some measure of AU intensity is provided by evaluating for a test instance the distance to the separating hyperplane provided by the trained SVM. Haar like features were used in an AdaBoost classifier [44].

Koelstra et al. [47] have used an appearance-based approach that explicitly models a facial expression's temporal dynamics . In their work, they propose a method that detects AUs and their temporal phase onset, apex, and offset using free-form deformations and motion history images as appearance descriptors and hidden Markov models as machine learning technique [44].

Valstar and Pantic [48] automatically detect 20 facial points and use a facial point tracker based on particle filtering with factorized likelihoods to track this sparse set of facial points. From the tracked points, they compute both static and dynamic features., such as the distances between pairs of points or the velocity of a facial point. With this approach, they are able to detect both AU activation and the temporal phase onset, apex, and offset [44].

Simon et al. [49] use both geometric and appearance-based features and include modeling of some of the temporal dynamics of AUs in a proposed method using segment-based SVMs. Facial features are first tracked using a person-specific AAM so that the face can be registered before extracting SIFT features. They have applied Principal component analysis (PCA) to reduce the dimensionality of this descriptor. The proposed segment-based SVM method combines the output of static SVMs for multiple frames and uses structured-output learning to learn the beginning and end time of each AU. The system was evaluated for eight AUs on the M3 database (previously called RU-FACS), attaining an average of 83.75% area under the ROC curve [44].

Summary of aforesaid discussed methods is presented in tabular form in Table 2.1 through 2.5.

2.6.1 Comparison of surveyed approaches

Comparison of surveyed approaches based on the following characteristics of ideal facial expression recognition system is given in Table 2.6.

1. Automatic face detection
2. Automatic feature extraction
3. Deals with variation in lighting
4. Deals with partial occluded faces
5. Initial labelling or manual fitting required
6. Deals with tilted faces
7. Real time process

8. Recognize all six basic expressions

Table 2.1: Summary of facial expression recognition systems up to 2001

Ref	Feature extraction	Classifier	Database	Performance	Remarks / Limitations
Yacoob 1996 [10]	statistical characterization of motion pattern in specified regions of face	rule based, prepared dictionary of rules	46 sequences of 32 subjects own database	—	Only front view faces without hair and glasses allowed
Black 1997 [11]	Local parametrized model of image motion, optical algorithm	temporal consistency of the mid level predicates which describes the motion of the facial features	70 sequences of 40 subjects own database	88%	head motion and light variation allowed
Kimura 1997 [12]	Potential net fitting to normalized face image by Gaussian filter	3D emotion space (PCA)		—	Only front view faces without hair and glasses allowed
Essa 1997 [14]	Optical flow method	Spatio temporal motion energy templates	52 sequences of own database	98%	Front views, Face with hair and glasses, light variation allowed
Cohn 1998 [17]	Optical flow algorithm of Lucas Kanade	Discriminant functions	504 sequences of 100 subjects	88% eye region 83% nose and mouth region 92% brow region	Front views, Face without hair and glasses, manual labeling on first frame
Wang 1998 [19]	labeled graph fitting	Averaged Bsplines of feature trajectories	29 image sequences of own database	100% Happy 100% Surprise 85.7% Anger	Front views, Face without hair and glasses, manual labeling on first frame only 3 expressions happy, surprise, anger
Tian 2001 [21]	Optical flow, Gabor wavelets, Canny edge detection	ANN	Cohn Kanade and EKMAN Hager	Recognition of AUs (upper face) 96.4% Recognition of AUs(lower face): 96.7%	Automatic Face detection. Invariant to scaling, reduction in processing time using facial feature tracker

Table 2.2: Summary of facial expression recognition systems 2001 to 2006

Ref	Feature extraction	Classifier	Database	Performance	Remarks / Limitations
Cohen 2003 [22]	Piecewise B´ezier Volume Deformation (PBVD) tracker	Bayesian Classifier HMM	Cohn kanade database	74%	Use of semi supervised learning to work with some labeled and unlabeled data
Bartlett 2003 [7]	Gabor filter with 8 orientations and 5 spatial frequencies	SVM Adaboost	Cohn Kanade	93.3%	The system has been deployed in a variety of platforms
Michel 2003 [24]	neutral and peak frame feature displacements (Eu-clidean distance measure)	SVM	Cohn Kanade	Person independent: 71.8% Person dependent: 87.5%	Real-time system. Does not require any preprocessing.
M. Valstar 2004 [25]	AU recognition using temporal templates	Neural networks	Cohn kanade database	76.2%	Front views, Face without hair and glasses
Pantic 2005 [28]	20 facial fiducial points tracking	Temporal Rules	MMI and Cohn-Kanade	90%	27 AUs Recognition, handles occlusions like facial hair and glasses, gives a better performance than the AFA system
Zheng 2006 [29]	Labeled Graph (LG) Gabor wavelet KCCA	The correlation is used to estimate semantic expression vector which is given for classification	JAFFE Ekman	JAFFE DB: LOIO 85.79%, LOSO 74.32%, On Ekman's DB: 81.25%	Recognition of facial expression using KCCA Gram matrix singularity problem is solved using KCCA algorithm

Table 2.3: Summary of facial expression recognition 2007 onwards

Ref	Feature extraction	Classifier	Database	Performance	Remarks / Limitations
Kotsia [2007] [30]	Geometric displacemet of wire frame model, Pyramidal Kanade Lucas Tomasi tracker	Multi class SVM	sequences from Cohn Kanade database	99.7%	Frontal views, face without glass allowed Manual fitting of model on first frame is necessary
Seyed [2008] [33]	Log Gabor filters with gaussian transfer functions	Naive Bayesian Classifier	172 sequences of 100 subjects from Cohn Kanade database	68.9%	Only front view faces without hair and glasses allowed
Kotsia 2008 [32]	Gabor features, and extraction of Geometric displacement vectors tracker	SVM	Cohn Kanade JAFFE	Using JAFFE: with Gabor: 88.1%, Using Cohn-Kanade: with Gabor: 91.6%	recognition of expressions with occlusions.
Liga ng 2011 [35]	Distance features obtained by salient patch based gabor features	SVM	JAFFE Cohn Kanade	92.92% 93.14%	patch-based Gabor features show a better performance over point-based Gabor features
M. Valstar 2012 [36]	local appearance descriptors are subsequently extracted from registered images. As dense local appearance descriptors, they chose to use uniform LBPs., PCA	SVM	MMI	80%	AU and facial expression detection, describes the first facial expression recognition challenge, organized under the name of FERA 2011

Table 2.4: Summary of facial expression recognition 2012 onwards

Ref	Feature extraction	Classifier	Database	Performance	Remarks / Limitations
Qiang 2013 [37]	probabilistic model based on the Dynamic Bayesian Network (DBN) to capture the facial interactions at different levels	Facial activity model	Cohn Kanade MMI	87.43%	The flow of information is two way, not only bottom-up, but also top-down. Not only the facial feature tracking can contribute to the expression AUs recognition, but also the expression/AU recognition helps to further improve the facial feature tracking performance.
Khan 2013 [38]	Extraction of pyramidal LBP features	SVM	Cohn Kanade IMM	91%	robust for low resolut ion images, spontaneous expressions and generalizes well on unseen data
Fang 2014 [39]	Dynamic parameter space, simple descriptors, PCA, auto regressive model	FRNN VQNN SMO-SVM	MMI	71.56%	the primary concern is whether features extracted from dynamic sequences can improve the facial expression recognition.

Table 2.5: Summary of facial expression recognition 2012 onwards

Ref	Feature extraction	Classifier	Database	Performance	Remarks / Limitations
Lijun Yin 2013 [40]	3D and 4D range models, Temporal deformabel	Given the fit points of the mesh model, we compare them with the instances of our TDSM. The smallest D value from these comparisons is used as a measure for the classification	BU-3DFE BU-4DFE	87% and 98%	They have presented a new 3D temporal deformable shape model for both detecting and tracking key landmarks on 3D range mesh models. They have evaluated the accuracy of the feature detection and validated its utility for subject verification and expression classification in multiple public databases.
Bart lett [41]	Gabor motion energy filters	SVM	Cohn Kanade	78.56% for onset and 97.8% for apex	only the magnitude of the responses are used
Sebe [43]	Piecewise Bezier volume deformation	K nearest neighbor	Cohn Kanade	93%	use of geometric features to detect emotions
Sung and Kim [45]	Active appearance model	Linear discriminant analysis	Cohn Kanade MMI		only 3 expressions are detected and no result on benchmark databases
Little wort [46]	Gabor filters	SVM and Adaboost	Cohn Kanade		AU recognition
Koles tra [47]	AAM	HMM	Cohn Kanade		AU detection
Simo n [49]	AAM and PCA	SVM	RU FACS	83.75%	Combines the output of static SVMs for multiple frames and uses structured-output learning to learn the beginning and end time of each AU

Table 2.6: Comparison of all surveyed methods

References	Characteristics of ideal facial expression recognition system (as listed above in section 2.6.1)							
	1	2	3	4	5	6	7	8
Yaser [10]	√	√	×	×	√	×	×	√
Black [11]	×	√	×	×	√	×	×	√
Kimura [12]	√	√	×	×	×	×	√	×
Essa [14]	√	√	×	×	√	×	×	√
Cohn [50]	–	√	×	×	√	×	×	√
Wang [19]	√	√	×	×	√	×	×	√
Tian [21]	√	√	×	×	×	×	√	×
Cohen [22]	√	√	×	×	×	×	×	√
Michel [24]	√	√	×	×	×	×	√	√
Valstar [25]	×	√	×	×	×	×	√	√
Pantic [28]	√	√	×	√	×	√	√	√
Zheng [29]	×	×	×	×	√	×	×	√
Kotsia [30]	×	×	×	×	√	×	×	√
Sayed [33]	×	√	×	×	√	×	×	√
Kotsia [32]	–	√	×	√	×	×	√	√
Ligang [35]	–	√	×	×	×	×	√	√
Valstar [36]	–	√	×	×	×	×	√	×
Wang [51]	√	√	×	×	×	×	×	√
Khan [38]	–	√	×	×	×	×	√	√
Fang [39]	√	√	×	×	×	×	√	√

Legend √= Yes; ×= No; – = Missing entry

2.7 Review of Hardware Implementation of SVM

Davide A. et al [52] have proposed a digital architecture for SVM learning and its implementation on FPGA. Their algorithm consists of two parts, in the first part they compute the parameters of SVM, while in the second part they use a bisection process for computing the threshold. They have shown that Fibs algorithm [52] can be applied to solve SVM based classification problem with any kind of kernel functions. They have divided the functionality of SVMblock into three phases.

1. Loading phase: In this phase, they load the target vector and the kernel matrix.

2. Learning phase:As soon as loading completes, learning phase starts, according to fib algorithm.

3. Output phase: The results of learning phase, i.e., the values of bias and Lagranges multiplier are produced as output

They use four blocks, namely counters, DSVM, bias, and S-blocks and three controllers for loading, learning, and output phase. Counter block mainly consists of the column counter to index a column of the kernel matrix. It is also used to select a particular RAM, during loading and to select a particular value of alpha during output phase. Counter block also consists of a row counter to index a row of the kernel matrix. DSVM block consists of one dimensional systolic array composed of processing elements, each of which updates the elements of the vector. Kernel matrix is precomputed and stored in RAM. Bias block is composed of adder, left shift registers, multiplexers, and a binary comparator, which detects termination of learning phase. Bisection process is performed in this block. The S - block is used for the computation of sign of product of class label and alpha. They have implemented and tested their algorithm on Virtex II. They have used Sonar dataset for testing their algorithm.

Biasi I. et al [53] designed a system on chip, where SVM module is integrated with MicroBlaze soft processor core of an FPGA from Xilinx. They developed the system on a Memex VB 1000 board containing a Xilinx Virtex-II-1000 FPGA, a P160 communication module, and 32 Mbytes of DDR memory. A MicroBlaze RISC 32 bit soft processor core consists of 32 general purpose registers and has several ports for connecting peripheral buses. External memory is used to store the parameters of classification function. 32 Mbyte DDR is used to store Support vectors, Lagranges multipliers and the input sample to be classified. Computation of dot products is performed by multipliers available on FPGA. The architecture is built around the FSL bus provided by Xilinx for the development of SOC. The main modules are Scalar module, which computes parallel dot product or squared norm, the Kernel module and the output MAC (multiplication and accumulation) module. These three modules are connected to one another by buffering buses so that they can sustain a pipelined stream of data. Once the input vector to be classified is stored, each SV is pushed in scalar module to execute the given computation. It needs 5 clock cycles for each SV. The parameter of the exponential function is stored in Kernel module, which is bypassed for linear kernel. The result obtained after multiplication is used as the input of a LUT, which maps the kernel. This module needs 6 clock cycles. The last MAC module consists of multiplier, adder and accumulator which needs 6 clock cycles to complete the computation. They have tested their algorithm on HEP database with classification error rate of 24.2%.

Boni A. et al [54] have proposed implementation of SVM on a low power and low cost 8 bit microcontroller. The proposed solution is used to implement smart sensors and sensor networks for intelligent data analysis and pervasive computing. Their architecture consists of four blocks, namely flash memory, Ktron Drive,

Kernel unit and MAC unit. All support vectors, alphas, LUT table and CORDIC variables are stored in the FLASH memory. Input vector and all temporary variables are stored in RAM. Ktron drive receives the data and starts computation. The Kernel unit computes an inner product or a squared norm according to the chosen kernel. In case of Gaussian kernel, the exponential function is computed by LUT or the CORDIC algorithm. The MAC unit multiplies the output of the kernel function by the coefficient alpha and accumulate the result in the output register.

Kyrkou C. et al [55] proposed systolic array architecture for SVM. They have presented a distributed pipelined architecture, which can be expanded in a systolic manner to provide efficient management of memory and data resources. The systolic chain of processing elements consists of identical PEs. They also address the wiring and fan out complexity of routing the input vector. Their architecture consists of three main parts, input (front end PEs), computation (middle PEs), and output (back end PEs). The input part consists of input vector memory, address generator unit and front end PEs. In middle part bulk computation occurs, where each PE receives data from previous PE, processes it and propagates it to next PE in a pipelined manner. Output part consists of back end PEs to transfer all the data outside the chain. It also consists of kernel unit, MAC unit and the memory for alpha coefficients. Finite state machine control unit controls all three parts. MAC is the main processing unit, which performs the kernel's vector operation between SV and the input vector. The kernel can be implemented either by custom logic or LUT. Time required for the first input vector element to reach the back end PE is n cycles, where n is the number of PEs. The PEs then requires k cycles to compute the scalar value, where k is the number of elements in vector. Additional n+2 cycles are required for the front-end PE's scalar value to reach the kernel unit and accumulate. The chain is processing, n SVs in parallel. Therefore, for m SVs they need m/n repetitions for all SVs. The total cycles required for classification of the input vector is $(n + k + (n + 2)) * [m/n]$. They have implemented their algorithm on a Virtex 5 FPGA board and evaluated using training data from a face detection application. They have mapped MAC units both on the FPGA's embedded DSP units and on the slice logic. The internal SV memory in each PE was implemented using the dedicated FPGA block RAM. They train SVM in MATLAB, using 6800 samples out of which 4400 are non faces, and 2400 are faces. From the training they got 818 SVs that were distributed among 100 PEs. Some PEs had 9 SVs and some had 8 SVs. Each vector consists of pixels from 20 x 20 image, i.e., 400 elements of 8 bits. They have used polynomial kernel of order 2. They achieved 88% detection accuracy.

Mahmoodi D. et al [56] have proposed a simple hardware architecture for implementation of pairwise Support Vector Machine on FPGA. Training part of the SVM is performed offline, and the extracted parameters are used to implement testing phase on the hardware. They perform vector multiplication operation and classification of pairwise classifiers in parallel and simultaneously. For experimentation they use a dataset of Persian handwritten digits in three different classes. They use graphical simulator and System Generator to simulate the desired hardware design. They have Implemented linear and nonlinear SVM classifier using simple blocks and functions. According to simulation results, total time required for computation of linear classification is 1.4 ms, while for non linear classification it is 0.27 ms. Classification error rate for linear classification is 12%, while for non linear classification it is 1.33%.

2.8 Conclusions

Is it possible to distinguish the basic facial expressions mutually? Is any expression creates confusion with the other? During research of facial expressions, in 1978 Ekman and Friesen [5] showed that *there is confusion between anger-disgust and fear-surprise.* The reason for that may be the shairing of common facial movements and actions by these expressions. This psychological observation also applies to modern day automatic facial expression recognition systems. We see similar confusion between fear-surprise and anger-disgust in the results of most of the surveyed papers, which is very interesting. However, we see in many of the systems fear is confused with happiness and in few systems fear is confused with anger. In some systems sadness is confused with anger. From the re*sults of the surveyed works,* we see that, out of the six basic expressions, surprise and happy are easy to recognize. It is always not possible that facial expression falls into any one of the six basic expressions. There can be other expressions that should be recognized. But, it is more challenging to capture and recognize non basic expressions. In surveyed paper, we see that all expressions are recognized with different accuracies. For different expressions the recognition percentages are different. There is need to work towards eliminating such confusions. It is also necessary to recognize all the expressions with equal accuracy. Facial features and facial expressions differ with different cultures such as Asians and Europeans. Features are different for different age groups such as adults and children. Facial expression recognition systems must be robust against these differences. From literature review, it seems that very little work is done on automatic AUs detection and recognition of expressions with different head angles and rotations. Pantic and Rothkrantz [4] have recognized facial expressions from profile faces. *But, as*

such no detailed literature is available related to this work. Manual processing is required in many systems. In many systems, to track facial feature points, manual labeling on the first frame is required. *To design a fully automatic system is a big challenge.* In recent years, researchers are trying to develop fully automatic facial expression recognition systems. To label the available data is another major challenge. Bulk of unlabeled data is easily available, but it is very difficult to get labeled data. Data labelling or modifying is a very time consuming process. To perform this task expert observer or the AU coder is required. Non availability of the spontaneous expression database is the major challenge that the researchers are facing.

From comparison of different approaches, we conclude that till today non of the approach has absolute accuracy. Most of the proposed methods are not real time, they are applicable only for frontal faces, tilted faces are not allowed. Many of the methods are semi automatic, they need initial fitting or labeling manually. The limitations in automatic expression recognition are to a large extent the result of high variability that can be found in images that contains a face. We will see an extremely large variety in lighting conditions, resolution, pose and orientation. In order to be able to analyze all these images correctly, an approach seems to be desirable that can detect and separate these source of variation from the actual information. We have used active appearance model (AAM), which enables us to automatically create a model of a face in an image. The created models are realistic looking faces. This ensures that variety in light variations, resolutions, pose and orientation will have no effect on expression recognition. In next Chapters, we discuss our proposed approaches to address many of these limitations.

Chapter 3

Proposed approach for facial expression recognition

We have proposed a framework, which is composed of four subsystems. The first subsystem is used for face detection. The second is used for facial feature extraction and the Candide wire frame fitting on the first frame of the image sequence. The third subsystem is used for feature tracking and geometrical displacement extraction of the wire frame grid nodes, and the fourth subsystem is used for grid node displacement classification. We have used Viola Jones algorithm [57] for face detection. We have developed algorithm for facial feature point detection using Gabor filters. Facial feature points are also detected by image segmentation and by applying suitable threshold. We have used Active appearance algorithm [58] for wire frame fitting on the first frame and feature point tracking in subsequent frames. For classification of grid node information we have used a multi class SVM system. The facial expression extraction component used in our method is the Active Appearance Model (AAM). AAM is used in expression recognition because of its ability to extract expressions from faces accurately. By combining a shape model and an appearance model, AAM creates a global model of the face. Changes of facial landmarks are captured by the shape model while the appearance model captures the textural changes of surrounding landmarks from the facial region. The procedure of creating AAM model of the face from the image sequences consists of three steps: (1) manually labelling a grid of landmarks for training images of faces, (2) creating statistical models of the appearance and shape of the training faces and (3) automatically fitting the grid of landmarks to the new face images (rest of the frames from the image sequence) using the created model. The flow diagram of the proposed framework is shown in Figure 3.1.

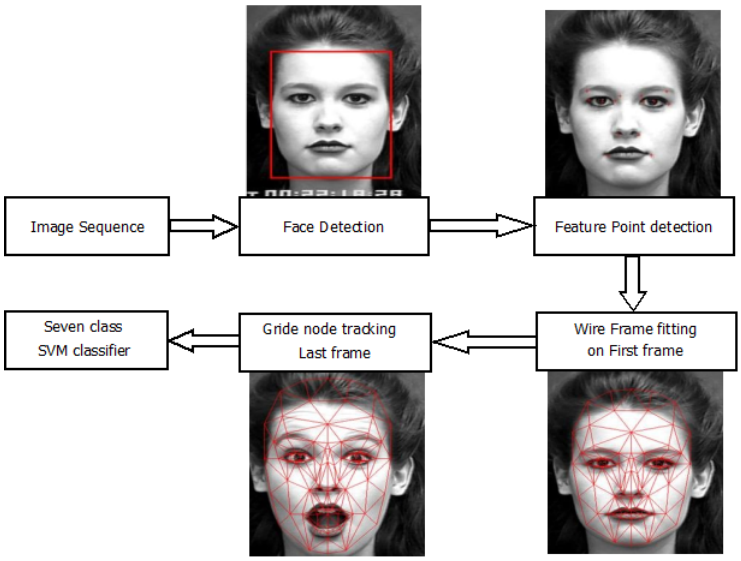

Figure 3.1: Flow diagram for facial expression recognition system

3.1 Face detection

In present work, as we wish the system to be fully automatic, we have to start by detecting the user's face inside the scene. Although, we seemed it an easy problem at first, we immediately realized that the high variability in the types of faces encountered would make the automatic detection of the face a tricky problem. Many different techniques have been reported in the literature for face detection. In our approach face area of an image is detected using the Viola Jones algorithm [57]. The result of face detection algorithm is shown in Figure 3.2. For details see the section 7.1

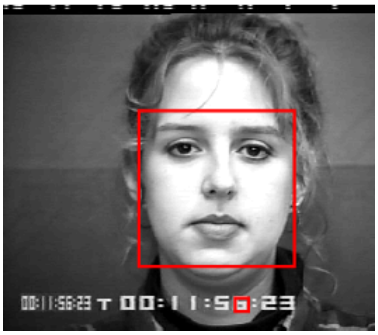

Figure 3.2: Face detection

3.2 Facial Feature Point Detection

We have developed two methods for automatic facial feature point detection in image sequences. One uses Gabor filter to detect 14 facial feature points such as

eyebrows corners, eyes corners, nostrils, lip corners, center of upper lip and center of lower lip. Other method uses image normalization, and suitable threshold to detect 14 facial feature points such as eyebrows corners, eyes corners, nostrils, lip corners and eyeballs.

3.2.1 Feature point detection using Gabor filter

We have used a set of Gabor filters with different frequencies and orientations for extracting useful features from an image. For details see section 7.2 and section 7.3. Figure 7.3 shows different features extracted when passed through Gabor filters and Figure 7.4 shows amalgamations of different features extracted. Different face images are shown in Figure 3.3. These images are passed through Gabor filters to extract different features. After amalgamations of different features extracted, we got images as shown in Figure 3.4. To these images, we have applied a suitable threshold, in order to get clear features as shown in Figure 3.5.

Figure 3.3: Face images

Figure 3.4: Images obtained after amalgamations of different features extracted

Figure 3.5: Images after applying suitable threshold

We have divided this image horizontally in to four equal parts, so that forehead, eyes, nostrils and mouth regions are separated from each other as shown in Figure 3.6.

Figure 3.6: Face divided into four regions

We neglect forehead region, as no facial feature point lies in this region. Rest of the three regions are vertically divided into two regions, so that left and right eyes, left and right nostrils, left and right lip corners are separated as shown in Figure 3.7.

Figure 3.7: Left and right eyes, nostrils and lip corners separated

Left and right eye regions are further divided to separate eyebrows and eyes from each other. Left regions are cropped from left sides and right regions are cropped from right side to remove unwanted face counters. Now all left regions (eyebrows, eyes, nostril and lip) are scanned horizontally from right to left to detect left eyebrow corner, left eye corner, left nostril and left lip corner. All right regions are scanned horizontally from left to right to detect right eyebrow corner, right eye corner, right nostril and right lip corner. Left mouth region is scanned vertically on extreme right side to get upper lip and lower lip middle points or we can scan right mouth region vertically on extreme left side to get same points. Detected points are shown in Figure 3.8.

Figure 3.8: Detected facial feature points

3.2.1.1 Experimental results

The MPEG-4 standard defines 84 key feature points (FPs) on the neutral face [2]. To recognize facial movements and to animate the faces, movement of FPs can be used.. Figure 3.9 shows a neutral face with the location of the 84 FPs defined by the MPEG-4 standard. Out of these 84 points, our algorithm detects 14 points. Detection rate of these points is given in Table 3.1 for Cohn kanade [59] as well as IMM [60] database. Average recognition rate obtained for Cohn Kanade database is 89% and IMM database is 86%. We required rough estimate of these points for further processing. Exact location is not required. Active appearance algorithm is used further for wire frame fitting on face image with reference to these points. AAA will take care of exact fitting.

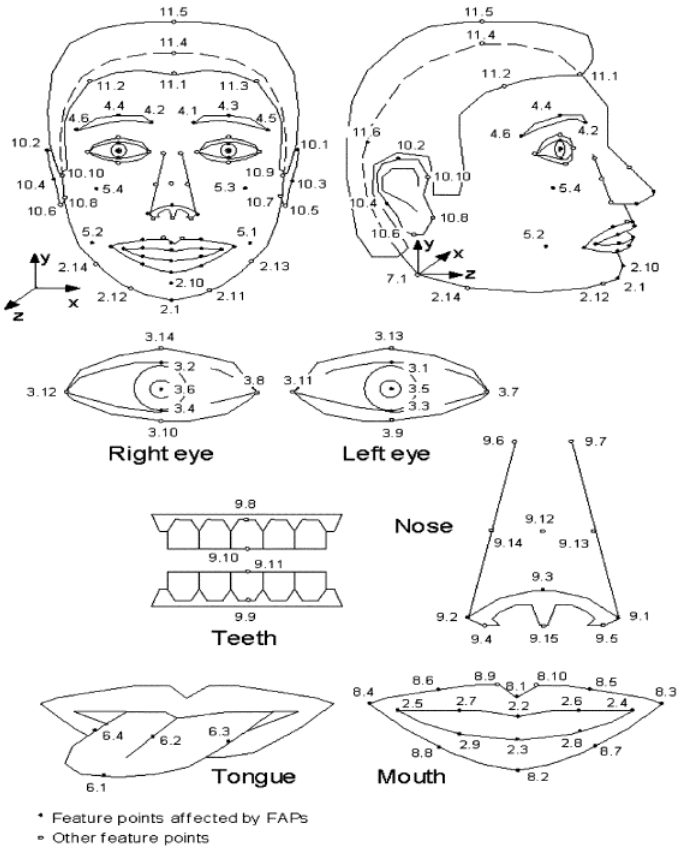

Figure 3.9: 84 Facial feature points as per MPEG-4 standard [2]

Table 3.1: Facial feature point detection results for Cohn Kanade and IMM database

FP	Description	Detection rate for samples from Cohn Kanade Database	Detection rate for samples from IMM Database
3.7	Outer corner of left eye	89%	86%
3.11	Inner corner of left eye	88%	82%
3.12	Outer corner of right eye	89%	85%
3.6	Inner corner of right eye	87%	89%
4.6	Outer corner of left eyebrow	85%	84%
4.1	Inner corner of left eyebrow	78%	82%
4.8	Outer corner of right eyebrow	84%	79%
4.2	Inner corner of right eyebrow	89%	82%
9.1	Left nostril	95%	90%
9.2	Right nostril	94%	92%
8.3	Left corner of lip	95%	93%
8.4	Right corner of lip	92%	89%
8.1	Upper middle point of upper lip	89%	92%
8.2	Lower middle point of lower lip	88%	91%
	Average recognition rate	89%	86%

3.2.2 Feature point detection by threshold

Once the face is detected using Viola Jones algorithm [57], we divide the detected face region into 5 relevant regions of interest, each of which is examined separately, further to detect the location of the facial feature points. In each region, we have normalized the image with respect to brightness. We have selected a suitable threshold for each region, using which image is converted into binary image. Then in each region extreme ends of binary image will locate the facial feature points. In the eye region horizontal and vertical histograms are analyzed to detect eyeballs.

3.2.2.1 Detecting Regions of Interest

The face detection algorithm detects the face as shown in Figure 3.10. Face is enclosed by a rectangle. This algorithm gives the coordinates of upper left corner of rectangle.

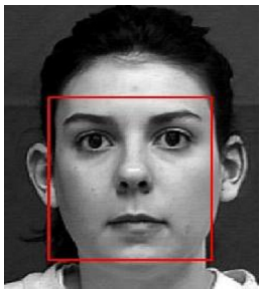

Figure 3.10: Face detection

It also gives the width (w) and height (h) of the rectangle as shown in Figure 3.11. We crop the image according to the rectangle.

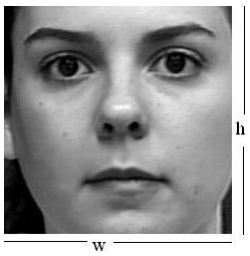

Figure 3.11: Cropped face image

Now we divide this image into three equal parts horizontally as shown in Figure 3.6, so that upper part contains eyes, middle part contains nostrils, and lower part contains mouth

Figure 3.12: Face divided into three regions

Again we divide upper part into two parts vertically, so that each part contains one eye, as shown in Figure 3.13.

Figure 3.13: Eyes Separated

Again each eye region is divided horizontally into two parts so that eyebrows and eyes are separated as shown in Figure 3.14.

Figure 3.14: Eyebrows and eyes separated

The position of the eyeball from the segmented eye region can be detected by sequentially applying analysis of the vertical histogram which shows the intensity differences between the successive rows pixel wise. The peak of this histogram gives the y coordinate of the eyeball. Similarly peak of horizontal histogram which shows the intensity differences between the successive columns pixel wise gives x coordinate of the eyeball. With this approach we have detected eyeballs as shown in Figure 3.15.

Figure 3.15: Eyeballs detected

3.2.2.2 Eyes corner detection

We have considered the segmented eyes region separately. For the left eye, image is cropped from left side, while for right eye image is cropped from right side. Then for eye region we have applied image normalization technique, which is a linear process. If the intensity range of the image is 40 to 170 and the desired range is 0 to 255 the process entails subtracting 40 from each of pixel intensity, making the range 0 to 130. Then each pixel intensity is multiplied by 255/130, making the range 0 to 255. If $N_1 \times N_2$ is the size of the image, then we compute average as follows

$$avg = \frac{1}{N_1 N_2} \sum_{x=1}^{N1} \sum_{y=1}^{N2} R(x,y) \tag{3.1}$$

where $R(x,y)$ is pixel value at coordinate (x,y). Then, we have selected a suitable threshold value. Using trial and error method we have selected threshold value as $\frac{avg}{3}$. Using this threshold gray scale image is converted into binary image as follows.

$$if\ R(x,y) > \frac{avg}{3},\ \ R(x,y) = 1,\ else\ \ R(x,y) = 0 \tag{3.2}$$

After threshold we got binary image. This binary image we have scanned vertically to get first and last black point, which gives us corners of eyes, as shown in Figure 3.16.

Figure 3.16: Eyes corner detection

3.2.2.3 Eyebrows corner detection

Let's consider another ROI that is eyebrows region. Left side of the left eye, and right side of right eye is cropped. Here, also we have applied image normalization, threshold and vertical scanning technique to get eyebrows corners. Eyebrows are detected as shown in Figure 3.17.

Figure 3.17: Eyebrows corner detection

3.2.2.4 Nostrils Detection

Now we consider next ROI that is nostril region. Again the same techniques are applied in this region to detect nostrils, as shown in Figure 3.18.

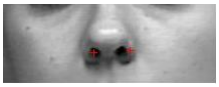

Figure 3.18: Nostrils detection

3.2.2.5 Lip corner detection

Now we consider the mouth region. In this region also we have cropped image from left as well as from right side. Then we have applied image normalization,

threshold, and vertical scanning techniques to detect lip corners as shown in Figure 3.19.

Figure 3.19: Lip corner detection

3.2.2.6 Experimental results

We have tested this algorithm on Cohn-Kanade [59] database as well as on IMM [60] database. We have detected 14 facial feature points with average recognition rate of 86% for Cohn kanade database and 83% for IMM face database. Some results are shown in Figure 3.20.

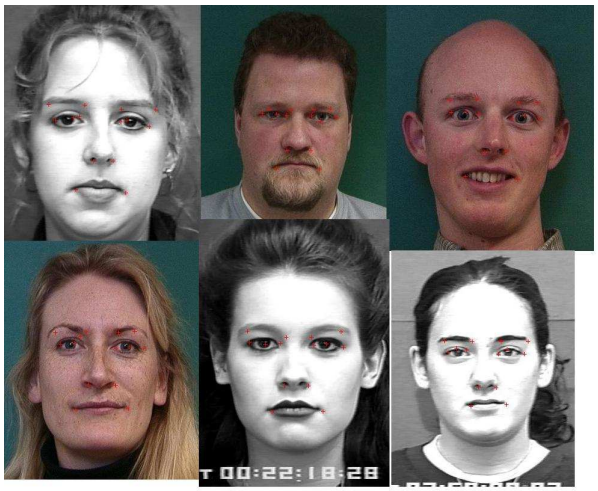

Figure 3.20: Results of 14 facial feature point detection

Exact points could not be located, however, we have approximately located 14 facial feature points. In comparison with the previously used techniques such as Gabor filters [61], we have used rather a simple technique. So computational complexity is very less. In [61] authors used Gabor filters and adaboost algorithm to detect 20 facial feature points. They divided face image into 20 ROI. Each ROI is convolved with set of 48 Gabor filters, and then Gabor coefficients are collected as features in a huge matrix form, which increases computational complexity as well as computational time. As they are using adaboost classifier, they need training, which is a time consuming process. Their method is applicable only for expressionless and frontal facial images. Even after huge computations the techniques are applicable only for one particular database. They got 93% average accuracy. In our case accuracy is 86%, but computational complexity is less. No training data is required. It can be applied for tilted faces with expressions. Our approach using Gabor filter is also computationally expensive.

Table 3.2: Facial feature point detection results for Cohn Kanade and IMM Database

FP	Description	Detection rate for samples from Cohn Kanade Database	Detection rate for samples from IMM Database
3.7	Outer corner of left eye	85%	82%
3.11	Inner corner of left eye	87%	83%
3.12	Outer corner of right eye	83%	81%
3.6	Inner corner of right eye	82%	80%
4.6	Outer corner of left eyebrow	81%	81%
4.1	Inner corner of left eyebrow	80%	82%
4.8	Outer corner of right eyebrow	85%	84%
4.2	Inner corner of right eyebrow	84%	81%
9.1	Left nostril	90%	82%
9.2	Right nostril	91%	87%
8.3	Left corner of lip	92%	88%
8.4	Right corner of lip	90%	84%
3.5	Left eyeball	87%	80%
3.6	Right eyeball	90%	85%
	Average recognition rate	86%	83%

Table 3.2 shows the detailed analysis of results. By keeping the exact feature point at the center within 4×4 window, if point is detected, we consider that point is detected otherwise we consider it to be false detection.

3.3 Automatic wire frame fitting on frontal face

3.3.1 Candide wire frame model

We have used the Candide wire frame model for facial expression recognition. The Candide model was created by Mikael Rydfalk at the Linkoping Image Coding Group in 1987. Later, Bill Welsh at British Telecom created another version with 160 vertices and 238 triangles covering the entire front head (including hair and teeth) and the shoulders. This version, known as Candide-2 is delivered with only six Action Units. A third version of Candide has been derived from the original one by Jorgen Ahelberg [62]. The Candide wire frame is a parametrized face mask specifically developed for model-based coding of human faces. A frontal view of the model can be seen in Figure 3.21. It has 113 vertices and 184 triangles. The small number of its triangles, allows fast face animation with moderate computing power.

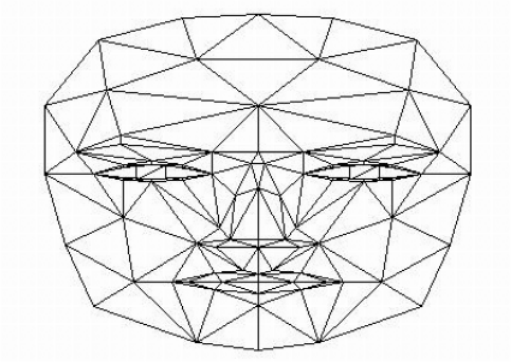

Figure 3.21: Candide wire frame model

The geometry of the model as discussed in [62] can be expressed as in (3.3).

$$V(\sigma, \alpha) = \overline{V} + \sum_{i=1}^{14} S_i \sigma_i + \sum_{i=1}^{65} A_i \alpha_i \tag{3.3}$$

Here the resulting vector V contains (x, y, z) coordinates of vertices of the model. $\overline{V}$ is a vector containing vertex coordinates of the standard model. S_i represents a shape unit, whereas σ_i represents shape parameter. There are 14 shape units, such as head height, eyebrows vertical position, mouth width, eyes width etc. A_i represents animation unit, whereas α_i is animation parameter. There are 65 animation units such as lip stretched, inner brow raiser, outer brow raiser, nose wrinkle, etc. The difference between shape and animation units is that the shape units define deformations that differentiate individuals from each other, while the animation unit define deformations that occur due to facial expression. We have added six more parameters, three for rotation, one for scaling, and two for translation to the formula in (3.3), to perform global motion of the model [63].

$$V(R, s, \sigma, \alpha, t) = Rs(\overline{V} + S\sigma + A\alpha) + t \tag{3.4}$$

Here, $R = (\theta_x, \theta_y, \theta_z)$ is a rotation matrix, s is scale, and $t = (t_x, t_{y,}, t_z)$ is a translation vector where $t_z = 0$.

$$\theta_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cosx & -sinx \\ 0 & sinx & cosx \end{bmatrix}, \theta_y = \begin{bmatrix} cosy & 0 & siny \\ 0 & 1 & 0 \\ -siny & 0 & cosy \end{bmatrix}, \theta_z = \begin{bmatrix} cosz & -sinz & 0 \\ sinz & cosz & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The geometry of the model is thus parametrized by (3.5). Shape unit parameters are static for the same person while the animation parameters are varying

throughout the tracking procedure.

$$p = [\theta_x, \theta_y.\theta_z, s, t_x, t_y, \sigma, \alpha]^T \tag{3.5}$$

Once the model is adapted properly on the first frame, for the subsequent frames only α will change. Our goal is to find the optimal adaptation of the wire frame model to the input image i.e. to find p that minimizes the distance between the wire frame model and the image [62]. Seven basic facial expressions are shown in Figure 3.22 and wire frame model deformations for these seven expressions are shown in Figure 3.23.

Figure 3.22: Seven basic facial expressions Neutral, Fear, Surprise, Sad, Anger, Disgust, Happy

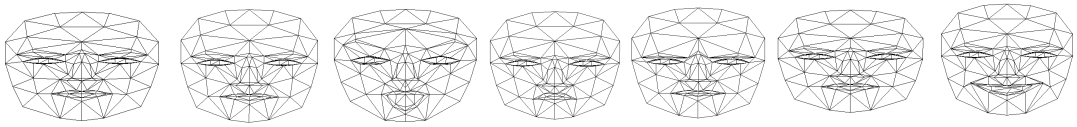

Figure 3.23: Wire frame deformations for Neutral, Fear, Surprise, Sad, Anger, Disgust, Happy

3.3.2 Active Appearance model

The aim of Facial Expression Recognition system is to determine the expression from appropriate facial features extracted from video images automatically. Facial features can be divided into two classes: geometric-based and appearance-based features. Geometric-based features consist of the shapes or locations of characteristic parts of the face (e.g., the shape of the mouth or the center of the eyes). Systems using such features often rely on Active shape model (ASM), particle filtering, or a statistical model that combines global and local information about the feature points to track specific points in the face. These methods allow for the precise tracking of facial shapes and points, they do not consider textural changes.

Appearance based features contain information about facial texture, such as the appearance of wrinkles or bulges. Systems based on appearance based features make the use of methods such as Gabor wavelets, optical flow. Appearance based

features do not handle variations in illuminations and partial occlusions, but they capture textural changes.

A method combining both geometric and appearance based features is called Active Appearance Models (AAM) and was proposed by Cootes, Edwards, and Taylor [58]. AAMs are used as a tool by scientists, because of their ability of automatic fitting of a wire frame model to the video recorded face. But their drawback is they require the manual selection of landmarks, i.e., the facial feature points on face image of a video sequence for a representative frame. The wire frame model automatically fit on the subsequent frames. Texture models and statistical shapes are combined to form a appearance model. These texture and shape models are trained so that, they know the modes of variation in texture and shape. The training set normally consists of face images, with landmark points manually annotated to outline the face. Principal component analysis (PCA) is applied to the vector of landmarks to reduce its dimensionality. The model learns the relationship between the displacement of the model parameters and the residual error. To fit the model automatically, the current residuals are measured. The model is used to find a fit which reduces the residual error. The minimization of the residual error is obtained using a gradient approach. The process stops, when the residual error is below a certain threshold. The most important thing is to initialize the model close to the observed face, otherwise there is possibility that, it will converge to a (incorrect) local minimum. Cootes and Taylor [58] used this model for tracking a face using 88 labeled training images and they got good results. Out of all images, 19% of images are failed to converge, but the images for which it converged were accurate. Bu it was not a real-time system. The reason for that is the large number of iterations are required to converge the search algorithm . Another problem was that it was not robust against occlusions, because all the landmarks must be visible in each frame to track the face. In our algorithm, instead of minimizing the distance between the image and its best approximation,we have minimized the distance between the synthesized image associated with the previous frame and the actual image.

A big disadvantage of AAM is that it is unable to handle partial occlusion. The main and important advantage of AAM is that they give a good pose estimation because they can find a precise location of the head. But its accuracy depends a lot on the training set that is used. The training set must consists of lot of different faces and different poses with different expressions.

3.3.3 Wire frame model fitting on the first frame

To fit wire frame model on the first frame of the image sequence, we consider scaling, rotation and translation parameters. For initial fitting, rough estimate of scaling and translation parameter is very essential. The face detection algorithm provides a rectangle enclosing the face as shown in Figure 3.10. The top left point of this rectangle has coordinates (x, y). Width of the rectangle is W_d and height is H_t. From this data we calculate a rough estimate of translation parameters (t_x, t_y) as in (3.6) and (3.7). Our aim is to match the center of the model with the center of the face image. Then we scale the model so that it will fit on the face.

$$t_x = x + W_d/2 \tag{3.6}$$

$$t_y = y + H_t/2 \tag{3.7}$$

The wire frame model is fitted manually on 100 images. From manual fitting we have selected rough estimate of scaling parameter as $s = 0.78 * W_d$. With a rough estimation of scaling and translation parameters, wire frame model roughly fits on face image. Initially rotation is assumed to be zero. Once the model roughly fits on face image, exact fitting is obtained using active appearance algorithm.

3.3.4 Geometrically Normalized model

By creating geometrically normalized or shape free face images, we can transform the face space into a convex one. Geometrical normalization of the face is used to remove texture variations caused by its global and local motion. It also removes geometrical differences between individuals [64]. We have decided to work with 33×40 pixels images, which are small and effective for image warping. Geometrically normalized model is shown in Figure 3.24. Translation, rotation and scaling is performed on wire frame model to produce normalized face model. We have removed twelve surfaces on forehead to avoid effects caused by hair. Parameters of normalized face model are given in Table 3.3.

Table 3.3: Parameters of normalized face model and removed surfaces

Translation	[17.11 16.926 0]
Rotation	[0 0 180]
Scaling	[25 25 25]
Shape Unit Parameters	[0 ..... 0]
Animation Unit Parameters	[0 ..... 0]
Removed Surfaces	[0 34 1] [44 0 34] [0 11 1] [45 44 34] [1 11 12] [45 46 34] [1 12 13] [47 45 46] [12 13 14] [1 13 2] [1 2 34][2 46 34]

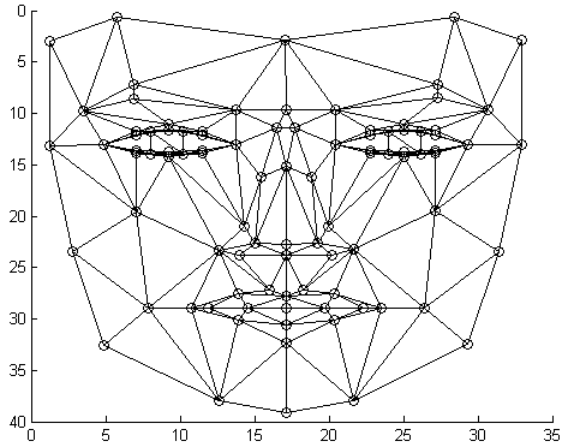

Figure 3.24: Geometrically Normalized model

We have set scaling and rotation parameters in advance. It is necessary to perform 180 degrees rotation around Z axis due to opposite direction of Y axis. Then we have calculated minimum coordinates for X, Y axis of the model vertices with unremoved surfaces. This is necessary because, the origin of image in MATLAB is at the top left corner. It starts from (1, 1) instead of central point. As the center of Wire frame model is at (0,0,0), translation is necessary.

3.3.5 Texture Mapping

Once the model is geometrically normalized, we map the texture of input image on this geometrically normalized image. For texture mapping we use the concept of barycentric coordinate computation of triangles. Wire frame model is having 184 triangles. We compute barycentric coordinates of all triangles and store. When the model fits on face, the texture of face image is mapped on geometrically normalized image.

We have two wire frames one is on the face image and the other is geometrically normalized wire frame. Consider any triangle of the geometrically normalized frame (destination) and same triangle of wire frame on the face image (source). I want to copy texture of source to destination. This can be done using barycentric coordinates computation. We precomputed the barycentric coordinates of the triangles of the geometrically normalized wire frame. We know the vertex coordinates, so we compute barycentric coordinates of all triangles using (3.18) (3.19) and (3.20). Our image size is 33×40, means we have 1320 pixels. Out of which 966 pixels are valid barycentric coordinates. From the destination wire frame, we consider first triangle and its barycentric coordinates. We consider the same triangle from source image, here we have barycentric coordinates, from which we compute the Cartesian (source) coordinates (x, y) using (3.11) and (3.12). Now,

we got source coordinates for corresponding destination coordinates, so we can copy the texture of the source image to the destination image.

Barycentric coordinate computation

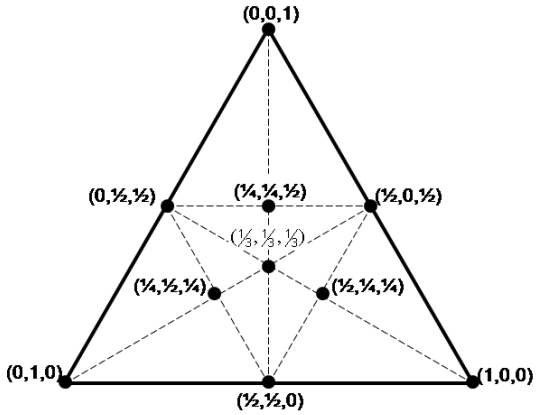

Figure 3.25: Barycentric Coordinate Computation

Let us consider a triangle T which is formed by three vertices r_1, r_2, r_3. A weighted sum of these three vertices, may be written as in (3.8), which locates any point r on this triangle. Here, λ_1, λ_2, and λ_3 are the area coordinates. These are subjected to the constraint given in (3.9), and λ_3 is given by (3.10) [65].

$$r = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3, \tag{3.8}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{3.9}$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 \tag{3.10}$$

$\lambda_1, \lambda_2, \lambda_3$values should be positive. If a point r lies inside a triangle, and it is desirable to obtain the barycentric coordinates λ_1, λ_2 and λ_3 at this point, then, we write the barycentric expansion of point r_i having Cartesian coordinates (x_i, y_i) in terms of the components of the vertices of triangle(r_1, r_2, r_3) as in (3.11) and (3.12)

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 \tag{3.11}$$

$$y = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3. \tag{3.12}$$

On substituting (3.10) into (3.11) and (3.12) we get (3.13) and (3.14).

$$x = \lambda_1 x_1 + \lambda_2 x_2 + (1 - \lambda_1 - \lambda_2) x_3 \tag{3.13}$$

$$y = \lambda_1 y_1 + \lambda_2 y_2 + (1 - \lambda_1 - \lambda_2) y_3 \tag{3.14}$$

On rearranging (3.13) and (3.14) we obtain (3.15) and (3.16).

$$\lambda_1(x_1 - x_3) + \lambda_2(x_2 - x_3) + x_3 - x = 0 \tag{3.15}$$

$$\lambda_1(y_1 - y_3) + \lambda_2(y_2 - y_3) + y_3 - y = 0 \tag{3.16}$$

We write this linear transformation as in (3.17)

$$R \cdot \lambda = r - r_3 \tag{3.17}$$

where, $R = \begin{bmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{bmatrix}$ $\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$ $r = \begin{bmatrix} x \\ y \end{bmatrix}$ and $r_3 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix}$
The matrix R is invertible, since $r_1 - r_3$ and $r_2 - r_3$ are linearly independent. The matrix R is not invertible if, r_1, r_2 and r_3 are collinear and would not form a triangle. Solving (3.17) for λ results into (3.18), (3.19), (3.20).

$$\lambda_1 = \frac{(y_2 - y_3)(x - x_3) - (x_2 - x_3)(y - y_3)}{\mid R \mid} \tag{3.18}$$

$$\lambda_2 = \frac{(x_1 - x_3)(y - y_3) - (y_1 - y_3)(x - x_3)}{\mid R \mid} \tag{3.19}$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 \tag{3.20}$$

Barycentric coordinates are a linear transformation of Cartesian coordinates. So they vary linearly along the edges and over the area of the triangle. All of the Barycentric coordinates of a point lie in the open interval (0,1), if that point lies in the interior of the triangle. At least one of the area coordinates $\lambda_1, \lambda_2, \lambda_3$ is zero, if a point lies on an edge of the triangle. The rest of the area coordinates lie in the closed interval [0,1]. So we can say that point r lies inside the triangle if and only if $0 < \lambda_i < 1 \; \forall i$ in 1,2,3. Else, r lies on the edge or corner of the triangle if $0 \leq \lambda_i \leq 1 \; \forall i$ in 1,2,3. Otherwise, r dose not belong to that triangle and it lies outside the triangle [65].

B) Texture mapping

We have a mesh of triangles in two different shapes. One is for source image and another is for destination image (geometrically normalized model). We have to warp an image from one shape to another. Our destination image or mesh i.e. geometrically normalized model DI consists of 172 triangles $T_{1,\ldots\ldots}T_{172}$. Each triangle is made up of three vertices, and each vertice has x, y coordinates. That is, $T_n = [r_1,\ r_2,\ r_3]$ and $r_i = [\ x_i \quad y_i]^T$. Our source image or mesh SI has 184 triangles. Here, our aim is to map texture of source image to destination image. Now, we perform texture mapping process as per following steps [64].

1. We compute barycentric coordinates $\lambda_1, \lambda_2, \lambda_3$ for each pixel coordinate (x, y)

in destination mesh (DI) with respect to the first triangle T_1 in the destination mesh. If $\lambda_1 + \lambda_2 + \lambda_3 = 1$, then only the barycentric coordinates are valid and the point lies inside the triangle, otherwise the point dose not lie inside the triangle and we go for the next triangle until the correct triangle T_n is found.

2. Our destination mesh is geometrically normalized model, so its coordinates are fixed. Barycentric coordinates for each pixel in destination mesh will also be fixed. So we compute them once and store $\lambda_1, \lambda_2, \lambda_3$ for each pixel. As we are working on 33×40 pixels, out of 1320 pixels, valid barycentric coordinates are found only for 966 pixels.

3. Our source coordinates are not fixed. For the source mesh, we know (x, y) coordinates of all vertices of triangle. We have a set of barycentric coordinates calculated in step 2. So, we compute the source coordinates as per (3.13) and (3.14).

4. Now, we have destination coordinates and source coordinates. We copy the texture of source coordinate to destination coordinate. Out of 1320 pixels we get texture copied on 966 pixels, for rest of the pixels, values are obtained using bilinear interpolation.

3.3.6 Synthesized Image

We have adapted the wire frame model to a set of images using different parameters such as rotation, translation, scale, and action units. We have collected those parameters in a vector p, which thus parametrizes the geometry of the model. For each image in the training set, we have mapped the texture of the image under the wire frame model on to the model. Then, we have normalized the model to a standard shape, size, and position, in order to collect a geometrically normalized set of textures. On this set of textures, we have applied a PCA and computed the eigentextures (geometrically normalized eigenfaces), as in (3.21).

$$x = \overline{x} + X\xi, \tag{3.21}$$

Here, $\overline{x}$ is mean texture, X is eigen texture and ξ is texture parameter. Now, we can describe the complete appearance of the model by the geometry parameters p and a N dimensional texture parameter vector. Where, N is the number of eigentextures to be used for synthesizing the model texture. When an input image and parameter p are given, the texture parameters are computed by projecting the normalized input image on the eigentextures. Thus in our case p is the only

necessary paramete. Thus the geometrical normalization of the face is used to remove the texture variations caused by its global and local motion and geometrical differences between individuals. We have decided to work with 33×40 pixel images which are conveniently small and effective for image warping. Barycentric coordinate computation and texture mapping explained in 3.3.5 is used to get geometrically normalized image. We have considered 100 training face images. So N=100. On these images, we fit the wire frame model manually, then we get geometrically normalized images. From these images, we compute the mean image which is called mean texture denoted with $\overline{x}$. We have performed PCA on the training set (stored as 33×40 texture vector), so that we obtain the principal modes of variation, i.e., the eigenfaces. We have collected 100 eigenfaces in a matrix X which could represent 90% of the variance. A face vector x can be parametrized as in (3.22), where $\overline{x}$ is the mean face, and we could synthesize a face image as in (3.23) [64].

$$\xi = X^T(x - \overline{x}) \tag{3.22}$$

$$\hat{x} = \overline{x} + XX^T(x - \overline{x}) \tag{3.23}$$

Normalized training set images are shown in Figure 3.26 and eigen face images are shown in Figure 3.27.

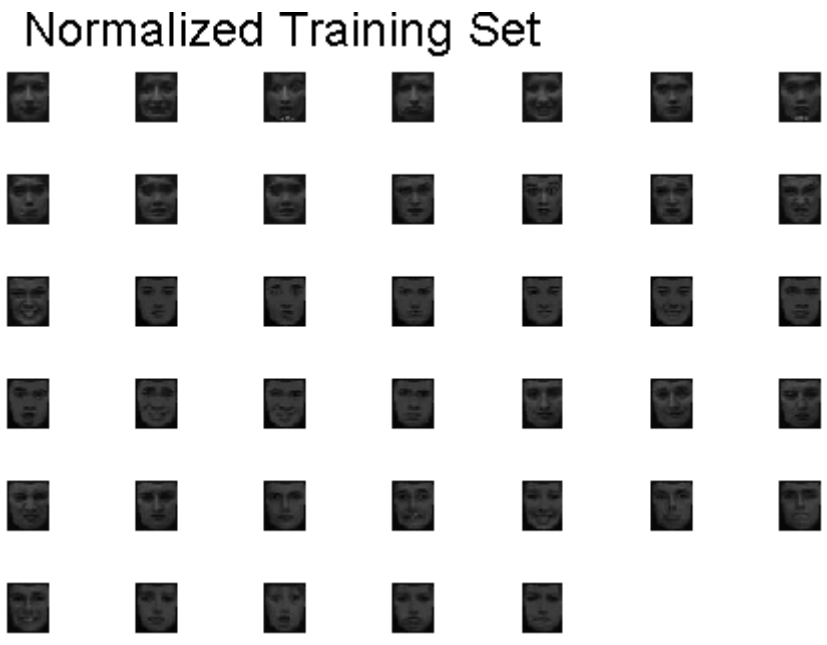

Figure 3.26: Normalized training set images

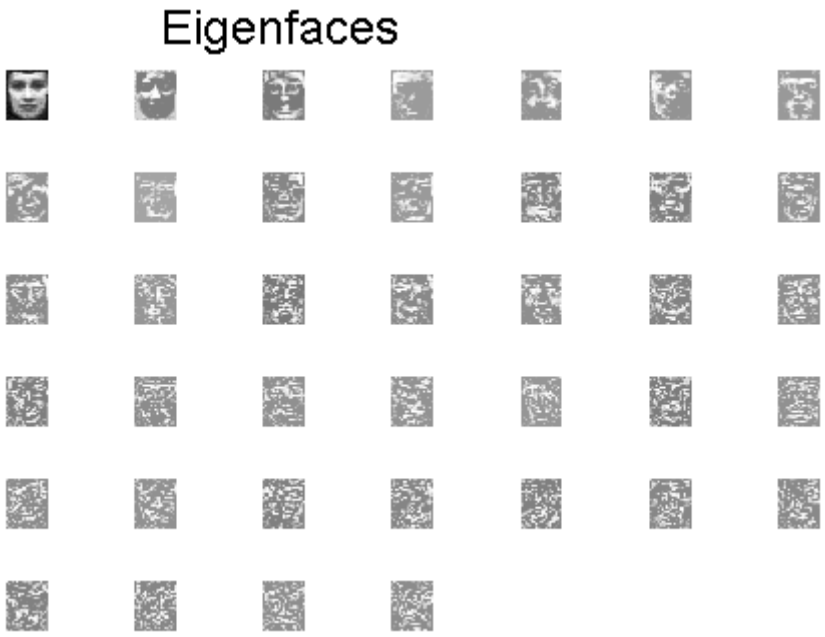

Figure 3.27: Eigen face images

3.4 Tracking

To track facial feature points, means to find an optimal adaptation of the model to facial frames in the image sequence. We have obtained this by finding the parameter vector p that minimizes the distance between synthesized and normalized faces. The initial value of p which we have used is the optimal adaptation to the previous frame. We have assumed that the motion from one frame to another is small enough. Then we reshape the model to $V(p)$ and map the image i (the new frame) onto the model. Then we have geometrically normalized the shape and got the resulting image as a vector. After that we have mapped the input image (i) on the model, and reshaped the model to the standard shape. Then we got the resulting normalized image as a vector as in (3.24) and then we have computed texture parameters from normalized image as in (3.25). Then we have computed synthesized texture as given in (3.26).

$$j(i,p) = j(i,V(p)) \tag{3.24}$$

$$\xi(i,p) = X^T(j(i,p) - \overline{x}) \tag{3.25}$$

$$x(i,p) = \overline{x} + XX^T(j(i,p) - \overline{x}) \tag{3.26}$$

We have computed residual image as in (3.27) and we have selected Summed square error (SSE) as a error measure as given in (3.28).

$$r(i,p) = j(i,p) - x(i,p) \tag{3.27}$$

$$e = \|r(i,p)\|^2 \tag{3.28}$$

For good model adaptation residual image and error e is much smaller. Then we have computed the update vector $\triangle p$ by multiplying the residual image with an update matrix U. The new error measure for updated parameter is as in (3.30).

$$\triangle p = \mathrm{U} r(p) \tag{3.29}$$

$$e_0 = \|r(i, p + \triangle p)\|^2 \tag{3.30}$$

If $e_0 < e$ we update $e_0 \rightarrow e$ and $(p + \triangle p) \rightarrow p$. If $e_0 > e$, we try smaller steps. Now e is recomputed as in (3.31).

$$e_k = \left\| r(i, p + \frac{1}{2^k} \triangle p) \right\|^2 \tag{3.31}$$

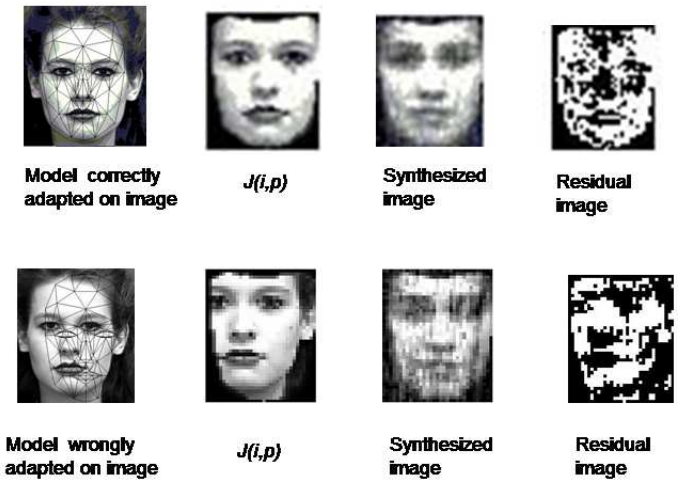

Figure 3.28: Candide wire frame model fitting on face image

For $k = 1, 2, 3, ...$ if $e_k < e$ we update $e_k \rightarrow e$ and $p + \frac{1}{2^k} \triangle p \rightarrow p$. We iterate the algorithm and declare the convergence when $e_k > e$. The model matching and texture approximation process is shown in Figure 3.28. A correct model adaptation is shown in top row, and wrong adaptation is shown in bottom row. The first image in both the rows shows a model adapted on face image. Second image in both the rows is a texture of face image mapped on geometrically normalized Candide wire frame model. The normalized texture is approximated by the eigentextures

producing the synthesized image. Residual image is computed by subtracting the normalized image and synthesized image. From the figure it is clear that normalized image and synthesized image are more similar for better model adaptation. Analysis of residual image tells us how to improve model adaptation [66].

3.4.1 Creating Update Matrix

We have assumed that $r(i,p)$ is linear in p. So, we write (3.32), where G is a gradient matrix

$$\frac{\partial}{\partial p} r(i,p) = G \tag{3.32}$$

By applying Taylor series expansion to $r(i,p)$around $(p + \triangle p)$, we got (3.33), where, O represents the higher order derivatives of $r(i,p)$.

$$r(i, p + \triangle p) = r(i,p) + G\triangle p + O(\triangle p^2) \tag{3.33}$$

$$e(i, p + \triangle p) = \|r(i,p) + G\triangle p\|^2 \tag{3.34}$$

We want to find $\triangle p$ that minimizes (3.34). Minimizing (3.34) is a least square problem whose solution is as in (3.35). From which we compute update matrix U as the negative pseudo inverse of the gradient matrix G as in (3.36).

$$\triangle p = -(G^T G)^{-1} G^T r(i,p) \tag{3.35}$$

$$\mathrm{U} = -(G^T G)^{-1} G^T \tag{3.36}$$

We have computed Gradient matrix G from training images in advance. j^{th} column in G is given as in (3.37) and the approximation is as in (3.38).

$$G_j = \frac{\partial}{\partial p_j} r(i,p) \tag{3.37}$$

$$G_j \approx \frac{r(i, p + hq_j) - r(i, p - hq_j)}{2h} \tag{3.38}$$

Here, h is the step size for perturbation. q_j is a vector with zero in elements except one in j^{th} column. We have adapted the wire frame model to every training image in the training set and computed the shape and texture modes. Thus, we have obtained, a set of corresponding parameter vectors p_n for a suitable step size to estimate G_j by averaging as in (3.39)

$$G_j \approx \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{r(i_n, p_n + khq_j) - r(i_n, p_n - khq_j)}{2h} \tag{3.39}$$

Here, N represents the number of training images while K represents the number of steps to perturb the parameter [66].

3.4.2 Experimental Results

We have constructed a separate update matrix only for scaling, translation and rotation parameters. Update matrix is constructed using (3.36). Where $N = 100$, and $K = 20$. For scaling, we have used $h = 100 \times 0.01$ in the range$[-hk, hk] = [-20, 20]$. For translation, we have selected h relative to the size of the model $h = 0.01 \times (s + 1)$ in the range $[-hk, hk] = [-0.2 \times (s + 1), 0.2 \times (s + 1)]$. For rotation, we have converted h into radians $h = 0.01 \times 180/\pi$ in the range $[-hk, hk] = [-11.459, 11.459]$. Wire frame model is manually fitted on 100 different face images. All parameters are varied according to the above steps, and residual images are collected. Gradient matrix is computed using (3.39). From gradient matrix update matrix is computed using (3.36). For unknown image, when model fits roughly, residual image is computed. Residual image gets multiplied by an update matrix to get updated parameter vector $p = [s, t_x, t_y, \theta_x, \theta_y, \theta_z]$. With this new parameter, the model gets deformed, again residual image is computed, and the process repeats till model fits exactly, where error e is reduced to a minimum. Once the model fits properly on the first frame, in the subsequent frames only animation parameters need to be processed. Results of wire frame fitting are shown in Figure 3.29

3.5 Wire frame fitting on tilted faces

In the previous section we have explained how the wire frame model automatically fits on the frontal faces. In this section we explain how the model is fitting automatically on tilted faces as well as on non neutral faces. Once the model is adapted properly on the first frame, for the subsequent frames only α is changing. We want to find the optimal adaptation of the wire frame model to the input image. So, our aim is to find p that minimizes the distance between the wire frame model and the image. We have obtained a set of feature points F from our feature point detection algorithm, see section 3.2. We have extracted 14 facial feature points, and each point has x and y coordinates, so F is vector of length 28×1. We know the vertices of wire frame model $\overline{V}$. Our goal is model adaptation, i.e., to find the deformed model V that fulfills

$$min \parallel V(p) - F \parallel^2 \tag{3.40}$$

Figure 3.29: Wire frame fitting on frontal faces results

In (3.40) out of 113 vertices, we have considered only (x, y) coordinates of 14 vertices corresponding to our 14 extracted feature points. Dimension of V is reduced to 28×1.

The action units and shape units can be applied to scaled model, which simplifies (3.4) as in (3.41). For small rotations we can write (3.42) and (3.43). Then we can write (3.41) as in (3.44)

$$V = R(s\overline{V} + S\sigma + A\alpha) + t \tag{3.41}$$

$$R\overline{V} = (\theta_x\theta_y\theta_z)\overline{V} = (I + r_x\theta_x)(I + r_y\theta_y)(I + r_z\theta_z)\overline{V} \tag{3.42}$$

$$R\overline{V} = (r_x\theta_x + r_y\theta_y + r_z\theta_z + I)\overline{V} \tag{3.43}$$

$$\begin{aligned} V =& (\sum_{i=1}^{3} \beta_i s_i)\overline{V} + \sum_{i=1}^{14} S_i\sigma_i + \sum_{i=1}^{65} A_i\alpha_i + (\sum_{i=x}^{y,z} r_i\theta_i)\overline{V}) \\ &+ \sum_{i=1}^{3} \tau_i t_i \end{aligned} \tag{3.44}$$

The terms in (3.44) are defined in section 3.3.1. We write (3.44) as matrix vector multiplication as $V = Cp$ where C defines the allowed deformations as in (3.45)

and p is the parameter vector as in (3.46). C is of size 28×88. (3+14+65+3+3) (scaling+shape+animation+rotation+translation). p is a parameter vector of size 88×1 which gives scaling, shape, animation, rotation and translation parameter values.

$$\begin{aligned} C =&[s_1\overline{V}, s_2\overline{V}, s_3\overline{V}, S_1, ...S_{14}, A_1, ... \\ &.....A_{65}, \theta_x\overline{V}, \theta_y\overline{V}, \theta_z\overline{V}, t_1, t_2, t_3] \end{aligned} \tag{3.45}$$

$$\begin{aligned} p =&[\beta_1, \beta_2, \beta_3, \sigma_1, ...\sigma_{14}, \alpha_1, ...\alpha_{65}, \\ &r_x, r_y, r_z, \tau_1, \tau_2, \tau_3] \end{aligned} \tag{3.46}$$

β_1, β_2 and β_3 represents scaling parameters while τ_1, τ_2 and τ_3 represents translation parameters. So (3.44) can be solved as in (3.47) which is a least square optimization problem whose solution is as in (3.48)

$$min \parallel V - F \parallel^2 = min \parallel Cp - F \parallel^2 \tag{3.47}$$

$$p = (C^T C)^{-1} C^T F \tag{3.48}$$

Instead of computing 88 parameter values at a time, first we find the solution only for global parameters using (3.49) and (3.50)

$$C_1 = [s_1\overline{V}, s_2\overline{V}, s_3\overline{V}, \theta_x\overline{V}, \theta_y\overline{V}, \theta_z\overline{V}, t_1, t_2, t_3] \tag{3.49}$$

$$p_1 = [\beta_1, \beta_2, \beta_3, r_x, r_y, r_z, \tau_1, \tau_2, \tau_3] \tag{3.50}$$

$C_1$is of size 28×9 i.e. scaling, rotation and translation. p_1 is a parameter vector of size 9×1 which gives scaling, rotation and translation parameters. We consider rotation of faces only in y direction. Initially we set $\theta_y = 0$, then we find p_1. Then model is geometrically normalized to standard shape, texture is mapped on this model, synthesized image is computed. Then we compute residual image, from which summed squared error (SSE) is computed. All these methods are explained in above sections. We repeat the procedure for $\theta_y = \pm 30^0, \pm 45^0, \pm 60^0$. The value of θ_y for which SSE is minimum, will give us rotating angle of face with respect to y axis. Then the shape unit parameters and animation unit parameters calculation is done using (3.51) and (3.52)

$$C_2 = [S_1, ...S_{14}, A_1, ...A_{65}] \tag{3.51}$$

$$p_2 = [\sigma_1, ...\sigma_{14}, \alpha_1, ...\alpha_{65}] \tag{3.52}$$

C_2 is of size 28×79 i.e. animation and shape. p_2 is a parameter vector of size 79×1 which gives shape and animation parameters. Once the model is adapted properly on the first frame, for the subsequent frames only animation parameters α will change. For neutral frame $\alpha_1, ...\alpha_{65} = 0$. If the frame is of

any expression other than the neutral then $\alpha_1, ...\alpha_{65} \neq 0$. From the analysis of different animation parameters we can recognize the facial expressions. By observing different animation parameters and their combinations, we can identify facial expressions. There are 65 animation units such as Lip stretcher, Jaw drop, eyebrow raiser, lip corner depressor etc. If the animation parameter value of Jaw drop and eyebrow raiser is approximately 1 then we can say the expression is a surprise. Similarly, if the lip stretcher animation parameter is 1 it could be a smile or happy expression. Previously, some researchers have identified facial expressions from a combination of animation units (or action units). For non neutral facial expression frame, once the model fits properly, then we set $\alpha_1, ...\alpha_{65} = 0$ so as to get shape of wire frame for neutral facial expression of the same face. So it is not necessary that first frame should be always of neutral facial expression. Model fitted on different faces, with different rotation is shown in Figure 3.30.

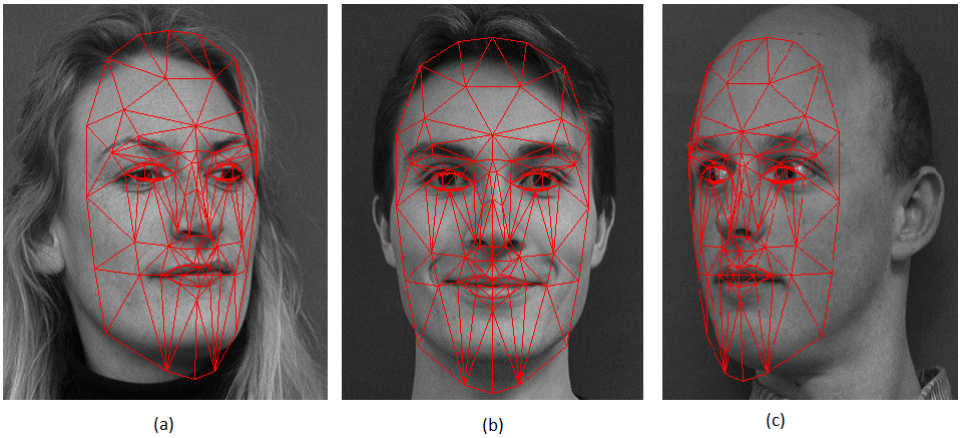

Figure 3.30: Model fitting on tilted faces

3.6 Extraction of wire frame grid node coordinates

The geometric information which, we have used is the displacement of one point. It is defined as the difference between the first and the last frame's node coordinates. We have constructed a feature vector, for every image sequence to be examined. Feature vector contains the geometrical displacement of every point taken into consideration. The feature vector is used as an input to a multi class Support Vector Machine, with seven classes. SVM classifies each set of wire frame grid node's geometrical displacements to one of the seven basic facial expressions happy, surprise, sadness, anger, fear, disgust and neutral. We have used Cohn Kanade and IMM database. This database consists of face images. It is clustered into seven different classes each one representing one of the seven basic facial expressions. We have used the geometrical information for facial expression recognition. The

displacement of one node d_{ij}, is the difference of the i^{th}grid node coordinates at the first and last frame of facial image sequence.

$$d_{ij} = [\triangle x_{ij} \quad \Delta y_{ij}]^T \tag{3.53}$$

Here, $i = 1, ..., E$ (total number of nodes =113) and $j = 1, ..., N$ (Number of facial image sequence = 25 per expression). $\triangle x_{ij}$, $\triangle y_{ij}$ are the (x, y) coordinate displacement of the i^{th} node in the j^{th}image respectively. Thus we have created a feature vector g_j, for every facial image sequence in the training set as in (3.54). The vector g_j is called grid deformation feature vector. It contains the geometrical displacement of every grid node.

$$g_j = [d_{1,j} d_{2,j} ..... d_{E,j}]^T \tag{3.54}$$

Here, $j = 1, ..., N$. The dimension of the vector g_j is $D = 113 \times 2 = 226$ dimensions. Out of 113 nodes, only 60 nodes contribute for facial expression recognition. So to reduce computations we have considered only 60 nodes. Now g_j is reduced to $60 \times 2 = 120$ dimension. Each grid deformation feature vector g_j belongs to one of the seven facial expression classes. This data is used to train multiclass SVM classifier, which is explained in the next chapter.

Chapter 4

Facial expression classification

Once the facial feature data is available in the form of geometrical deformation feature vectors, next task is to design a classifier that will classify this set of vectors into one of the basic facial expressions. We have employed Bayesian classifier and support vector machine. Support vector machine classifies data into two classes. We modified it for seven classes. We did analysis of results of all the classifiers. In this Chapter, we discuss about all these classifiers one by one alongwith results which we obtained. In results we have mentioned overall accuracy of FER system in tabular form using a particular classifier. It is not the accuracy of individual classifier, rather it is the accuracy of complete system including face detection, model fitting, tracking and classification.

4.1 Database

For training and testing samples of image sequences we have used Cohn Kanade database [59] and IMM database [60]. We divide the database into four sets containing 25% of the data for each class, chosen randomly. One set containing 25% of the samples for each class is used as test set, while the remaining three sets form the training set. After performing the classification procedure, the samples forming the testing set are merged into the current training set, and a new set of samples (25% of the samples from each class) is extracted to form the new test set. The remaining samples forms the new training set. This procedure is repeated four times. The average classification accuracy is computed as the mean value of the percentages of the correctly classified facial expressions.

4.1.1 Cohn Kanade database

Subjects in the released portion of the Cohn-Kanade AU-Coded Facial Expression Database are 100 university students. They ranged in age from 18 to 30 years.

Sixty-five percent were female, 15 percent were African- American, and three percent were Asian or Latino. Subjects were instructed by an experimenter to perform a series of 23 facial displays that included single action units and combinations of action units. Image sequences from neutral to target display were digitized into 640 $\times$ 480 or 490 pixel arrays with 8-bit precision for grayscale values. Included with the image files are "sequence" files; these are short text files that describe the order in which images should be read.

4.1.2 IMM database

The IMM Face Database consists of 240 images of 40 different human faces, all without glasses. The gender distribution is 7 females and 33 males. Images were acquired in January 2001 using a 640 x 480 JPEG format with a Sony DV video camera, DCR-TRV900E PAL. The facial structures that were manually annotated using 58 landmarks are eyebrows, eyes, nose, mouth and jaw.

4.2 Bayesian Classifier

Bayesian classifier is a simple probabilistic classifier based on applying Bayes theorem with strong independent assumptions. It assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. Bayes classifier is trained in a supervised learning setting. Bayesian learning methods are relevant to machine learning for two different reasons. First, they provide a useful perspective for understanding many learning algorithms that do not explicitly manipulate probabilities and second is Bayesian learning algorithms normally calculate the explicit probabilities for hypotheses, which is the most practical approach to certain types of learning problems. Researchers proved that the naive Bayes classifier is competitive with other learning algorithms in many cases. In some cases it outperforms other methods. One practical difficulty is that this classifier typically require initial knowledge of many probabilities. If these probabilities are not known in advance they are often estimated based on background knowledge and previously available data. Michie et al.[67] provide a detailed study comparing the naive Bayes classifier to other learning algorithms, including decision tree and neural network algorithms. One practical difficulty in applying Bayesian methods is that they typically require initial knowledge of many probabilities. When these probabilities are not known in advance they are often estimated based on background knowledge, previously available data, and assumptions about the form of the underlying distributions.

Features of Bayesian learning method includes

1. Each observed training example can incrementally decrease or increase the estimated probability that a hypothesis is correct.

2. Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.

3. Bayesian methods can accommodate hypotheses that make probabilistic predictions.

4. New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities [67] .

The probability model for a classifier is a conditional model $p(C \mid F_1, ..., F_n)$ over a dependent class variable C with a small number of outcomes or classes, conditional on several feature variables F_1 through F_n. Bayes theorem is

$$p(h \mid d) = \frac{p(h)p(d \mid h)}{p(d)} \tag{4.1}$$

Here, $p(h)$ is prior belief (probability) of hypothesis h before seeing any data. $p(d \mid h)$ is likelihood (probability) of the data if the hypothesis h is true. $p(h \mid d)$ is posterior probability of hypothesis h after having seen the data d). $p(d) = \sum p(d \mid h)p(h)$ data evidence (marginal probability of the data). Using this theorem we can write

$$p(C \mid F_1, ...F_n) = \frac{p(C)p(F_1, ...F_n \mid C)}{p(F_1, ...F_n \mid C)} \tag{4.2}$$

Here, the denominator does not depend on C. We are only interested in the numerator of that fraction. As, the values of the features F_i are given, the denominator is effectively constant. The numerator represents the joint probability model $p(C \mid F_1, ..., F_n)$. We rewrite it as in (4.3), using repeated applications of the definition of conditional probability

$$p(C \mid F_1, ...F_n) = p(C)p(F_1, ...F_n \mid C) \tag{4.3}$$

$$= p(C)p(F_1 \mid C)p(F_2, ...F_n \mid C, F_1) \tag{4.4}$$

$$= p(C)p(F_1 \mid C)p(F_2 \mid C, F_1)p(F_3, ...F_n \mid C, F_1, F_2) \tag{4.5}$$

Here, we have assumed that, each feature F_i is conditionally independent of every other feature F_j for $j \neq i$. This means that

$$p(F_i \mid C, F_j) = p(F_i \mid C) \tag{4.6}$$

So, we write the joint model as in (4.7)

$$p(C \mid F_1, .., F_n) = p(C)p(F_1 \mid C)p(F_2 \mid C)p(F_3 \mid C).... \tag{4.7}$$

$$p(C \mid F_1, .., F_n) = p(C) \prod_{i=1}^{n} p(F_i \mid C) \tag{4.8}$$

The classification decision is made using (4.9), where $C_i = 1, ..., 7$ is class label and f_i is feature related to each sample. Using this formula, a test grid deformation feature vector is classified to one of the seven facial expressions.

$$C = \text{argmax}\{P(C_i) \prod P(f_i \mid C_i)\} \tag{4.9}$$

From training samples we compute mean and covariance for each coordinate of geometrical deformation feature vector. Following equations are used to compute mean and variance.

$$\overline{y} = \frac{1}{n} = \sum_{i=1}^{n} y_i \tag{4.10}$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \overline{y})^2 \tag{4.11}$$

Where n is our number of wire frame node coordinates, it is 120. For computing probabilities with respect to each coordinate, we make use of Gaussian distribution as in (4.12)

$$P(x = v \mid C) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(v-\overline{y})^2}{2\sigma^2}} \tag{4.12}$$

For unknown sample we compute probabilities with respect to each attribute for all classes. The class which gives maximum value of probability is declared as a class of unknown sample as per (4.9). One practical difficulty is that this classifier typically require initial knowledge of many probabilities. If these probabilities are not known in advance they are often estimated based on background knowledge and previously available data. In our case we got less accuracy using this classifier. Results are given in Table 4.1. Overall accuracy of FER system is 74.5%. The table gives accuracy of complete system including face detection, model fitting, tracking and classification. It is not the accuracy of individual classifier. Table shows confusion matrices as well as accuracy, e.g. for happy expression, consider happy column, 64% of total samples are classified correctly as happy expression, 12% are missclassified as surprise, 8% are missclassified as sad, 12 % are missclassified as fear, and 4% are missclassified as neutral.

Table 4.1: Results of facial expression recognition using Bayesian classifier

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	64%	10%	0	0	0	10%	2%
Surprise	12%	80%	0	0	0	0	0
Sad	8%	0	78%	8%	12%	6%	0
Anger	0	0	0	72%	20%	10%	5%
Disgust	0	0	11%	9%	68%	0	0
Fear	12%	4%	11%	8%	0	74%	3%
Neutral	4%	6%	0	3%	0	0	90%

4.3 SVM classifier

Basically SVM is a binary classifier that classifies data into two classes. During the training phase it constructs separating hyperplanes between training samples of two classes, labeled as +1 and -1. The separating hyperplane that best separates the two classes is called the maximum-margin hyperplane and forms the decision boundary for classification. The data samples that lie at the boundary of each class are called support vectors (SVs). During the testing phase only support vectors are involved in the computation. When two data classes are not linearly separable, a kernel function is used to project data to a higher dimensional space, where linear classification is possible. This is known as the kernel trick and allows an SVM to solve nonlinear problems.

An n-dimensional pattern x has n coordinates, x = $(x_1, x_2, ..., x_n)$, where each $x_i \in R$ for $i = 1, 2, ..., n$. Each pattern x_j belongs to a class $y_i \in \{-1, +1\}$. Consider a training set T of m patterns together with their classes, where

$T = \{(\mathrm{x}_1, y_1), (\mathrm{x}_2, y_2), ..., (\mathrm{x}_m, y_m)\}$.

Consider a dot product space S in which the patterns x are embedded,

$\mathrm{x}_1, \mathrm{x}_2, ...\mathrm{x}_m \in S$. Any hyperplane in the space S can be written as

$$\{\mathrm{x} \in S \mid \mathrm{w}{\cdot}\mathrm{x} + b = 0\},\ \mathrm{w}{\in}S,\ b{\in}R \tag{4.13}$$

The dot product w·x is defined by

$$\mathrm{w}{\cdot}\mathrm{x} = \sum_{i=1}^{n} w_i x_i \tag{4.14}$$

A hyperplane w·x + b = 0 can be denoted as a pair (w, b). A training set of patterns is linearly separable if at least one linear classifier exists defined by the pair (w, b), which correctly classifies all training patterns. All patterns from class +1 are located in the space region defined by w·x + $b > 0$, and all patterns from class -1 are located in the space region defined by w·x + $b < 0$. Using the linear classifier defined by the pair (w, b), the class of a pattern x_k is determined with

$$\text{class}(\text{x}_k) = \left\{ \begin{array}{ll} +1 & \text{if w·x} + b > 0 \\ -1 & \text{if w·x} + b < 0 \end{array} \right\} \tag{4.15}$$

The distance from a point x to the hyperplane defined by (w, b) is

$$d(\text{x} \ ; \ \text{w}, b) = \frac{\mid \text{w·x} + b \mid}{\parallel \text{w} \parallel} \tag{4.16}$$

where $\parallel$ w $\parallel$is the norm of vector w. Of all the points on the hyperplane, one has the minimum distance $d_{\min}$ to the origin

$$d_{min} = \frac{\mid b \mid}{\parallel w \parallel} \tag{4.17}$$

We consider the patterns from the class -1 that satisfy the equality $\text{w} \cdot \text{x} + b = -1$ and that determine the hyperplane H_1, the distance between the origin and the hyperplane H_1 is equal to $\mid -1 - b \mid / \parallel \text{w} \parallel$. Similarly, the patterns from the class $+1$ satisfy the equality $\text{w} \cdot \text{x} + b = -1$ and that determine the hyperplane H_2. the distance between the origin and the hyperplane H_2 is $\mid +1 - b \mid / \parallel \text{w} \parallel$. Hyperplanes H, H_1, and H_2 are parallel and no training patterns are located between hyperplanes H_1 and H_2. So the margin of linear classifier H is $2/_{\|\text{w}\|}$. The optimum separation hyperplane conditions can be formulated into the following expression that represents a linear SVM, minimize $f(x) = \frac{\|\text{w}\|^2}{2}$ with the constraints $g_i(\text{x}) = y_i(\text{w} \cdot \text{x}_i + b) - 1 \geq 0, i = 1, ..., m$. The optimization problem represents the minimization of a quadratic function under linear constraints (quadratic programming). A convenient way to solve constrained minimization problems is by using a Lagrangian function. When the Lagrange function is introduced, a Lagrange multiplier λ_i is assigned to each training pattern via the constraints $g_i(\text{x})$. The training patterns from the SVM solution that have $\lambda_i > 0$ represent the support vectors. The training patterns with $\lambda_i = 0$ can be removed from training without any effect, because they are not important in obtaining the SVM model. After training, the classifier is ready to find the class membership for new patterns, those are different from training patterns. The class of a pattern x_k is determined with

$$\text{class } (\text{x}_k) = \left\{ \begin{array}{lll} +1 & if & \text{w} \cdot \text{x}_k + b > 0 \\ -1 & if & \text{w} \cdot \text{x}_k + b < 0 \end{array} \right\} \tag{4.18}$$

Only on the sign of the expression w·x + b is required for the classification of new patterns. To classify new patterns, we will use the support vectors from the training set and the corresponding values of the Lagrange multipliers $\lambda_i$without computing the vector w explicitly.

$$\text{class } (\text{x}_k) = \text{sign} \left(\sum_{i=1}^{m} \lambda_i y_i \text{x}_i \cdot \text{x}_k + b \right) \tag{4.19}$$

Training samples those they are not support vectors do not affect the classification

of new patterns. For the classification of a new pattern x_k, we compute the dot product between x_k and every support vector.

Some common kernels include:

1. Linear : $K(\vec{x}, \vec{z}) = (\vec{x}.\vec{z})$

2. Polynomial : $K(\vec{x}, \vec{z}) = (1 + (\vec{x}.\vec{z}))^d$

3. Sigmoid : $K(\vec{x}, \vec{z}) = \tanh((\vec{x}, \vec{z}) + \theta)$

4. Radial Basis Function :$K(\vec{x}, \vec{z}) = \exp(||(\vec{x} - \vec{z})||^2/(2\sigma^2))$

Basically SVM is a binary classifier, which classifies data in to two classes. To use it for multiclasses, mainly two schemes are used.

4.3.1 *One against One multi-class SVM classifier*

In this approach $C(C-1)/2$ classifiers are constructed. Where C is the number of classes. Classifier i, j is trained using all patterns from class i as positive instances, and all patterns from class j as negative instances and disregarding the rest. To combine obtained classifiers a simple voting scheme is used. When classifying a new instance each one of the base classifier casts a vote for one of the two classes used in its training. The class that gets maximum vote will be declared as class of new instance. In our case we need 21 classifiers for 7 expressions. The decision function is given by following equation

$$D(\overline{x}) = (\sum_{i=1}^{m} \alpha_i y_i K(\overline{x} \bullet \overline{s_i}) + b) \tag{4.20}$$

Where $\alpha_i$is weight of support vectors, $y_i$is class label, $s_i$is support vector, x is input vector and b is bias value. We have used Linear kernel, Polynomial kernel and RBF kernel. Results using Linear Kernel are given in Table 4.2. Overall accuracy of the FER system is 76.57%. Results using Polynomial kernel are given in Table 4.3. Overall accuracy of FER system is 68.85%. Results using RBF kernel are given in Table 4.4. Overall accuracy is 66.85%.

Table 4.2: Results of facial expression recognition using One vs One SVM with linear kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	68%	0	0	0	0	10%	0
Surprise	8%	80%	0	0	0	0	0
Sad	0	8%	74%	0	0	10%	0
Anger	4%	0	0	84%	13%	0	0
Disgust	0	6%	11%	16%	65%	0	5%
Fear	12%	6%	11%	10%	22%	75%	5%
Neutral	8%	0	4%	0	0	5%	90%

Table 4.3: Results of facial expression recognition using One vs One SVM with polynomial kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	62%	15%	0	0	0	0	10%
Surprise	12%	75%	0	0	0	0	10%
Sad	0	0	70%	0	0	0	0
Anger	10%	0	0	64%	16%	0	0
Disgust	0	0	10%	36%	60%	20%	0
Fear	0	10%	10%	0	24%	71%	0
Neutral	16%	0	10%	0	0	9%	80%

Table 4.4: Results of facial expression recognition using One vs One SVM with RBF kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	60%	16%	0	0	0	0	18%
Surprise	18%	78%	0	0	0	12%	0
Sad	0	0	62%	10%	0	0	0
Anger	0	0	0	66%	34%	0	0
Disgust	0	0	0	24%	55%	23%	0
Fear	12%	0	25%	0	11%	65%	0
Neutral	10%	6%	13%	0	0	0	82%

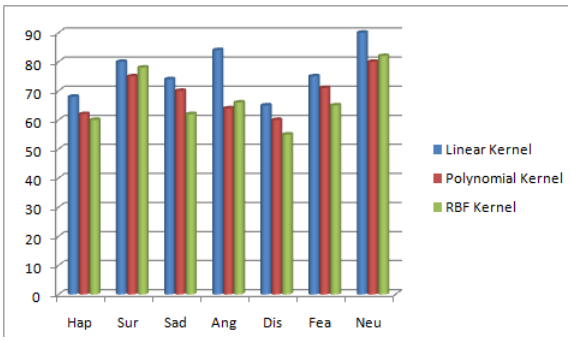

Figure 4.1: Graph of SVM One Vs One with all Kernels

4.3.2 Binary SVM Tree

To solve multiclass problem binary SVM is used in a hierarchical structure, called binary SVM tree. In binary SVM tree data set is divided into two subsets from root to the leaf until every subset consists of only one class. Only C -1 binary classifiers are constructed. Where C is number of classes. The steps in constructing binary tree are as follows [68].

1. Compute class centers using $m_i = \frac{1}{n}\sum x_t$ where m_i is mean of class i. x_t is training samples of i^{th} class.

2. Compute distance between classes using $D_{ij} = m_i - m_j$.

3. Compute radius of hyper-sphere in feature space using

$$R_i = \max_{t = 1.....I_j} \parallel x_t - m_i \parallel \tag{4.21}$$

4. Compute class similarity using

$$\text{similarity}(i,j) = \frac{R_i^2 + R_j^2}{\parallel m_i - m_j \parallel^2} \tag{4.22}$$

5. Select the two classes with biggest similarity to train SVM and unit these two classes.

6. Compute new united class center and repeat step 2, 3 and 4.

7. Construct the most upper SVM of binary tree when the number of the class equals to 2.

Table 4.5: Results of facial expression recognition using Binary Tree SVM with Linear Kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	69%	0	0	0	0	15%	3%
Surprise	8%	60%	25%	0	0	0	0
Sad	0	25%	55%	0	0	5%	0
Anger	0	0	0	82%	37%	0	0
Disgust	0	0	10%	13%	52%	0	0
Fear	22%	7%	10%	5%	11%	80%	3%
Neutral	1%	8%	0	0	0	0	94%

Table 4.6: Results of facial expression recognition using Binary Tree SVM with Polynomial Kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	61%	12%	0	0	0	15%	6%
Surprise	15%	70%	0	0	0	0	0
Sad	0	0	63%	0	0	15%	0
Anger	0	0	0	62%	35%	0	0
Disgust	0	0	13%	34%	60%	0	0
Fear	18%	10%	24%	0	5%	70%	10%
Neutral	6%	8%	0	4%	0	0	84%

Table 4.7: Results of facial expression recognition using Binary Tree SVM with RBF Kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	60%	13%	0	0	0	25%	10%
Surprise	8%	67%	0	0	0	0	0
Sad	0	0	58%	0	0	10%	0
Anger	0	0	0	57%	32%	0	0
Disgust	0	0	12%	33%	61%	0	0
Fear	16%	12%	28%	6%	5%	65%	10%
Neutral	16%	8%	2%	4%	2%	0	80%

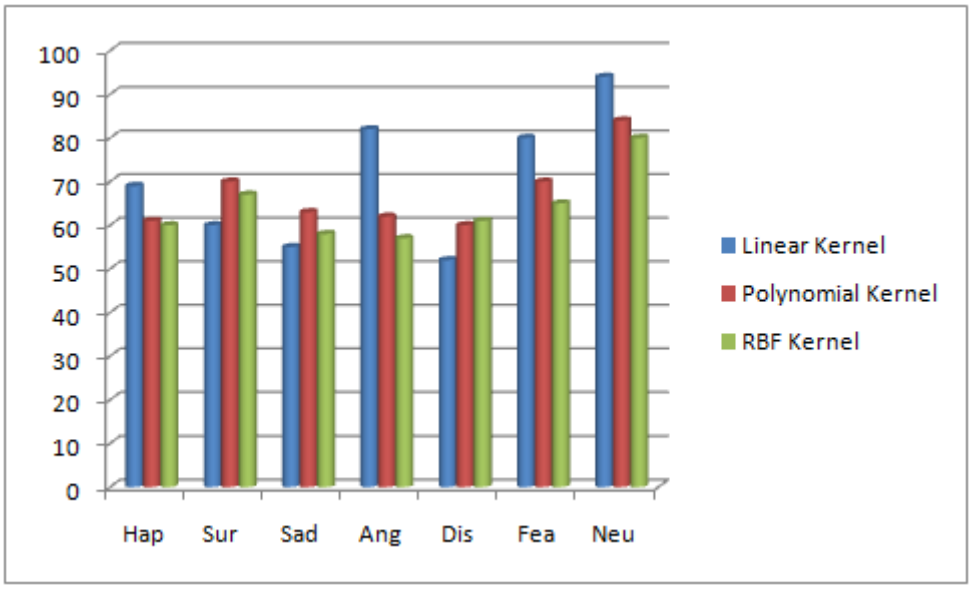

Figure 4.2: Graph of Binary SVM tree with all kernels

Results of facial expression recognition using Binary SVM tree with linear kernel are shown in Table 4.5. Overall accuracy is 70.28%. Results of facial expression recognition using Binary SVM tree with polynomial kernel are shown in Table 4.6. Overall accuracy is 61.14%. Results of facial expression recognition using

Binary SVM tree with RBF kernel are shown in Table 4.7. Overall accuracy is 64%. Graph comparing accuracy of classifier with Linear, Polynomial and RBF kernel is shown in Figure 4.2. SVM tree which we obtained with above algorithm is shown in Figure 4.3.

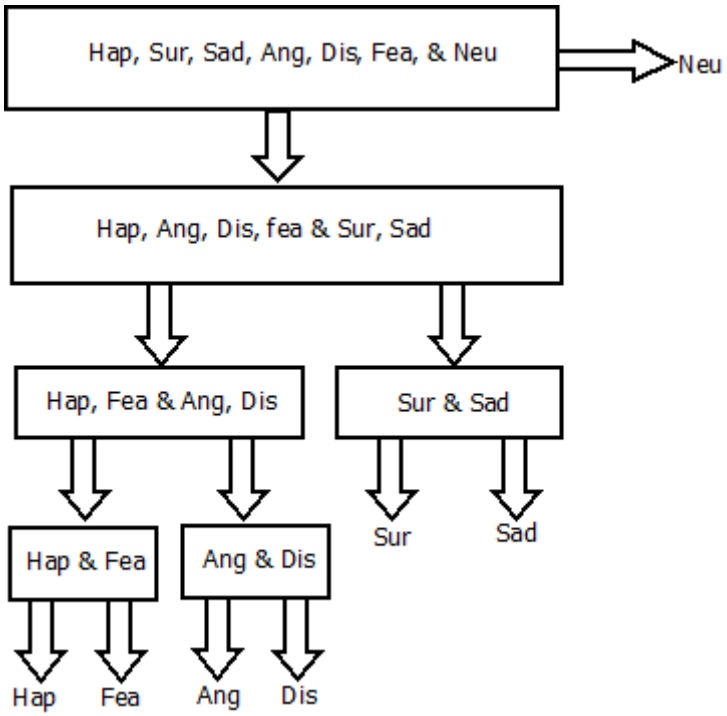

Figure 4.3: Binary SVM tree for seven expressions

4.3.3 One vs All SVM

In this scheme there is one binary SVM for each class to separate members of that class from members of other classes. We have number of classifiers equal to number of classes. Classifier i, j is trained using all patterns from class i as positive instances, and all patterns from rest of the classes is assumed to be in class j as negative instances. The class for which decision function gives maximum value will be declared as class of new instance. The decision function is given as in (4.23). We compute decision function for all seven classes, the class which gives maximum value is considered as a class of unknown sample.

$$D(\overline{x}) = (\sum_{i=1}^{m} \alpha_i y_i K(\overline{x} \bullet \overline{s_i}) + b) \tag{4.23}$$

Using this procedure, a test grid deformation feature vector is classified to one of the seven facial expressions. The results are given in Table 4.8. Overall accuracy is 78.14%. Results with Polynomial kernel are given in Table 4.9. Overall accuracy is 73%. Results with RBF kernel are given in Table 4.10. Overall accuracy is 66.28%.

Graph comparing accuracy of classifier with Linear, Polynomial and RBF kernel is shown in Figure 4.4.

Table 4.8: Results of facial expression recognition using One Vs All SVM with Linear Kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	76%	0	10%	0	0	0	0
Surprise	8%	90%	0	0	0	0	4%
Sad	0	0	73%	8%	28%	15%	0
Anger	0	0	17%	84%	0	0	0
Disgust	8%	6%	0	8%	62%	15%	4%
Fear	8%	0	0	0	0	70%	0
Neutral	0	4%	0	0	10%	0	92%

Table 4.9: Results of facial expression recognition using One Vs All SVM with Polynomial Kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	62%	8%	14%	0	0	0	3%
Surprise	8%	82%	0	0	0	0	2%
Sad	0	0	68%	10%	21%	19%	0
Anger	0	0	15%	78%	3%	0	0
Disgust	10%	6%	0	12%	67%	16%	6%
Fear	10%	0	0	0	0	65%	0
Neutral	10%	4%	3%	0	9%	0	89%

Table 4.10: Results of facial expression recognition using One Vs All SVM with RBF Kernel

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	64%	10%	0	0	0	0	10%
Surprise	10%	68%	0	0	0	0	4%
Sad	0	0	66%	18%	25%	25%	0
Anger	0	0	14%	71%	10%	10%	0
Disgust	10%	16%	20%	11%	60%	10%	6%
Fear	10%	0	0	0	0	55%	0
Neutral	6%	6%	0	0	5%	0	80%

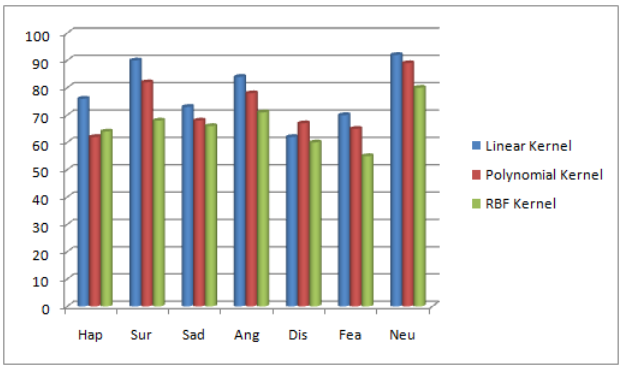

Figure 4.4: Graph of SVM One Versus All with all kernels

4.4 Comparison of Classifiers

Comparison of all above classifiers is given in tabular form in the Table 4.11. From this comparison, we observe that One vs All SVM gives maximum accuracy for complete system. If only classifiers accuracy is considered, still one vs all gives maximum accuracy. Figure 4.5 shows comparison of accuracy of all classifiers with linear kernel. If the computation time required for classifier is considered Bayesian classifier consumes less time, but its accuracy is low and computational complexity is high. In case of One vs All precomputations required are less, its execution time is next to Bayesian and accuracy is high. Hence, we chose One vs All SVM for hardware implementation.

Table 4.11: Comparison of the methods

Scheme	***Accuracy of FER system***	***Computation time required for classifier***	***No. of classifiers/ Efforts in computation***
Bayesian classifier	74.57%	0.84 s	*Computation of probabilities w.r.t. eac attribute*
One against one SVM	76.5%	1.24 s	$C(C-1)/2$ i.e. 21 classifiers required
Binary SVM tree	70.28%	1.4 s	6 classifiers required
One against all SVM	78.14%	1.12 s	7 classifiers required

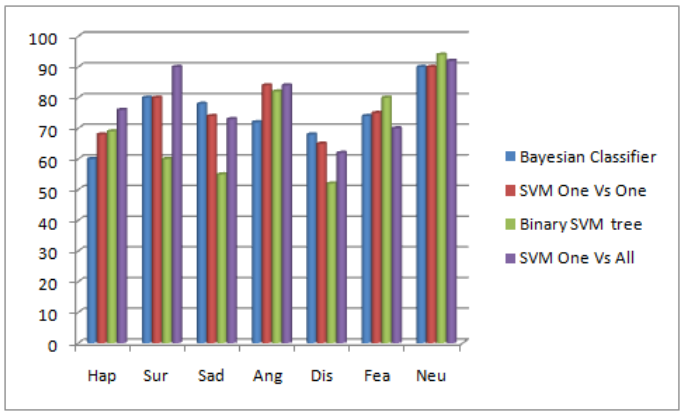

Figure 4.5: Graph showing accuracy of all classifiers

Chapter 5

Hardware Implementation of facial expression recognition algorithm

5.1 Motivation for hardware implementation

With the onset of automation in all the fields, computer vision has got a larger to play. Normally, these computer vision systems are implemented using a typical personal computers. Using a personal computer is easy, as it has been used for many years, and the technology is widely known. But if the performance is considered, the personal computer is not necessarily the right choice, because when the complexity of a vision system increases, the frame rate at which the personal computer is capable of processing images in real-time, decreases. This is due to the sequential operation mode of the central processing unit of personal computer. Many of the demanding tasks of a machine vision systems are trivial low level algorithms, where the same instructions are applied to each pixel in the frame. We show that these low level algorithms could be implemented more efficiently on a parallel structure such as two dimensional array of processing elements (PEs) mapped on the Field Programmable Gate Array (FPGA). In the framework for feature recognition, wire frame model fitting, tracking and classification are the important steps, where large matrix computations are involved. The parallel computation in the form of pipelined structures, ensures that the computation as close as possible to real time. For classification, we have selected One Vs All SVM with linear kernel for FPGA implementation, as we got maximum accuracy using this approach.

5.2 FPGA

Image processing is difficult to achieve on a serial processor like CPU in PC. This is due to the fact that large data set is required to represent the image and the complex operations to be performed on the image. Consider video rates of 15 frames per second, a single operation performed on every pixel of a 640×560 color image requires 16 million operations per second. In many image processing algorithms, there are several operations need to be performed on each pixel in the image. This results in large number of operations per second. These operations take large time if performed sequentially. Thus the only alternative is to make use of an FPGA, so that we can perform these operations in parallel. An FPGA consists of a matrix of logic blocks that are connected by an interconnect network. Both the logic blocks and the interconnect network are reprogrammable, thus allowing application specific hardware to be constructed, while at the same time maintaining the ability to change the functionality of the system with ease. As such, an FPGA offers a compromise between the flexibility of general purpose processors and the hardware based speed of ASICs.

The structure of FPGA is shown in Figure 5.1. Configurable logic blocks (CLBs) are the heart of the FPGA. CLBs are arranged in rows and columns, like matrix, and implement the logic functions specified by the programmer. Most CLBs perform this with a lookup tables (LUTs), which are digital memory arrays that contain truth tables for any logic function that can be implemented by the given number of logic inputs for a CLB. The output of the CLB is the logical result of the function recorded in the look up table. In order to program the CLBs, truth tables be loaded into the LUTs of each CLB .

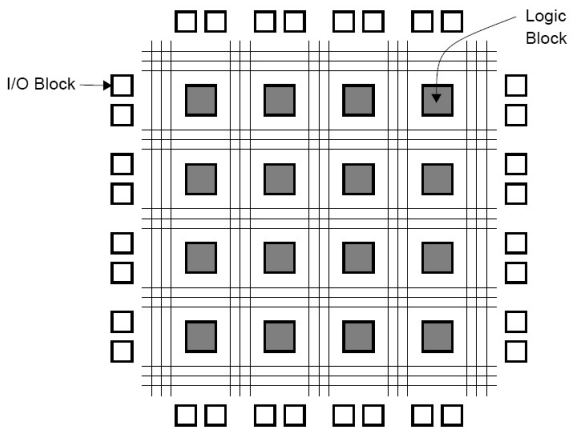

Figure 5.1: FPGA

5.3 Implementation of algorithm

We implement the complete algorithm for feature recognition by dividing it into two parts. First part includes wire frame model fitting and tracking while other part includes classification using One vs All support vector machine classifier.

5.3.1 Implementation of wire frame model fitting and tracking

We have developed a hardware solution for the fitting and tracking subsystem in order to enhance the speed of the system since real time performance is of paramount importance here. Implementation of wire frame model fitting and tracking is again subdivided into model fitting and tracking. The implemented design consists of two main units named fitting and tracking. Model unit distorts the wire frame model on the basis of the parameters fed to it. The Grid Displacement unit calculates the difference between the two wire frame models and save the result in disp.txt as shown in Figure5.8. Initially the fitting unit performs the placement of the wire frame model on the first image of the image sequence and updates the placement iteratively till the model is correctly fitted to the face. Then the tracking unit performs the tracking of facial expressions in successive frames that follow in the image sequence. Finally, with the help of Grid Displacement unit, the computed difference between the neutral image of face and the image containing the greatest intensity of expressions is given as input to the classifier which classifies the present expression into the basic classes of expressions.

5.3.1.1 Systolic array architecture

Systolic Arrays are regular arrays of simple Processing Elements (PEs), where each processing element in the array is identical. The systolic algorithm relies on data from different directions arriving at PEs in the array at regular intervals and being combined. This combination means any computation like successive multiplication or addition etc. SIMD (Single instruction multiple data) array is a synchronous array of PEs under the supervision of one control unit and all PEs receive the same instruction broadcast from the control unit but operate on different data sets from distinct data streams. SIMD array usually loads data into its local memories before starting the computation. SIMD array usually loads data into its local memories before starting the computation. Systolic arrays usually pipe data from an outside host and also pipe the results back to the host. Systolic arrays show both pipelining and parallel computation. Systolic Arrays are the architectures preferably used in the matrix multiplication operations. This chain of

PEs operates in a pipelined manner and can be expanded vertically or horizontally with minor modifications to operate in a systolic manner. In our algorithm large matrix computations are involved, so we make use of systolic arrays, that will reduce the number of resources and time also. Structures of systolic arrays are shown in Figure 5.2. Each PE consists of three registers $R_a, R_b$and R_c. These registers perform multiplication and addition in one unit of computational time, as shown in (5.1). The unit of computational time is time required to perform one addition and one multiplication i.e., $t_a + t_m$.

$$R_c = R_c + R_a * R_b \tag{5.1}$$

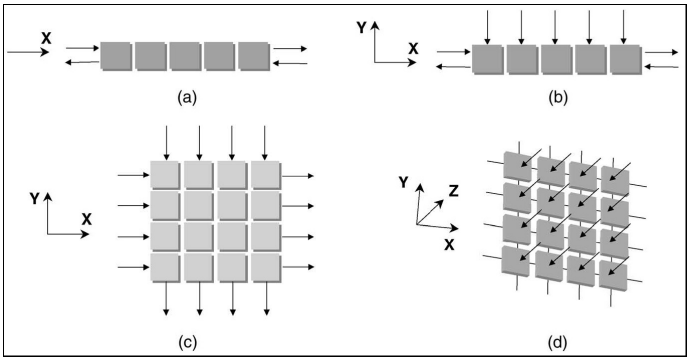

Figure 5.2: Systolic array

Suppose, we have two matrices A and B. We want to perform the computation $C = A * B$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{5.2}$$

We perform the matrix multiplication as (5.3) in iterations

$$C^{(k)} = C^{(k-1)} + A_k * B_k \tag{5.3}$$

Here, $k = 1, 2, ...n$ where $n = 3$ for above example. These iterations are performed according to the concept of a wavefront. Successive pipelining of the wavefronts is accomplished to compute all iterations. When the first wave propagates, we execute the second iteration in parallel by pipelining a second wavefront immediately after the first. Computational activity propagates to the neighboring PE. Total time required to transfer the result out of the array is $T = 4n - 2$. Time required when result stay in the array is $T = 3n - 2$. Number of PEs required are

n^2. Structure of such array is shown in Figure 5.3.

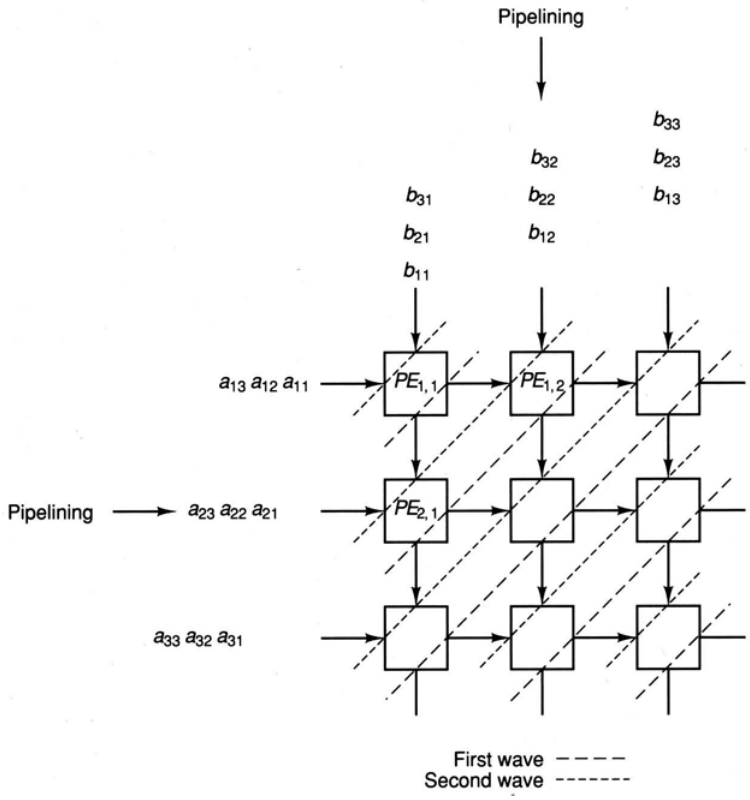

Figure 5.3: Systolic array architecture

5.3.1.2 Implementation of model fitting

We have used the output of the face detection algorithm to get the inputs to the hardware design we have proposed. These inputs are the location of the face in the image, x, y co-ordinates of the center of the face and the height and width of the face. For the output of the face detection algorithm, refer Figure 7.1. Using the height and width given by the face detection algorithm, we scale the wire frame model. This scaling is done on the basis of estimated equation developed during manual test runs. Then this scaled model is placed on the image at the x, y coordinates returned by the face detection algorithm. This process is a part of model re-structuring. After this, Geometrical normalization of the face is done to obtain its normalized texture, thereby removing texture variations caused by its local and global motions and geometrical differences between individual. We are working with 33×40 pixel images, which are conveniently small and effective for texture mapping. Texture mapping here basically means texture from the original image is mapped to the normalized image. For texture mapping barycentric coordinate concept of triangle is used, as our wire frame model consists of triangles. Barycentric coordinates of triangles are precomputed and fed to

our system. Source coordinates are computed for input image and then texture mapping is performed. Flow diagram for model fitting is shown in Figure 5.4.

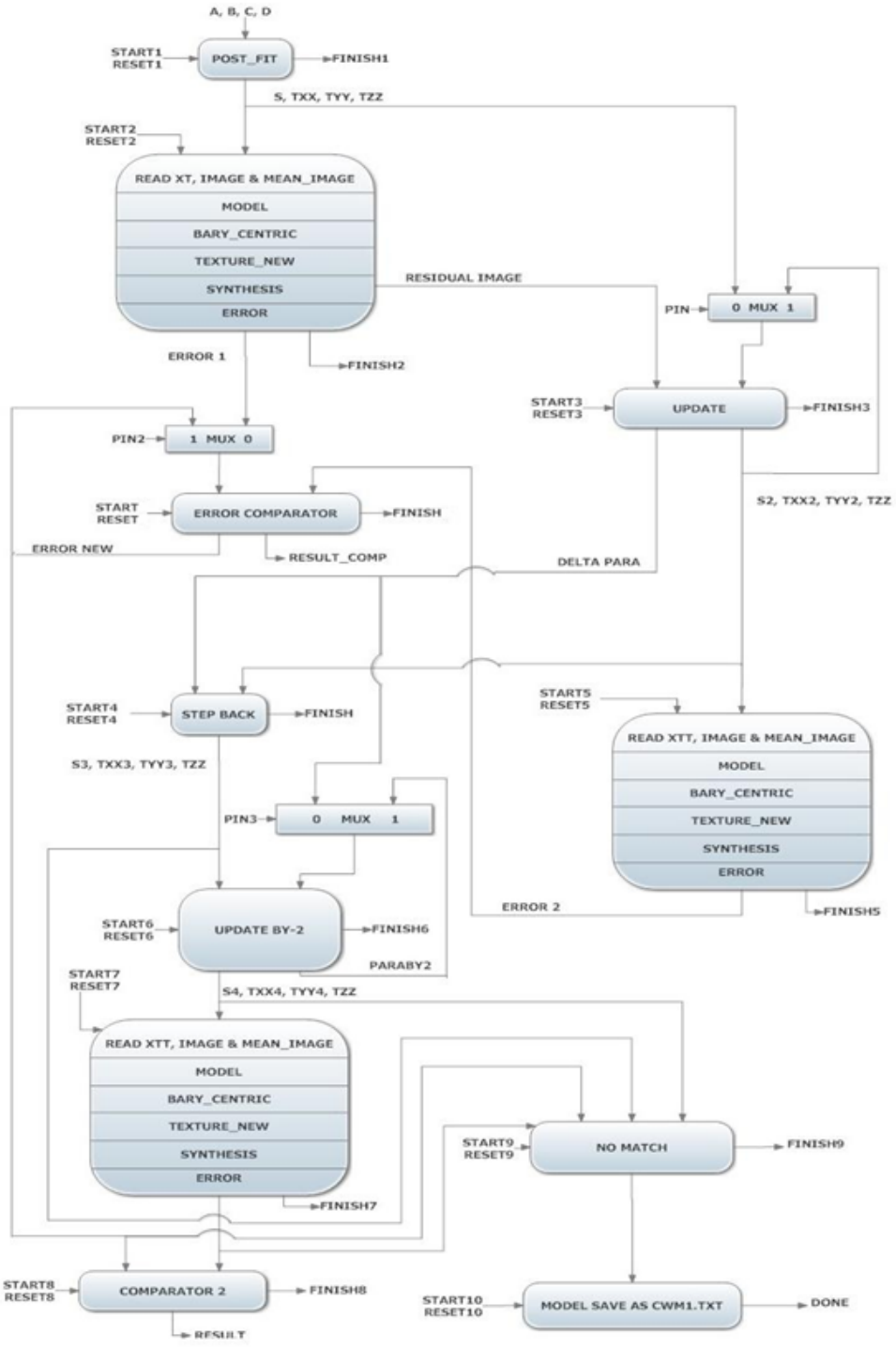

Figure 5.4: Flow diagram of model fitting subsystem

Next step is to compute synthesized image as in (3.26). Mean image $\overline{x}$ is precomputed. Matrix $XX^T$is also precomputed. Next step is to compute residual image $r(i,p)$, using the geometrically normalized image $j(i,p)$ and the synthesized image

$x(i,p)$. Summed square error (SSE) is selected as error measure. For good model adaptation residual image and error e is much smaller. The error is computed as in (3.28). Then we find the update vector ∇p by multiplying the residual image with an update matrix U, and the new error measure for updated parameter is calculated as per Eq 3.30. If $e_0 < e$ we update $e = e_0$ and $p = p + \nabla p$ if not, try smaller steps. e is recomputed as in (3.28). Iterate the scheme and declare the convergence when $e_k > e$. Control flow diagram for the model fitting is shown in Figure 5.5.

Fitting consist of 11 components. These are named post_fit, model, bary_centric, texture_new, synthesis, error, error comparator, update, update by 2, comparator2 and no match.

1. Post_fit – It estimates the initial values of the scaling unit and translation vector with the help of the outputs (x coordinate, y coordinate, height, width) of the face detection algorithm. Using x and y coordinates of the face in the image, we calculate the translation vector, and using the height and width of the face, we compute the scaling unit.

2. Model – It reshapes the wire frame model on the basis of the parameters fed as input to it.

3. Bary_centric - From precomputed barycentric coordinates, it computes source coordinates.

4. Texture-new – From source coordinate texture mapping is done on geometrically normalized image.

5. Synthesis – It computes synthesized image from mean image, geometrically normalized image and eigen textures. Then the residual image is computed. Wire frame model is fitted manually on training images. Each image is geometrically normalised to standard shape. Each image is converted to a vector. So number of images will construct a matrix. PCA is applied on this matrix and mean texture as well as eigen textures are computed and stored in memory.

6. Error – It calculates the Sum Squared Error using the residual image.

7. Error Comparator – It compares error values of two runs and takes decision of whether to update in parameter values is acceptable or not.

8. Update – Finds the update vector ∇p by multiplying the residual image with an update matrix. Update matrix is constructed from training images.

Wire frame model is fitted manually on training images. Model is geometrically normalized to standard shape. Then one by one all parameters are varied in 20 steps and residual images are collected. From these residual images average variation for each parameter is calculated. That will give us gradient matrix, from which we will construct update matrix. Update matrix is computed in MATLAB and it is stored in memory. Accordingly, the scaling factor and translation vectors are updated to a new value which would be used to compute a new set of grid nodes of the wire frame model and thus, a new value of error. Step back If this computation has resulted in a larger error than the previously calculated error, then the scaling factor and translation vectors are restored to their previous values corresponding to the smaller error with the help of this process.

9. Update by 2 – It reduces the step size of the update vector by 2.

10. Comparator2 – It compares error given after reducing step size and error with original step size. If new error is smaller then, the update in step size is accepted else ignored.

11. No Match – This component passes the parameters which result in minimal error as output.

The control flow of the fitting subsystem shown in Figure 5.5 is completed in 15 states. After initially resetting the system, start signal is given to post_fit unit to calculate s, txx, tyy, and tzz. The control is passed on to the next state which initially reads the files XT, image (over which the wire frame has to be fitted) and mean_image. Then model, bary_centric, texture _ new, synthesis and error computing units kicks into action in the same state and gives an initial fitted version of the wire frame model as well as primary error. With the commencement of the next state, control is forwarded to update unit, which computes s, txx, tyy, and tzz depending on $\triangle p$. These changed values are passed on to the next system to calculate new error. If this new error is less than the previous error, then the above stated process is repeated iteratively until new error is less than previous error at which the control is passed to step back unit which goes back a step to regain previous s, txx, tyy, and tzz. It then enters a loop to calculate the new values based on half of ∇p which iterates upon two conditions, either a new error becomes less than previous one or count becomes 4. Lastly, the control enters no match unit where the parameters corresponding to minimal error are used to compute almost perfect fitted model.

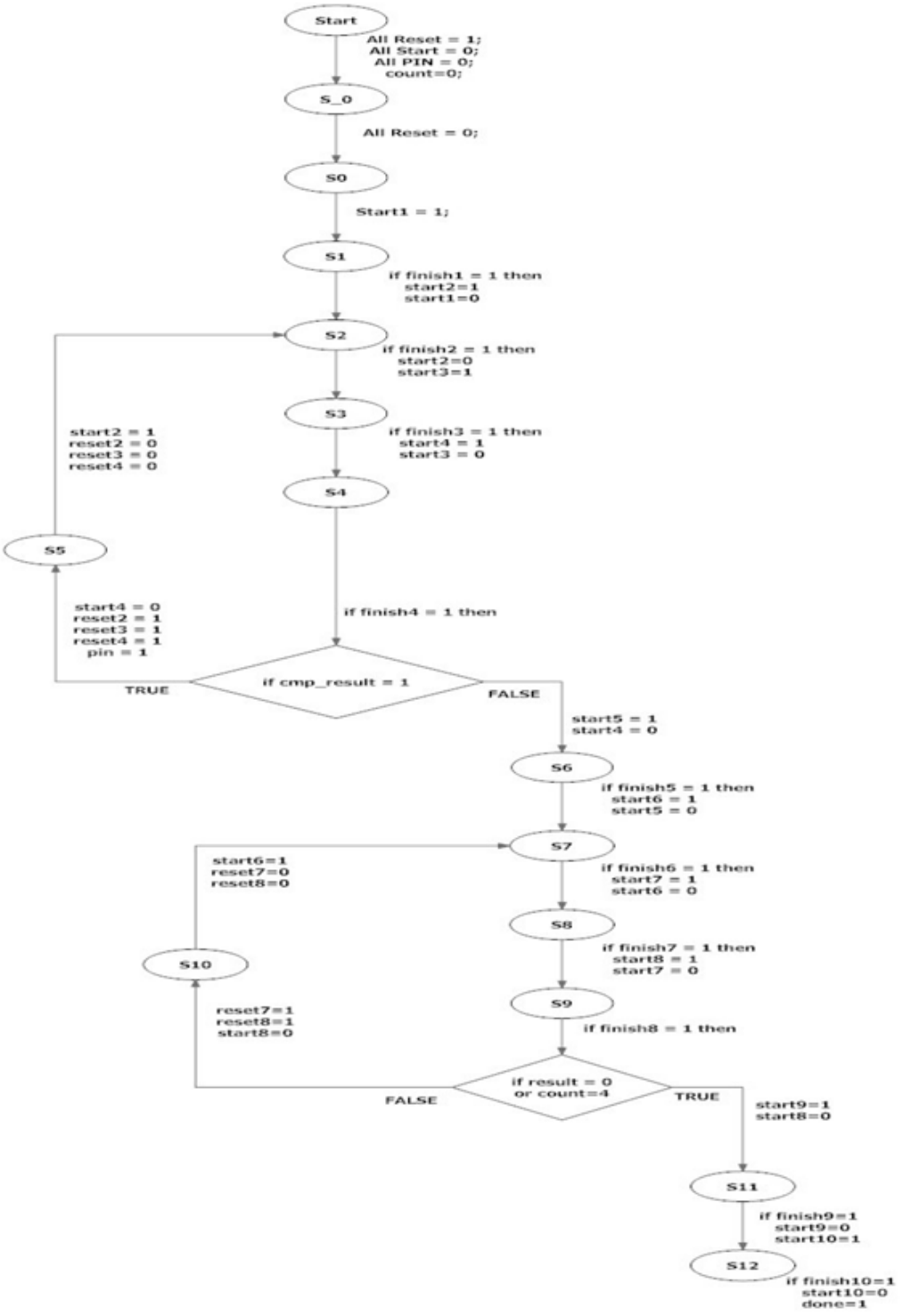

Figure 5.5: Control flow diagram of model fitting subsystem

5.3.1.3 Tracking subsystem

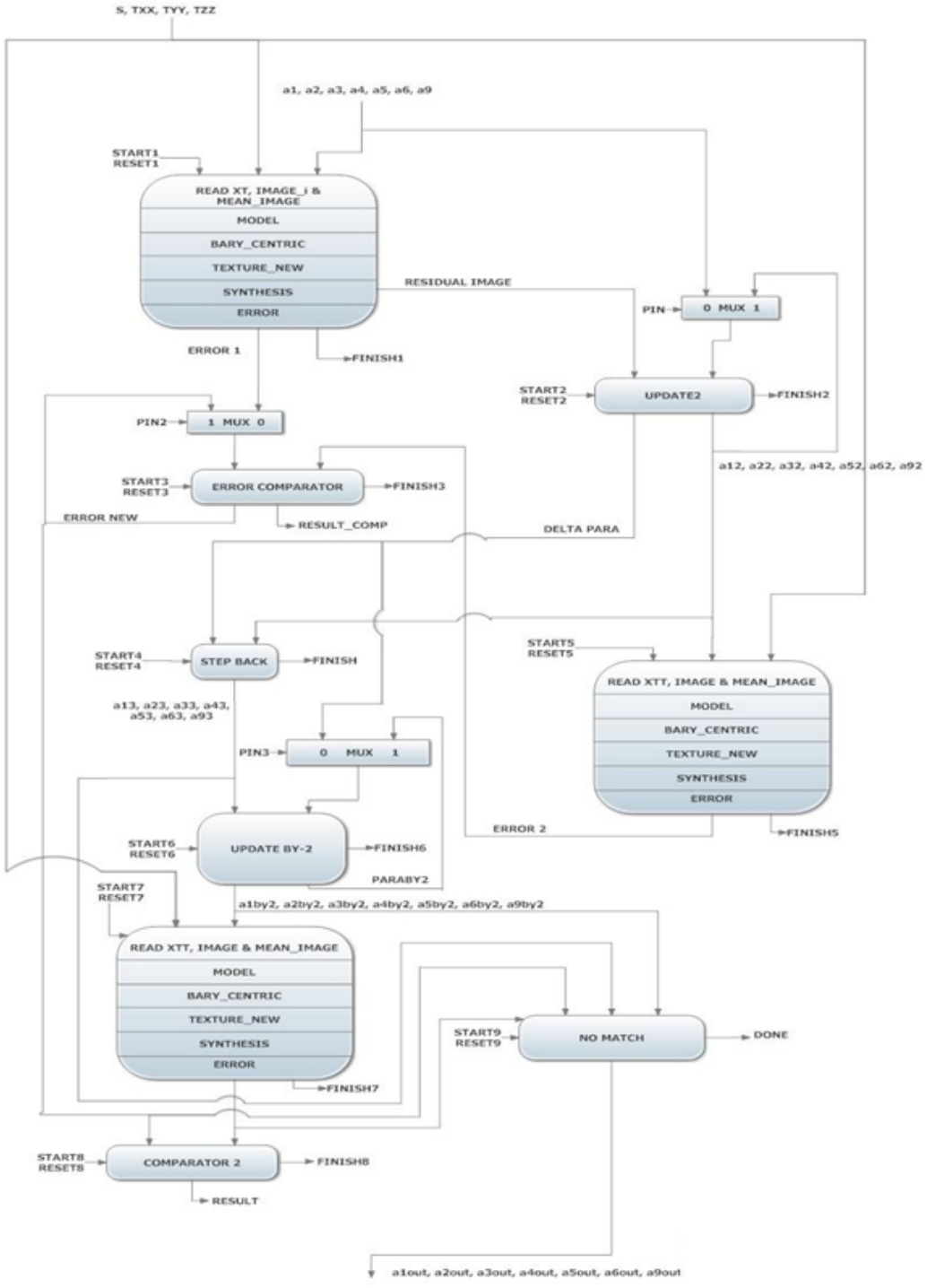

Figure 5.6: Flow diagram for tracking subsystem

Once the wire frame model fits on first frame, in subsequent frames only animation parameters will change. So during tracking s, txx, and tyy remain constant, only animation parameters are going to change. Analysis of residual image tells us how to improve model adaptation. During tracking again all process i e., compu-

tation of geometrically normalized image, synthesized image, residual image and error computations are performed. Figure 5.6 shows a flow diagram for tracking subsystem. Control flow diagram of tracking subsystem is shown in Figure 5.7.

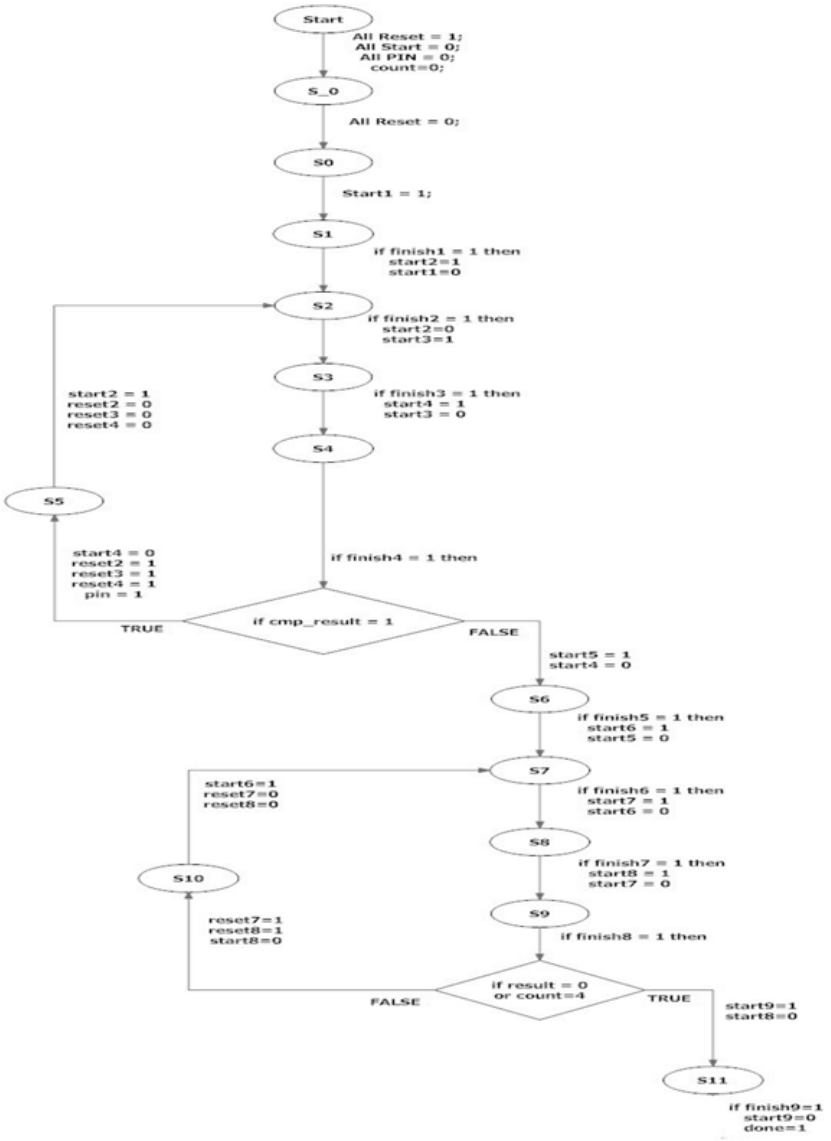

Figure 5.7: Control flow diagram of tracking subsystem

Complete flow diagram of designed system is shown in Figure 5.8

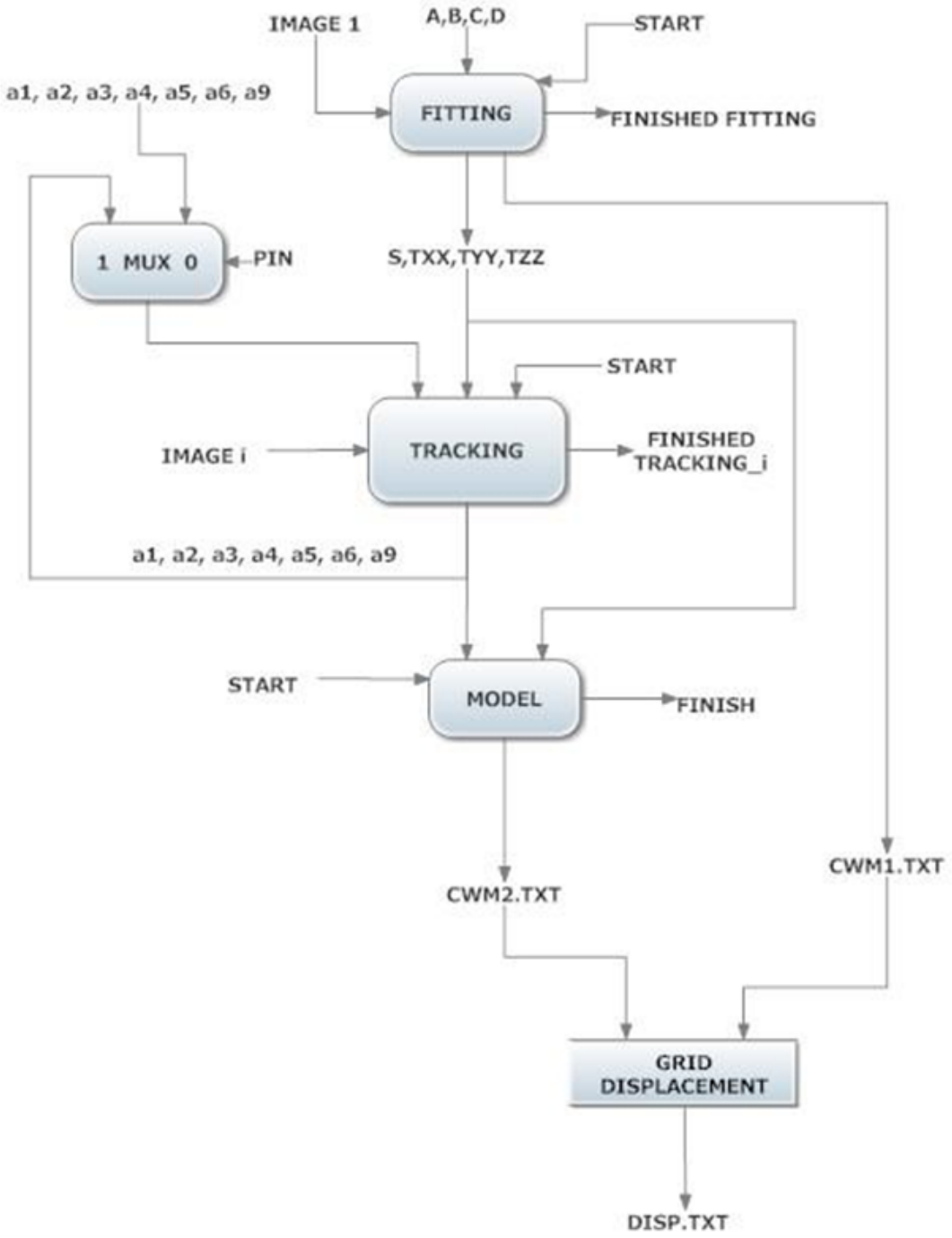

Figure 5.8: Flow diagram of the designed system

The control flow diagram of implemented system is shown in Figure 5.9. It is composed of 5 states which are entered or left upon depending on the occurrence of various conditions. Initially the system is reset and all start signals are set to zero to initialize the system and get rid of any unwanted values that may be associated with different variables or signals. In the next state, fitting unit gives the start signal and it starts computing its operations. As soon as the fitting unit gives the finish signal, control is passed on to the tracking unit in the next state. Whenever,

the finish signal from tracking unit is obtained, a decision is taken based on the count variable. If count < 11, the tracking system is reset in one state and then gives start signal in the next state to track the facial expressions pertaining to the next frame. After the completion of tracking process, grid displacement is taken into account and saved for the next stage to commence.

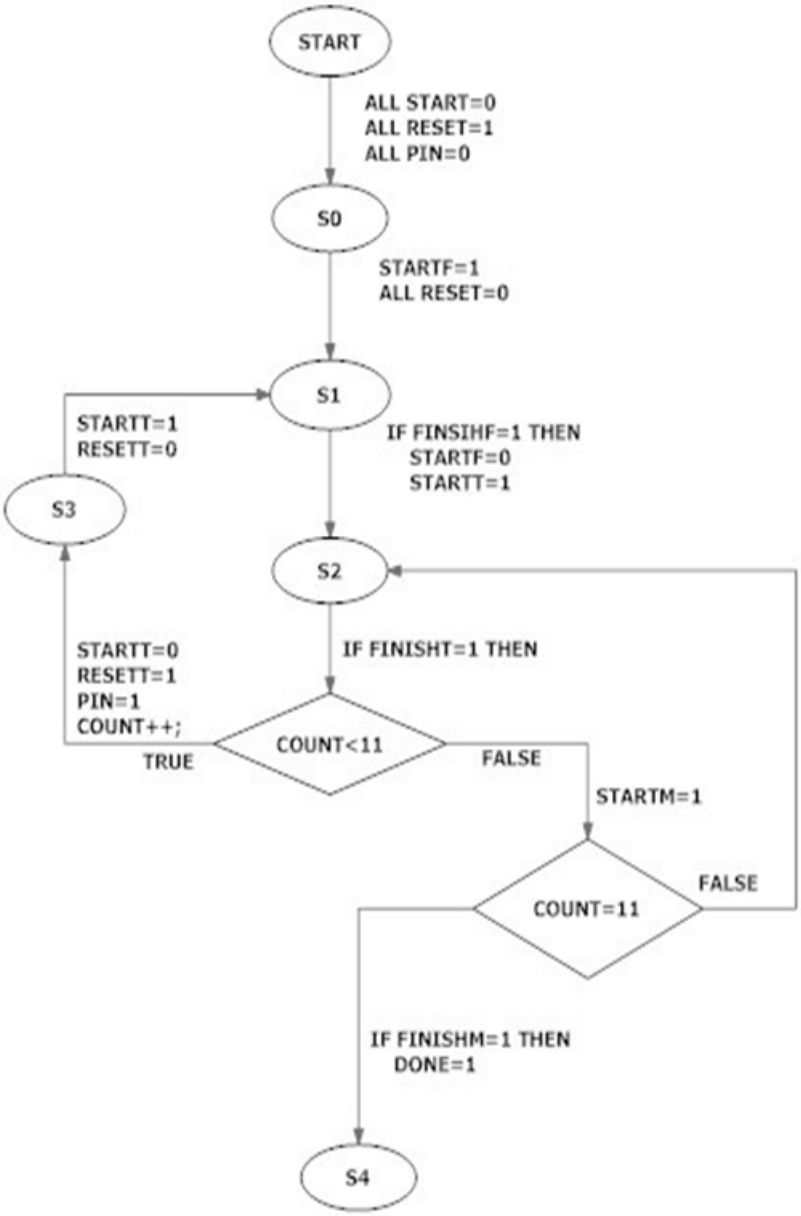

Figure 5.9: Control flow diagram of implemented system

Our algorithm mainly involves large matrix computations. So we make use of systolic array architecture. After making the use of systolic array architecture our compact hardware design flow diagram is given in Figure 5.10.

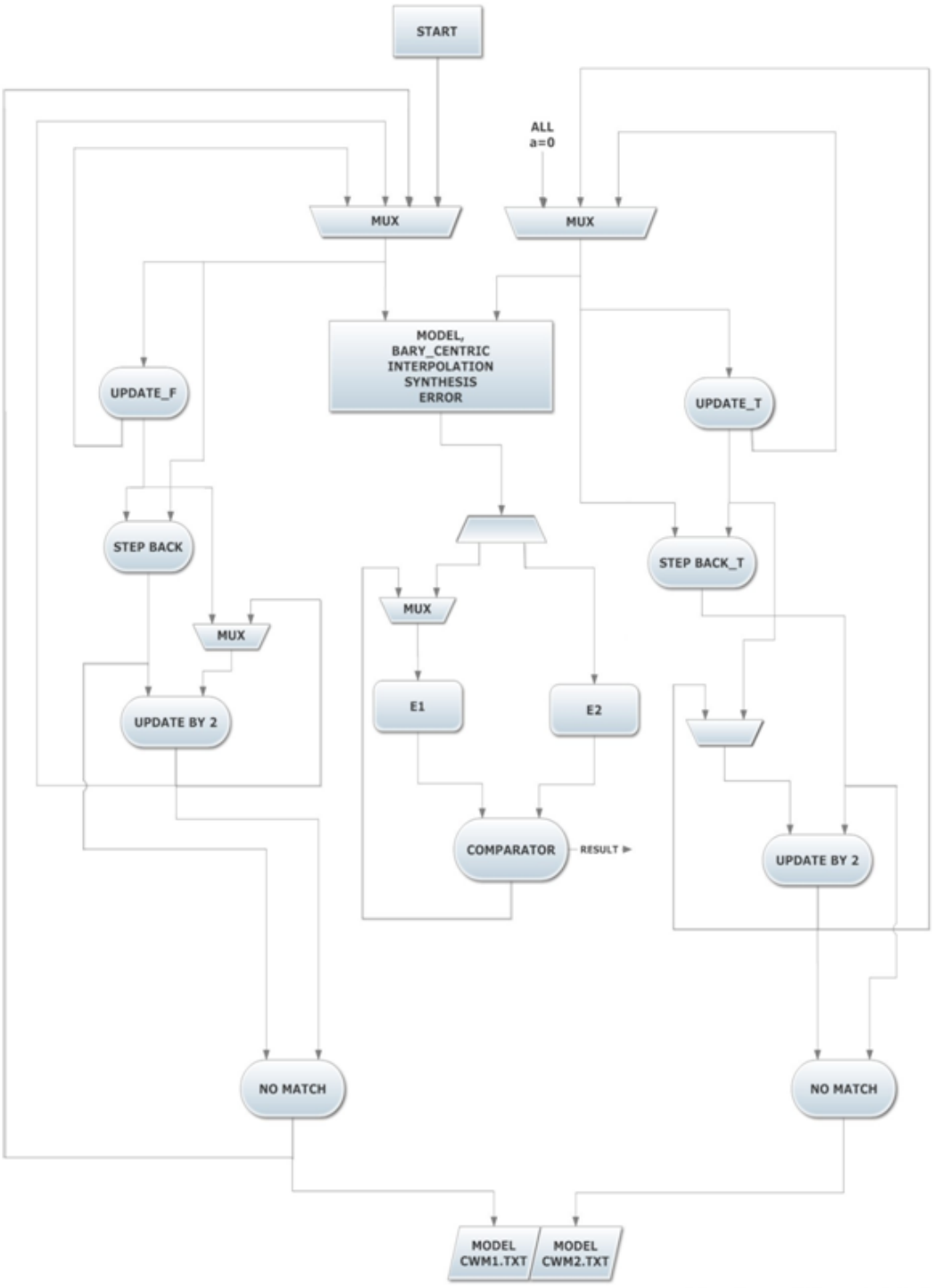

Figure 5.10: Compact hardware design

5.3.2 Experimental results

To implement wire frame model fitting and tracking algorithm, we have written coding in VHDL. VHDL code is simulated in Modelsim. Results of simulation are given in Table 5.1. These results are quite similar to the results obtained with MATLAB. We have tried to synthesize it on Xilinx ISE, but code with

real numbers can not be synthesized on Xilinx, either it should be converted to integers or floating point arithmetic should be used. Our model coordinates, scaling parameters, translation parameters, animation parameters etc. all are real numbers. In our algorithm huge matrix multiplications are involved e.g. matrix of size1320 $\times$ 1320 is multiplied with 1320 $\times$ 10 and so on, there are many. We have converted all real numbers into integers, and simulated it on Modelsim. In this case, we are not getting proper accuracy, because when our AAM model runs, node coordinates of the model are changing by few fractions, which when converted to integers our model fails to fit. Then we have recoded our VHDL program for real numbers using floating point arithmetic, but resources on available FPGA are not sufficient to implement it. So, we have implemented only multiclass SVM part on FPGA. Here, the SVs and their weights, input vector and bias values are rounded off to the nearest digit and then fed to FPGA. Implementation of SVM is explained in next section.

Table 5.1: Accuracy of FER obtained through Modelsim simulation

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	76%	0	10%	0	0	0	0
Surprise	8%	90%	0	0	0	0	4%
Sad	0	0	73%	8%	28%	15%	0
Anger	0	0	17%	84%	0	0	0
Disgust	8%	6%	0	8%	62%	15%	4%
Fear	8%	0	0	0	0	70%	0
Neutral	0	4%	0	0	10%	0	92%

5.4 Implementation of Multiclass SVM

The output of the wire frame model fitting and tracking, is the difference between wire frame node coordinates of first and last frame of image sequence. It is in the form of a vector. We have considered 60 nodes and each node has (x, y) coordinates. So our input vector or test vector $\overline{x}$ is as in (5.4)

$$\overline{x} = [d_1, d_2, \ldots\ldots\ldots\ldots\ldots d_{60}]^T \tag{5.4}$$

Here, d is nodes of the wire frame model, in our case it is 60 and $d_i = (x_i, y_i)$. So, $\overline{x}$ is a vector of 120 dimension. It is classified using One Vs All SVM with linear kernel into one of the seven facial expressions. We have selected one Vs all SVM, because we got maximum accuracy with minimum number of classifiers. See Chapter 5. In this scheme there is one binary SVM for each class to separate members of that class from members of other classes. We have number of classifiers

equal to number of classes. Classifier i, j is trained using all patterns from class i as positive instances, and all patterns from rest of the classes is assumed to be in class j as negative instances. The class for which decision function gives maximum value will be declared as class of new instance. The decision function is given as in (5.5). We compute decision function for all seven classes, the class which gives maximum value is considered as a class of unknown sample.

$$Decision\ Function\ D(\vec{x}) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i K(\overline{x} \bullet \overline{s_i}) + b\right) \tag{5.5}$$

Here $K(\overline{x} \bullet \overline{s_i})$ represents the kernel function. As we got maximum accuracy using linear kernel, for hardware implementation, we use linear kernel. We take dot product of support vector and test vector. Here $\overline{s}_i$ represents the support vector (which is independently a column vector) of dimension 120×no of support vectors of one class and $\overline{x}$ is the test vector which is a row vector of dimension 1×120. α_i is the weight of support vector obtained after training. We have performed training part of the SVM in MATLAB. After training, we got support vectors, their weights, and bias values. Table 5.2. shows the data extracted from the training phase.

Table 5.2: Number of Support Vectors for Each Class

S. No.	Facial Expression	No of Training Samples	No of Support Vectors	Bias Values
1	Happy	25	24	-2.6214
2	Surprise	25	27	-9.3837
3	Sad	21	25	-0.7933
4	Anger	20	26	-3.1861
5	Disgust	23	36	-1.8676
6	Fear	21	28	-0.9999
7	Neutral	25	22	-2.8796

Here, major computation involved is the kernel computation i.e., the dot product of support vector and the test vector. Total number of SVs are 188. Multiplication of a 1×120 vector with 120×188 matrix is very crucial for parallel implementation, because of limited number of resources. So, we divide the problem into seven parts, as our classes are seven. Each part performs multiplication of 1×120 vector with $120 \times$SV of that particular class. Let us consider the Happy class for which number of SVs are 24. So we perform multiplication of 1×120 vector with 120×24 matrix. Size of the matrix is reduced, but number of vector elements is same i.e., 120. To increase the speed of computation, we divide this computation in four parts, and then we run these four parts in parallel. Each part now performs multiplication

of vector1 $\times$ 30 with 30 $\times$ 24 matrix using systolic array architecture as shown in Figure 5.11. To do this we need 24 PEs, where each PE performs multiplication and accumulation. Our output is 24 scalar values. Rest of the three parts have the same architecture. These three parts also produce 24 scalar values. Now we have matrix K_1 of size 4×24 as shown in (5.6). Now, we add all the elements vertically as $(p_i + q_i + r_i + t_i)$ to get final 24 scalar values. As we are using systolic array architecture, we need zero padding in input vector as well as in SV matrix. SV matrix is precomputed and is stored in RAM. Input vector is obtained from our previous section i.e., model fitting and tracking. As we know, dimensions of all vectors and matrices, we can do zero padding easily while storing input vector. Once we get 24 values from kernel computation, we multiply it with product of $\alpha_i y_i$. Here, α_i is weight of support vector and y_i is the class label. There product w_i is precomputed and stored in RAM, where $w_i = \alpha_i \times y_i$. Now, we have two vectors $K_1$of size 1×24 and w_i of size 24×1. Their product gives one scalar value with which we add bias value, which is precomputed and stored in RAM. Thus we got value of Decision function $D(\overline{x})$for class 1 i.e., happy.

$$K_1 = \begin{bmatrix} p_1 & p_2 & -- & -- & p_{24} \\ q_1 & q_2 & -- & -- & q_{24} \\ r_1 & r_2 & -- & -- & r_{24} \\ t_1 & t_2 & -- & -- & t_{24} \end{bmatrix} \tag{5.6}$$

Same sequence of operation is repeated for seven classes. Finally the class which has maximum value is declared as the class of input vector.

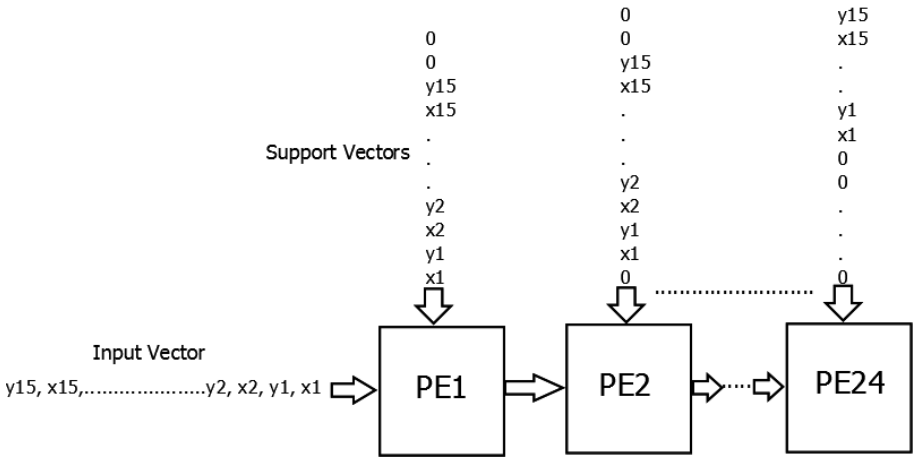

Figure 5.11: Systolic array architecture using 24 PEs

The data flow of this computation is summarized as follows.

- α_i weight of support vectors of each class are multiplied with their class labels y_i and the product w_i is stored in RAM .

- Kernel Function Calculation: In this calculation two matrices (one is a input vector and another is support vector) are multiplied and a row vector is the output
- w_i is then multiplied with the kernel function output, producing a scalar value.
- Now this scalar value is added with precomputed bias value to produce value decision function.
- The class for which decision function $D(\vec{x})$ is maximum, will be the class of input vector.

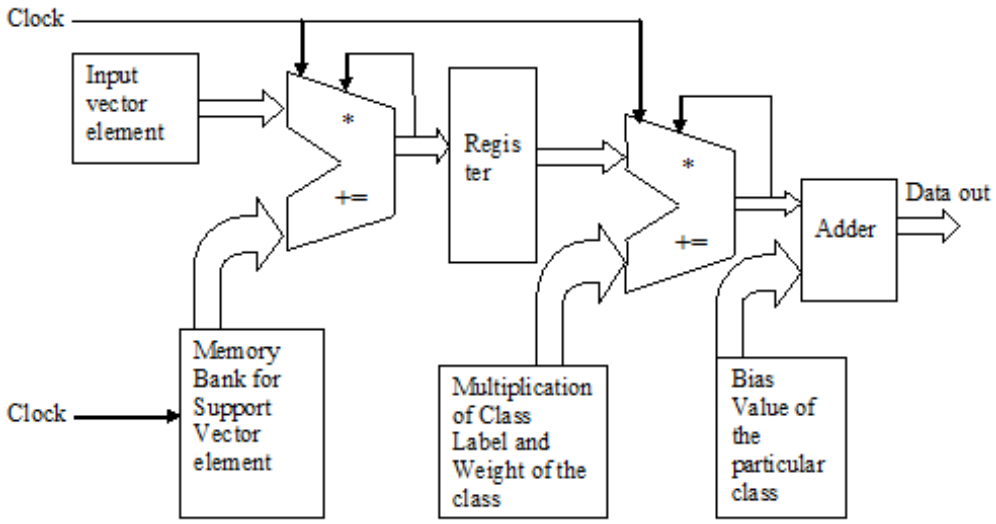

Figure 5.12: Flow diagram of proposed architecture

5.5 Partial reconfiguration approach for low power implementation of MC SVM

Then we perform partial reconfiguration. Partial Reconfiguration is the ability to dynamically modify blocks of logic by downloading partial bit files, while the remaining logic continues to operate without interruption. Partial Reconfiguration enables system flexibility, perform more functions while maintaining communication links. Other advantages are size and cost reduction, time-multiplex the hardware to require a smaller FPGA. It also reduces the power by shutting down power-hungry tasks, when not needed. High performance, special purpose computer systems are typically used to meet specific application requirements or to off-load computations that are especially taxing to general-purpose computers. As

hardware cost and size continue to drop and processing requirements become well-understood in areas such as signal processing and image processing, more special purpose systems are being constructed. Because the knowledge gained from individual experiences is neither accumulated nor properly organized, the same errors are repeated. I/O and computation imbalance is a notable example-often, the fact that I/O interfaces cannot keep up with device speed is discovered only after constructed a high speed, special-purpose device.

5.5.1 Low Power Reconfiguration Strategy

Partial reconfiguration (PR) is the ability for a portion of an FPGA to be reprogrammed while the remainder of the system remains unchanged. A partial bit stream loads only a portion of the design onto the FPGA rather than rewriting the entire design. Partial reconfiguration is especially useful for reprogramming a portion an FPGA during operation without affecting the rest of the system. This practice is called dynamic partial reconfiguration. Static partial reconfiguration refers to reprogramming a portion of the FPGA while the rest of the board is in a reset state **[69]**. Difference-based partial reconfiguration is used to only make small changes in the FPGA design. The generated bit stream only includes differences between designs. Difference-based partial reconfiguration allows for faster reprogramming of the device since only the changes must be rewritten, but it has limited applications as compared to the module-based solution. Systolic array implementation of SVM classifier is reconfigured as difference based approach. In a system implementing module-based partial reconfiguration, modules that are to be kept in continuous operation without the capability of being partially reprogrammed are referred to as static modules. One or more modules can be designated as the partially reconfigurable module(s), which require additional considerations in the design, synthesis, and implementation stages. Specifically, a modular design flow must be used which will synthesize and create separate bit streams for each of the reconfigurable modules as well as a total system bit stream including all of the static logic and one implementation of each of the reconfigurable modules. Partial reconfiguration can be implemented through a JTAG connection to a PC or internally through custom logic or an on-board processor, such as the embedded Power PC in the Virtex II Pro FPGA. Partially reprogramming the FPGA through internal circuitry, referred to as self-reconfiguration, is a much more useful method of partial reconfiguration since it eliminates the need for an external PC. The partial bit streams are stored in memory and are written to the Internal Configuration Access Port (ICAP) of the FPGA in order to reconfigure the specified region of the board with the new logic.

5.6 Implementation and synthesis results of SVM

For FPGA implementation, seven class SVM is coded using VHDL. Simulation is performed on Modelsim. Then the code is synthesized on Xilinx ISE. Synthesis results are given in Table 5.3. Device utilization summary is given in Table 5.4.

Table 5.3: Primitive and black box usage

Items	Used
GND	1
INV	936
LUT1	760
LUT2	15685
LUT3	3750
LUT4	4355
LUT5	2810
LUT6	1475

Items	Used
MUXCY	22265
VCC	1
XORCY	21935
LD_1	7680
LDE_1	7680
BUFG	31
OBUF	193
DSP48E1	135

Table 5.4: Device Utilization Summary

6vlx75tlff484-11	Utilized	Available	Percentage utilization
Number of slice registers	15360	93120	16%
Number of slice LUTs	29771	46560	63%
Number of LUT Flip Flop pairs with unused Flip Flop	22571	37931	59%
Number of LUT Flip Flop pairs with unused LUT	9160	37931	21%
Number of fully used LUT FF pairs	7200	37931	18%
Number of bonded IOBs	193	240	80%
Number of BUFG/BUFCTRLs	31	32	96%
Number of DSP481Es	135	288	46%

For testing our design, we run the program of model fitting and tracking. We extract the geometrical deformation feature vector for a test sample in integer form. This vector is then applied as a input vector to SVM. Accuracy of the classifier using implemented design in given in Table 5.5. Average accuracy obtained is 74.42%. It is less than the accuracy obtained using MATLAB, which is 78%. The reason for this is real numbers are converted into integers.

Table 5.5: Accuracy of classifier using implemented design on FPGA

Expression	*Happy*	*Surprise*	*Sad*	*Anger*	*Disgust*	*Fear*	*Neutral*
Happy	70%	0	15%	0	0	0	0
Surprise	10%	82%	0	0	0	0	2%
Sad	0	0	65%	11%	15%	15%	0
Anger	0	0	13%	79%	0	0	0
Disgust	6%	8%	7%	10%	67%	15%	3%
Fear	4%	6%	0	0	0	68%	5%
Neutral	10%	4%	0	0	18%	2%	90%

5.7 Conclusions

FPGA implementation of the design involving, large number of big matrix multiplications of real number is a challenging problem. Either real numbers should be converted into integers or floating point arithmetic must be used. But, for floating point arithmetic, more number of resources are required on FPGA. For small programs, it will work fine, but for complex one it is difficult. In some applications, conversion of real numbers into integers can given desirable results, but in our case it fails as far as model fitting and tracking part is considered. But, we got considerable accuracy in case of implementation of SVM. For model fitting and tracking part, we have tried using floating point arithmetic, but we got error as it requires large number of resources which are not available on FPGA. To reduce the complexity and matrix size, we have modified our model fitting and tracking code in MATLAB. Our image size is 33×40 so all vectors are of size 1320×1. Eigen matrix is of size 1320×1320. So all computations are running around this matrix. So we reduced our image size to 15×20. But, our image processing algorithm fails on this size of image.

The only possible solution over this could be hardware software codesign approach, where some part should be performed on software and some on FPGA. In our future work, we are trying to design such system.

Chapter 6

Facial expression recognition using sparse representation classifier

The shortcoming of the SVM is that it is often not as compact as the other classifiers such as neural networks. The performance is not ideal when the classes are highly correlated to each other. Sparse representation classifier (SRC), first codes a testing sample as a sparse linear combination of all the training samples, and then classifies the testing sample by evaluating which class leads to the minimum representation error. SRC is much more effective than state-of-art methods in dealing with facial expressions. If an appropriate kernel function is utilized for a test sample, more neighbors probably have the same class label in the high dimensional feature space. Sparse representation in the high dimensional space can improve the performance of recognition and discriminative ability. Using kernel approach we can change the distribution of samples by mapping them into a high dimensional feature space by a nonlinear mapping. The sample can be represented more accurately by sparse representation dictionary, in the high dimensional feature space.

Representing signals as linear combinations of basis vectors sparsely selected from an overcomplete dictionary has proven to be advantageous for many applications in pattern recognition, machine learning, signal processing, and computer vision. In the sparse representation classifier algorithm, it is assumed that the whole set of training samples form a dictionary, and then the recognition problem is cast as one of discriminatively finding a sparse representation of the test image as a linear combination of training images [70].

6.1 Sparse representation based classifier

For the training samples of a single class, this assumption can be expressed as

$$d_{k,test} = \alpha_{k,1}d_{k,1} + \alpha_{k,2}d_{k,2} + \cdots + \alpha_{k,n_k}d_{k,n_k} + \varepsilon_k \tag{6.1}$$

$$d_{k,test} = \sum_{i=1}^{n_k} \alpha_{k,i}d_{k,i} + \varepsilon_k \tag{6.2}$$

$d_{k,test}$ is the test sample (deformed feature vector of test image) of the k^{th} class. $d_{k,i}$ is the i^{th} training sample (deformed feature vector of training image) of k^{th} class.

$\alpha_{k,i}$ is the corresponding weight and $\varepsilon_k$is the approximation error. For the training samples from all c classes the above equation can be expressed as

$$d_{k.test} = \alpha_{1,1}d_{1,1} + \cdot\cdot + \alpha_{k,1}d_{k,1} + \cdot\cdot + \alpha_{k,n_k}d_{k,n_k} + \cdot\cdot \alpha_{c,n_c}d_{c,n_c} + \varepsilon$$

$$d_{k,test} = A\alpha + \varepsilon \tag{6.3}$$

$$A = [d_{1,1} \cdots d_{1,n_1} \cdots d_{k,1} \cdots d_{k,n_k} \cdots d_{c,1} \cdots d_{c,n_c}]$$

$$\alpha = [\alpha_{1,1} \cdots \alpha_{1,n_1} \cdots \alpha_{k,1} \cdots \alpha_{k,n_k} \cdots \alpha_{c,1} \cdots \alpha_{c,n_c}]$$

The linearity assumption in the SRC algorithm coupled with 6.3 implies that the weight vector α should be zero except those associated with the correct class of the test sample. To obtain the weight vector α we have used $l_1$norm minimization problem [71].

min $\|\alpha\|_1$subject to $\|d_{k,test} - A\alpha\|_2 \leq \varepsilon$

This is a convex optimization problem and can be solved by quadratic programming. Once a sparse solution of α is obtained, for each class i the reconstructed sample is calculated as

$$d_{recons}(i) = \sum_{j=1}^{n_i} \alpha_{i,j}d_{ij} \tag{6.4}$$

For each class i compute the residual between test sample and reconstructed sample as

$$r(d_{test}, i) = \|d_{k,tets} - d_{recons}(i)\|_2 \tag{6.5}$$

The class of the given test sample is determined as

$$\text{class of test sample} = \text{argmin } i \; r(d_{test,}i) \tag{6.6}$$

6.2 Kernel SRC

We know that kernel approach can change the distribution of samples by mapping the samples into a high dimensional feature space by a nonlinear mapping. In the high dimensional feature space, the sample can be represented more accurately by sparse representation dictionary.

We have c classes and the set of training samples is $A = [A_1, A_2, ...A_c] = [d_{1,1}, d_{1,2}, ...d_{c,n_c}]$ where n is total number of training samples. A is a matrix of

training samples in original space. Let us say y is a test sample, the samples are mapped from original feature space into a high dimensional feature space $y \rightarrow \phi(y)$; $A = [d_{1,1}, d_{1,2}, ...d_{c,n_c}] \rightarrow U = [\phi(d_{1,1}), \phi(d_{1,2}).....\phi(d_{c,n_c})]$ by a nonlinear mapping. Here U is a matrix of training samples in high dimensional space and ϕ is mapping function from original dimensional space to high dimensional space [71]. It can be formulated as

$$\phi(y) = Uv \tag{6.7}$$

Where $\phi(y)$ is test sample in high dimensional feature space and v is vector of sparse coefficients in high dimensional space. We use $l_1$norm minimization problem. min $\|v\|_1$subject to $\|Uv - \phi(y)\|_2 \leq \varepsilon$
We can write

$$\left\|U^T Uv - U^T \phi(y)\right\|_2 \leq \delta \tag{6.8}$$

where $\delta = U^T \varepsilon$ and $U^T U = [\phi(d_{1,1}), \phi(d_{1,2}), ...\phi(d_{c,n_c})]^T [\phi(d_{1,1}), \phi(d_{1,2}), ...\phi(d_{c,n_c})]$

$$U^T U = \begin{bmatrix} k(d_{1,1}d_{1,1}) & k(d_{1,1}d_{1,2}) & \cdots & k(d_{1,1}d_{c,n_c}) \\ k(d_{1,2}d_{1,1)}) & k(d_{1,2}d_{1,2}) & \cdots . & k(d_{1,2}d_{c,n_c}) \\ \vdots & \vdots & \vdots & \vdots \\ k(d_{c,n_c}d_{1,1}) & k(d_{c,n_c}d_{1,2}) & \cdots & k(d_{c,n_c}d_{c,n_c}) \end{bmatrix} \tag{6.9}$$

$$U^T \phi(y) = [\phi(d_{1,1}), \phi(d_{1,2}).....\phi(d_{c,n_c})]^T \phi(y)$$

$$U^T \phi(y) = \begin{bmatrix} k(d_{1,1}y) \\ k(d_{1,2}y) \\ \vdots \\ k(d_{c,n_c}y) \end{bmatrix} \tag{6.10}$$

Then, we have computed $\widetilde{v}$ as follows

$$\widetilde{v} = argmin \|v\|_1 \left\|U^T Uv - U^T \phi(y)\right\|_2 \leq \delta \tag{6.11}$$

Then, we compute residual

$$r_i(y) = \left\|U^T \phi(y) - U^T U\widetilde{v}\right\|_2 \tag{6.12}$$

Class of y is now computed as

$$(y) = argmin(r_i(y)) \tag{6.13}$$

6.3 SRC using multiple kernels

It is observed that with different kernels, we get different accuracies. One kernel may not be sufficient to classify the samples. So, we combine several kernels and design a sparse representation classifier with multiple kernels. It is a way of optimizing kernel weights while training dictionary [72]. The mode of multiple kernel is

$$k(d_i, d_j) = \sum_{k=1}^{m} \alpha_k k_k(d_i, d_j) \tag{6.14}$$

where m is the number of kernel function. α_k is kernel weights. k is a kernel function. We restrain the weights of kernel by

$$\sum_{i=1}^{m} \alpha_k^2 = 1 \tag{6.15}$$

Here $\alpha_k \geq 0$. Now our objective function becomes

$$min \left\| \phi(y) - Uv \right\|_2^2 \tag{6.16}$$

Where $\|v\|_1 < \sigma$. For sample d and y, we write

$$\phi(d_i)^T \phi(y_j) = k(d_i y_j) \tag{6.17}$$

Because $\phi(y)$ and U are unknown, (6.16) can not be solved directly, but it can be transformed to

$$\tilde{v} = argmin \left\{ \left\| U^T \phi(y) - U^T U v \right\|_2^2 + \lambda \left\| v \right\|_1 \right\} \tag{6.18}$$

where λ is Lagrange's multiplier and

$$U^T \phi(y) = \begin{bmatrix} \sum_{k=1}^{m} \alpha_k k_k(d_{1,1}, y) \\ \sum_{k=1}^{m} \alpha_k k_k(d_{1,2}, y) \\ \vdots \\ \sum_{k=1}^{m} \alpha_k k_k(d_{c,n_c}, y) \end{bmatrix} \tag{6.19}$$

$$U^T U = \begin{bmatrix} \sum_{k=1}^{m} \alpha_k k_k(d_{1,1}, d_{1,1}) & \sum_{k=1}^{m} \alpha_k k_k(d_{1,1}, d_{1,2}) & \cdots & \sum_{k=1}^{m} \alpha_k k_k(d_{1,1}, d_{c,n_c}) \\ \sum_{k=1}^{m} \alpha_k k_k(d_{1,2}, d_{1,1}) & \sum_{k=1}^{m} \alpha_k k_k(d_{1,2}, d_{1,2}) & \cdots & \sum_{k=1}^{m} \alpha_k k_k(d_{1,2}, d_{c,n_c}) \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{k=1}^{m} \alpha_k k_k(d_{c,n_c}, d_{1,1}) & \sum_{k=1}^{m} \alpha_k k_k(d_{c,n_c}, d_{1,2}) & \cdots & \sum_{k=1}^{m} \alpha_k k_k(d_{c,n_c}, d_{c,n_c}) \end{bmatrix} \tag{6.20}$$

The implementation of SRC with multiple kernels is an iterative process, as the

initial weights are not exact weights, they are estimator. Estimator is not optimal. When the difference of weights α_i is small enough, the iteration process is stopped. Recognition rate remains stable after several iterations. After the convergence, we compute residual as

$$r_j(y) = \left\| U^T \phi(y) - U^T U \widetilde{v} \right\|_2^2 \tag{6.21}$$

Class of sample y will be computed as

$$(y) = argmin \;\; r_j(y) \tag{6.22}$$

6.4 Experimental results

We have used Cohn Kanade database and IMM database for training as well as for testing of seven facial expressions. For nine facial expressions of Bharatnatyam, a classical dance style of south India, we have created our own database with the help of Naatyashastra Institute of fine arts, Navi Mumbai. Out of nine expressions, seven expressions are similar to seven basic facial expressions. These nine expressions are Shanta (Neutral), Adbhuta (Surprise), Hasya (Smile), Bhayanak (Fear), Roudra (Anger), Karuna (Sad), Bibhatsa (Disgust), Shringar (Erotic or love), and Virya (Heroic). It is shown in Figure 6.1 Recognition accuracy of all classifiers for our database is given in Table 6.1. Accuracy using Sparse representation classifier is maximum as compared to ANN, Bayesian and SVM classifier. Four sets containing 25% of the data for each class, chosen randomly, were created. One set containing 25% of the samples for each class is used for the test set, while the remaining sets form the training set. After the classification procedure is performed, the samples forming the testing set are incorporated into the current training set, and a new set of samples (25% of the samples for each class) is extracted to form the new test set. The remaining samples create the new training set. This procedure is repeated four times. The average classification accuracy is the mean value of the percentages of the correctly classified facial expressions. Detailed results of only nine facial expressions are given here. Confusion matrices of nine facial expressions and accuracy with ANN classifier is shown in Table 6.2. We got 74% accuracy with ANN. Confusion matrices of nine facial expressions and accuracy with Bayesian classifier is shown in Table 6.3. We got 75.77% accuracy with Bayesian classifier. Confusion matrices of nine facial expressions and accuracy with one Vs all SVM classifier is shown in Table 6.4. We got 80% accuracy with SVM. Confusion matrices of nine facial expressions and accuracy with SRC classifier using linear kernel is shown in Table 6.5. We got 84% accuracy

with SRC using linear kernel. Confusion matrices of nine facial expressions and accuracy with SRC classifier using polynomial kernel is shown in Table 6.6. We got 87% accuracy with SRC using polynomial kernel. Confusion matrices of nine facial expressions and accuracy with SRC classifier using RBF kernel is shown in Table 6.7. We got 88% accuracy with SRC using RBF kernel. Confusion matrices of nine facial expressions and accuracy with SRC classifier using multiple kernel is shown in Table 6.8. We got 95.7% accuracy with SRC using multiple kernels. As a multiple kernel we have used linear, polynomial and RBF kernels. From results, we can see that SRC with multiple kernels outperforms others classifiers. Comparison of all classifiers with SRC is shown in Figure 6.3. Comparison of SRC with different kernel and multiple kernel is shown in Figure 6.4.

Figure 6.1: Nine facial expressions

Figure 6.2: Few more images from our database

Table 6.1: Comparison of recognition results for different classifiers on our database

Classifier	Accuracy with our own database for 9 facial expressions
ANN	74%
Bayesian	75.77%
SVM	80.11%
SRC with linear kernel	84.77%
SRC with polynomial kernel	87.44%
SRC with RBF kernel	88.22%
SRC with multiple kernel	95.77%

Table 6.2: Confusion Matrices and accuracy for nine facial expressions using ANN

Expressions	Shanta (%)	Adbhuta (%)	Hasya (%)	Bhayanak (%)	Roudra (%)	Karuna (%)	Bibhatsa (%)	Shringar (%)	Virya (%)
Shanta	84	0	0	0	0	10	0	10	5
Adbhuta	0	80	9	8	10	0	10	0	8
Hasya	0	10	76	8	0	0	10	9	4
Bhayanak	0	10	5	78	10	0	10	0	8
Roudra	2	0	0	0	70	5	0	0	4
Karuna	8	0	0	0	0	69	0	10	0
Bibhatsa	0	0	5	4	5	0	70	0	0
Shringar	4	0	5	2	0	6	0	68	0
Virya	2	0	0	0	5	10	0	3	71

Table 6.3: Confusion Matrices and accuracy for nine facial expressions using Bayesian classifier

Expressions	Shanta (%)	Adbhuta (%)	Hasya (%)	Bhayanak (%)	Roudra (%)	Karuna (%)	Bibhatsa (%)	Shringar (%)	Virya (%)
Shanta	88	0	0	0	2	8	2	10	3
Adbhuta	0	80	8	10	3	0	10	0	0
Hasya	0	6	78	6	0	0	0	10	0
Bhayanak	0	6	8	77	0	5	0	0	4
Roudra	2	0	0	0	75	0	10	2	10
Karuna	4	2	0	0	5	72	2	2	10
Bibhatsa	0	4	6	6	5	2	71	0	0
Shringar	2	0	0	0	0	5	0	72	4
Virya	4	2	0	1	10	8	5	4	69

Table 6.4: Confusion Matrices and accuracy for nine facial expressions using one Vs all SVM classifier

Expressions	Shanta (%)	Adbhuta (%)	Hasya (%)	Bhayanak (%)	Roudra (%)	Karuna (%)	Bibhatsa (%)	Shringar (%)	Virya (%)
Shanta	87	0	0	2	2	13	0	5	5
Adbhuta	0	84	5	5	0	0	5	0	0
Hasya	0	6	85	5	0	0	5	15	0
Bhayanak	0	5	5	81	10	0	5	0	5
Roudra	0	0	0	0	78	0	5	0	10
Karuna	3	0	0	0	0	77	0	0	0
Bibhatsa	0	5	2	5	0	0	80	0	0
Shringar	5	0	0	0	0	5	0	75	6
Virya	5	0	3	2	10	5	0	5	74

Table 6.5: Confusion Matrices and accuracy for nine facial expressions using SRC classifier with linear kernel

Expressions	Shanta (%)	Adbhuta (%)	Hasya (%)	Bhayanak (%)	Roudra (%)	Karuna (%)	Bibhatsa (%)	Shringar (%)	Virya (%)
Shanta	92	0	0	0	5	4	2	4	0
Adbhuta	0	91	6	5	0	4	5	0	4
Hasya	0	5	89	5	0	0	8	8	4
Bhayanak	0	4	0	85	0	0	0	2	8
Roudra	0	0	0	0	80	3	3	2	2
Karuna	4	0	0	1	4	81	0	2	0
Bibhatsa	0	0	0	0	4	0	82	0	0
Shringar	4	0	5	3	0	4	0	81	0
Virya	0	0	0	1	7	4	0	1	82

Table 6.6: Confusion Matrices and accuracy for nine facial expressions using SRC classifier with polynomial kernel

Expressions	Shanta (%)	Adbhuta (%)	Hasya (%)	Bhayanak (%)	Roudra (%)	Karuna (%)	Bibhatsa (%)	Shringar (%)	Virya (%)
Shanta	93	0	0	0	5	5	2	4	0
Adbhuta	0	89	6	5	0	0	5	0	4
Hasya	0	7	92	5	0	0	6	5	4
Bhayanak	0	4	0	90	0	0	0	0	8
Roudra	0	0	0	0	85	3	3	5	2
Karuna	3	0	0	0	4	87	0	0	0
Bibhatsa	0	0	0	0	0	0	84	0	0
Shringar	4	0	2	0	0	0	0	85	0
Virya	0	0	0	0	6	5	0	1	82

Table 6.7: Confusion Matrices and accuracy for nine facial expressions using SRC with RBF kernel

Expressions	Shanta (%)	Adbhuta (%)	Hasya (%)	Bhayanak (%)	Roudra (%)	Karuna (%)	Bibhatsa (%)	Shringar (%)	Virya (%)
Shanta	93	0	0	0	0	3	0	3	2
Adbhuta	0	91	7	9	2	0	0	0	0
Hasya	0	0	87	0	5	3	4	10	0
Bhayanak	0	6	3	89	3	0	6	0	3
Roudra	0	0	0	2	88	0	0	0	9
Karuna	0	0	0	0	0	87	0	0	4
Bibhatsa	0	0	0	0	0	0	90	0	0
Shringar	4	0	3	0	0	0	0	87	0
Virya	3	3	0	0	2	7	0	0	82

Table 6.8: Confusion Matrices and accuracy for nine facial expressions using SRC with multiple kernel

Expressions	Shanta (%)	Adbhuta (%)	Hasya (%)	Bhayanak (%)	Roudra (%)	Karuna (%)	Bibhatsa (%)	Shringar (%)	Virya (%)
Shanta	98	0	0	0	0	3	0	0	0
Adbhuta	0	97	0	6	2	0	0	0	0
Hasya	0	0	97	0	0	3	2	4	0
Bhayanak	0	3	0	94	3	0	2	0	3
Roudra	0	0	0	0	95	0	0	0	0
Karuna	0	0	0	0	0	94	0	0	2
Bibhatsa	0	0	0	0	0	0	96	0	0
Shringar	2	0	3	0	0	0	0	96	0
Virya	0	0	0	0	0	0	0	0	95

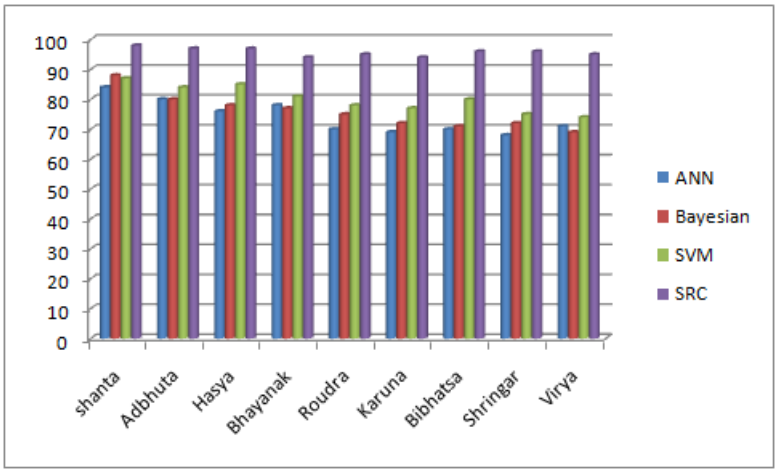

Figure 6.3: Comparison of all classifiers

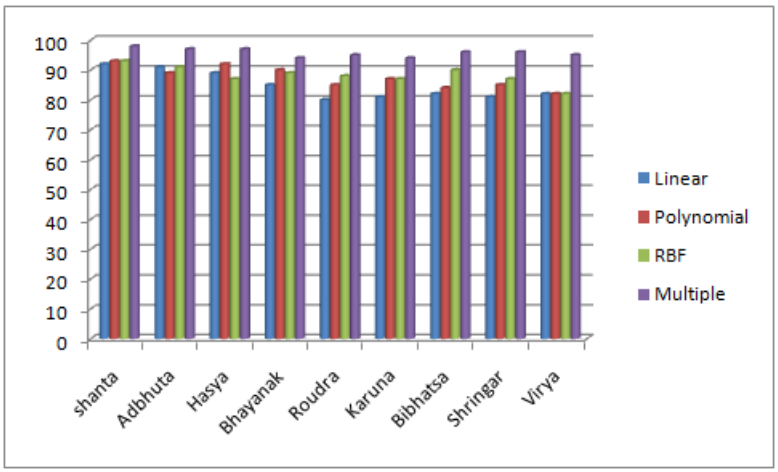

Figure 6.4: Comparison of SRC classifier different kernels

Chapter 7

Development of mathematical model for facial expression recognition

Topographical maps are being recognized as one of the major computational structures underlying neural computations in the brain. They provide dimension reducing feature spaces that seem to be established and maintained under the participation of self organizing adaptive processes. The structure of these maps can be replicated by simple adaptive processes and can be realized by the use of mathematical models. We propose a novel way of recognizing facial emotion expressions by using mathematical modeling [73] of the brain functioning i.e. finding neuronal structures that takes place in the brain while it learns to recognize various facial expressions.

Faces are considered dynamic objects that can undergo a vast number of non rigid transformations. Facial images contain features that can be used to understand, different expressions and facial states. For instance, the shape of the mouth can be used to detect the emotive state of a subject, such as happiness and anger. In order to identify and analyze facial expressions, a well defined description of the deformations is required. It is relatively straightforward to identify a description of rigid transformations such as rotation and translation. However, a simple, natural description of non rigid deformations need be defined. One method for creating a framework to study facial deformations is to construct a model of the facial muscles and skin tissue based on anatomical descriptions. This approach is limited by the quality of existing biological models and its complexity. Another approach models the parts of the face, that are relevant for understanding motion. Only the eyes and mouth are studied, since these are considered to be the most important features possessing expressive power. On static images, expressions have been represented as the deviation of spatial points, i.e. about 100 special points and deviation from their neutral position. In this Chapter, we propose a

framework that allows the description of facial expressions and movement based on image features.

7.1 Face detection

Face detection is the first step of facial expression recognition system. It is a method that determines the locations and sizes of human faces in arbitrary images, and detects facial features and ignores anything else, such as buildings, trees and bodies. It can be regarded as a specific case of object class detection. In object class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. To design fully automatic system, the primary requirement is to automatically detect the face in the scene. The high variability of faces encountered in nature, the pose and illumination variations, the gender, ethnicity and age problems combine to make this task tougher and more complicated than it seems. The Viola Jones object detection framework [57] was probably the first object detection framework to provide competitive object detection rates in real time. Although, it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. The object detection procedure classifies images based on the value of simple features. The most common reason for using features rather than pixels is that features can act to encode ad-hock domain knowledge that is difficult to learn using a finite quantity of training data. The second critical reason for using features is that the feature based system operates much faster than a pixel based system.

7.1.1 Viola Jones algorithm

Paul Viola and Michael Jones [57] proposed an efficient image representation called the integral image that allows fast image computations and manipulations. They have combined this representation with an extension of the AdaBoost machine learning algorithm to design a face detector. The face detector scans patches of the input image for the potential presence of a face. The detector extracts features relevant for the detection of faces. Each feature detects a certain visual pattern, such as a vertical contour, and translates the pattern into a number indicating the degree of presence of the pattern. The translation is based on summations of the individual pixel values. Within a small patch, typically a 24×24 pixel region, features are detected at all positions and sizes, which can lead to a very large number of features and associated numbers. They have used AdaBoost method to discard the vast majority of features and to retain only those that contribute to face detection. They create a strong classifier from a series of weak classifiers.

Each of the features can be regarded as a weak classifier, i.e., each weak classier is a straightforward threshold function that returns a 1 if the feature value lies above the threshold, and a 0 otherwise. The threshold value is the only parameter of the weak classifier. The classification performance of one weak-learner is presumably low. However, given a training set i.e. image regions labeled with a '1' if containing a face and with a '0' otherwise, AdaBoost incrementally selects the best weak learner and the appropriate threshold value and then constructs a summed (strong) classifier until a certain level of performance is reached. They further improved the efficiency of the face detector by creating a cascade of boosted classifiers. Patches are processed in a stage wise fashion through the cascade. If at any stage of the cascade a patch is rejected, it is classified as containing a non face. Only those patches that reach the final stage are classified as containing a face. The combination of the integral image representation and the cascaded AdaBoost algorithm has been shown to be a highly effective face detector that is used in virtually all real time face detection software. Face detection result using Viola Jones algorithm is shown in Figure7.1.

Figure 7.1: Face detection

7.2 Pre processing

Image Pre processing is similar to signal conditioning. We preprocessed the image for noise removal, and normalization against the variation of pixel position or brightness, together with segmentation, location, or tracking of the face or its parts. Expression representation can be sensitive to translation, scaling, and rotation of the head in an image. To combine the effect of these unwanted transformations, the facial image is geometrically standardized prior to classification. Pre processed image is shown in Figure 7.2

Figure 7.2: Pre Processed Image

7.3 Feature extraction

Feature extraction generally reduces the dimensionality of the input space. The reduction procedure retains essential information possessing high discrimination power and high stability. Here, we have used Gabor filters for the facial feature extraction.

7.3.1 Gabor Filters

A Gabor filter, named after Dennis Gabor, is a linear filter used for edge detection [74]. Frequency and orientation representations of Gabor filters are similar to those of the human visual system. They have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. The Gabor filters are self similar. All filters can be generated from one mother wavelet by dilation and rotation. Its impulse response is defined by a harmonic function multiplied by a Gaussian function. Because of the multiplication convolution property, the Fourier transform of a Gabor filter's impulse response is the convolution of the Fourier transform of the harmonic function and the Fourier transform of the Gaussian function. The filter has a real and an imaginary component representing orthogonal directions. Gabor filters are directly related to Gabor wavelets, since they can be designed for a number of dilation's and rotations. However, in general, expansion is not applied for Gabor wavelets, since this requires computation of bi orthogonal wavelets, which may be very time consuming. Therefore, usually, a filter bank consisting of Gabor filters with various scales and rotations is created. The filters are convolved with the signal, resulting in a so called Gabor space [74]. The Gabor space is very useful in image processing applications such as optical character recognition, iris recognition and fingerprint recognition. Relations between activations for a specific spatial location are very distinctive between objects in an image. Furthermore, important activations can

be extracted from the Gabor space in order to create a sparse object representation. This process is closely related to processes in the primary visual cortex. Jones and Palmer showed that the real part of the complex Gabor function is a good fit to the receptive field weight functions found in simple cells in a cat's striate cortex.

We have used a set of Gabor filters with different frequencies and orientations for extracting useful features from an image. The features extracted are given as an input to the Mathematical model and the outputs are generated. Figure 7.3 shows different features extracted when passed through Gabor filters and Figure 7.4 shows amalgamations of different features extracted.

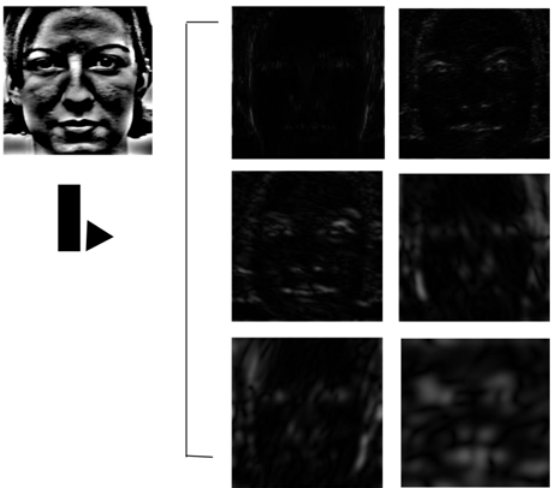

Figure 7.3: Different features extracted when passed through Gabor filters

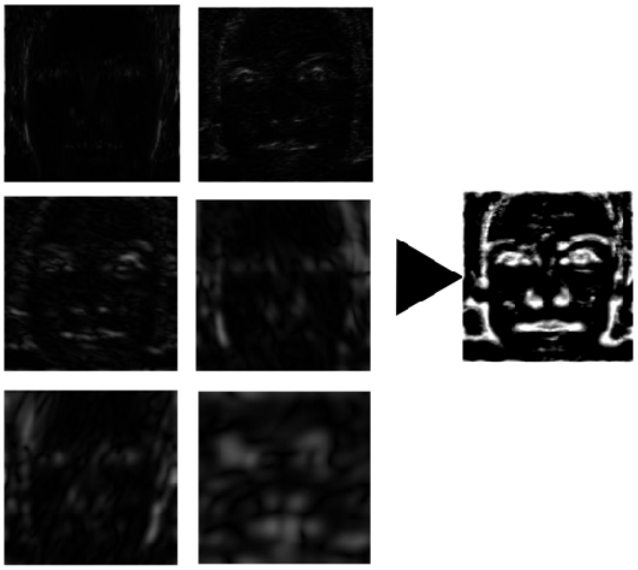

Figure 7.4: Amalgamation of Features extracted

7.4 Generation of inputs to mathematical model

As we proceed to the next section explaining the mathematical model, it will be understood that there is a need of generation of four dimensional data sets for the images based on the position and orientation or the direction of the gradients. To generate such a data set, following procedure is adopted. Gradient images are created from the original image. Gradient of an image is the measure of change in the intensity of image in the direction of X and Y. We use this property to determine the magnitude and orientation of the vector using the following property.

$$\|\nabla F\| = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2} \tag{7.1}$$

$$\theta = tan^{-1}\left[\frac{\frac{\partial F}{\partial y}}{\frac{\partial F}{\partial x}}\right] \tag{7.2}$$

Magnitude of image gradient is given as in (7.1), while (7.2) gives orientation angle. Each pixel of a gradient image measures the change in intensity of that same point in the original image, in a given direction. To get the full range of direction, gradient images in the x and y directions are computed. Figure 7.5 shows gradient magnitude and orientation extracted.

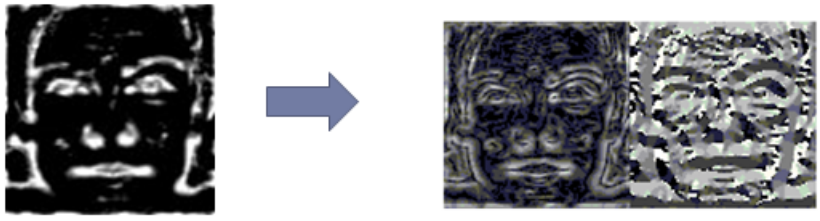

Figure 7.5: Image showing gradient magnitude and orientation extracted

7.5 Neural Algorithm

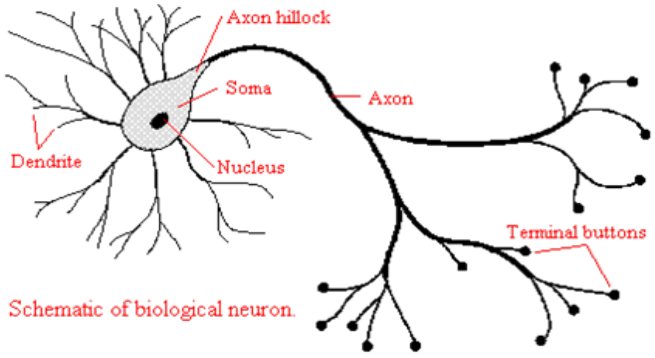

Figure 7.6: Neural architecture in biological systems

A typical schematic of biological neuron is shown in Figure 7.6. When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficacy, as one of the cells firing cell B is increased. Brain is a self organizing system that can learn by itself by changing(adding, removing, strengthening) the interconnections between neurons. Any system that takes a form that is not imposed from outside (by walls, machines or forces) can be said to self organized. The result of brain's self organization is the formation of feature maps in the brain that have a linear or planar topology (that is, they extend in one or two dimensions). Examples are (i) nitpick map in which sound frequencies are spatially mapped into regions of the cortex in an orderly progression from low to high frequencies. (ii) Retinitis map in which visual field is mapped in the visual cortex (occipital lobe) with higher resolution for the center of the visual field.

The mathematical model used to classify the facial expression is modelled based on the striate cortex of higher intelligence animals. The striate cortex of higher animals contains a topographical representation of visual space. Neighborhood preserving maps of several variables describing the features such as position in the visual space, line orientation, movement direction, and ocularity are embedded in the representation. The representation of the multidimensional feature space onto the two dimensional cortical sheet is achieved in a hierarchical fashion. The topographic projection of the retina establishes a primary order, and for each small region of the visual field there are patches or stripes of cells with similar feature preference. Here, we have used a mathematical model based on the principle of continuous mapping to replicate the visual space. We have used it to determine the facial expressions. In the visual cortex, retinoptic location defines the primary feature that is mapped smoothly across the visual cortex. The spatial variation

of additional secondary feature, such as orientation and occularity is smooth only in local domains, which are separated by boundaries where rapid changes occur.

We have used a self organizing feature map (SOFM) algorithm for the formation of a topographical representation of a set of patterns given as vectors in some input or feature space [73]. We have extracted features from a face image using a Gabor filter. From these extracted features, we compute image gradient, and from image gradient, we construct a feature vector. This feature vector is mapped on a neuron structure using SOFM.

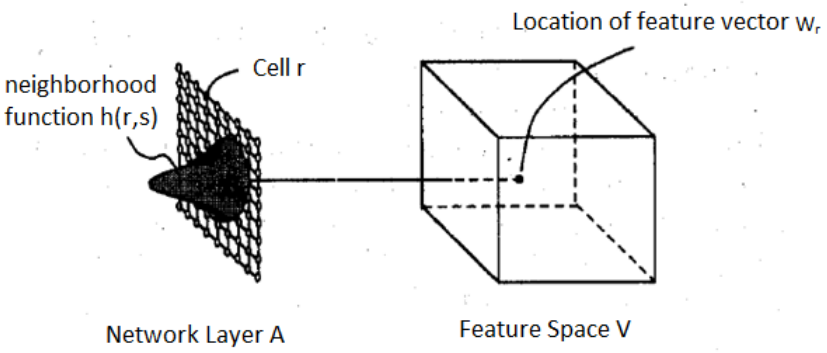

Figure 7.7: The Low dimensional network model [3]

Schematic drawing of the model is shown in Figure 7.7. The cells $r \epsilon A$ are arranged onto a two dimensional network layer to match the topology of cortical layer containing the feature map. The cell is not identified with the single neuron, but rather with a group of neurons or with some small patches of tissue where neurons, with common response properties are located. In the visual system, receptive fields are defined by the connection patterns that a higher level cell receives from the level immediately below. The retinal RFs are circular as shown below in 7.8. Cortical receptive fields are usually elongated in nature with subfields as shown in Figure 7.9. A cortical cell receives excitatory connections from green cells and inhibitory connections from the red cells. More deeper the color, higher is the excitation or inhibition.

Figure 7.8: Retinal Receptive Field

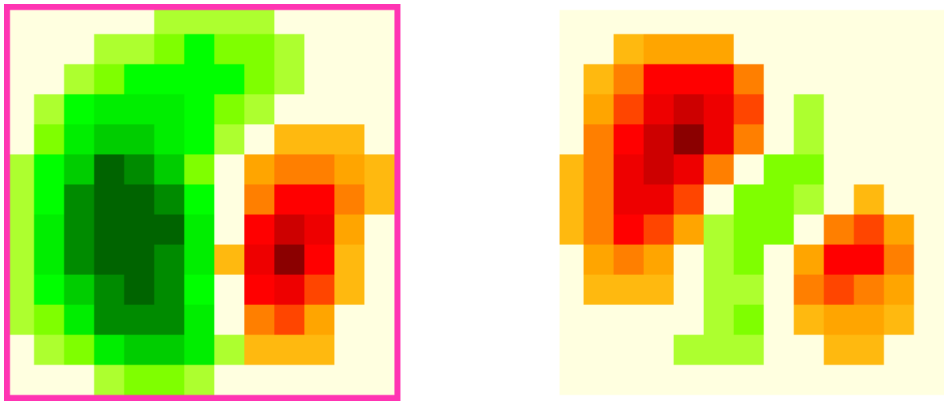

Figure 7.9: Cortical receptive field with subfield

To describe the receptive field properties position (x_r, y_r) of the receptive field centers in the visual space, preferred orientation ϕ_r and orientation specificity q_r for a cell r we use a four dimensional feature vector [3]

$$\overrightarrow{W_r} = (x_r, y_r, q_r cos(2\phi_r), q_r sin(2\phi_r)) \tag{7.3}$$

Orientation specificity is the tuning value of the receptive field. If the response of a cell to external stimuli is Gaussian in nature, then orientation specificity can be defined as half of the standard deviation.The dependence of these feature vectors on the cell locations r describes the spatial distribution of the selectivity of cells over the cortical layer. The position of the receptive field is given by the coordinates (x_r, y_r) of its centroid. Preferred orientation is given by the orientation ϕ_r of the receptive field major axis. Orientation specificity is given by its elongation. The net neural receptivity is determined by the preferred orientation and the specificity coordinates. Higher the specificity of r towards the direction ϕ_r, higher it is receptive to the input vector which is in the direction of ϕ_r. The input to the network layer consists of localized oriented stimuli. They are described by a feature vector, which is the same type as of W_r and is given in (7.4)

$$\overrightarrow{v} = (x, y, qcos(2\phi), qsin(2\phi)) \tag{7.4}$$

7.6 SOFM Algorithm

Self organizing feature maps (SOFM) are based on unsupervised learning. Supervised learning discover patterns in the data that relate data attributes with a target (class) attribute. These patterns are then utilized to predict the values of the target attribute in future data instances. While during unsupervised learning, the data have no target attribute. We want to explore the data to find some intrinsic structures in them. Different steps in the algorithm are given below.

- Structure of neurons is arranged in a two dimensional grid as shown in Figure 7.10. It is a structure of 10×10 neurons. In our case it is a structure of 50×50 neurons. Input vector is $[v_1, v_2, ...v_n]$.

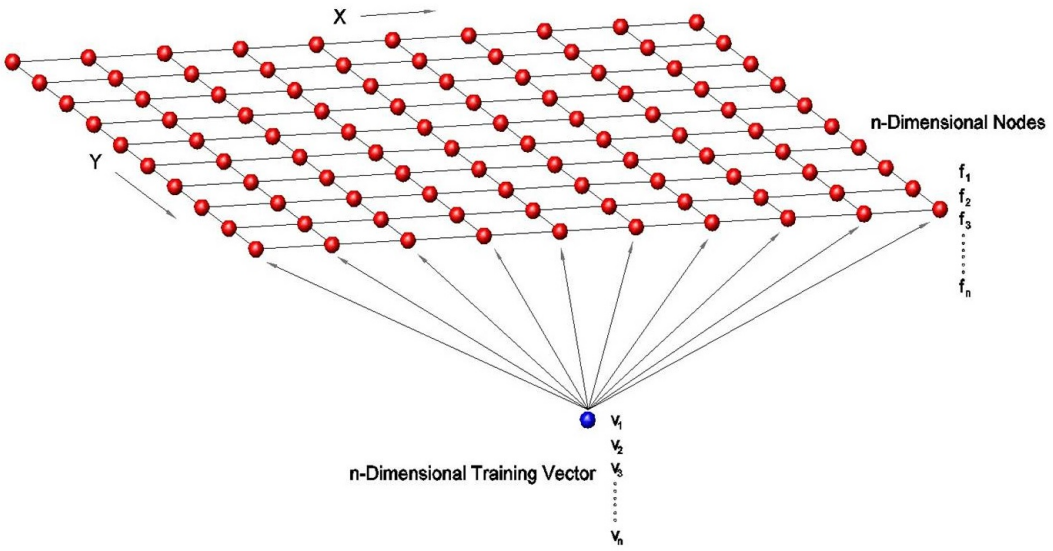

Figure 7.10: Neuron structure

- Weight is assigned to each neuron, so each neuron contains a weight vector $[f_1, f_2.....f_n]$.

- Initialize the weights to the neuron randomly or pregenerated.

- Iterate through inputs. Our neuron structure is 50×50 neurons. While our image size obtained after passing through the Gabor filter is 100×100. Thus, we have 10,000 feature vectors. These vectors are mapped on 50×50 neuron structure, so, our input is 10000. Number of iterations is 10000.

- Present training data to the map and let the cells on the map compete to win in some way. Usually the cell with closest distance to the presented training vector is called the "winner". Euclidean distance is usually used.

- Adjust weights of "winning" neuron and its neighbors by using Gaussian or Mexican hat. Update process is shown in Figure 7.11. While mapping input feature vector on neuron structure, we first find the neuron which best match with the input vector and it is declared as winning neuron. See equation (7.5). Weights of neighboring neurons are adjusted using (7.6) and (7.7).

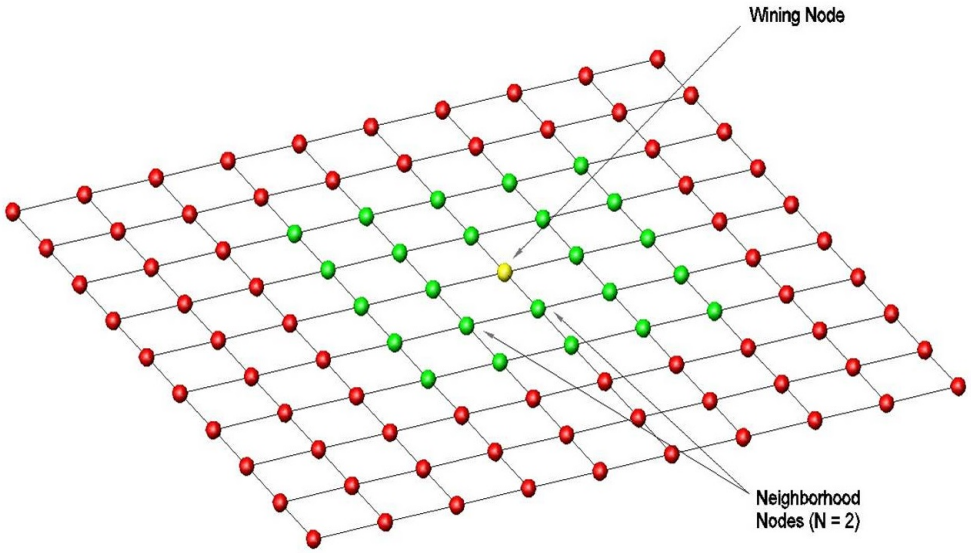

Figure 7.11: Update process

- Find the closest node to the presented feature vector. and their neighborhood nodes. Stimulate them by making them a little more like the presented feature vector as shown in Figure 7.12. After mapping input vector on neuron structure, weight of a neuron is updated. It is shown by dotted line in the Figure 7.12. Weight of neighboring neurons are also updated as per equation (7.6) and (7.7).

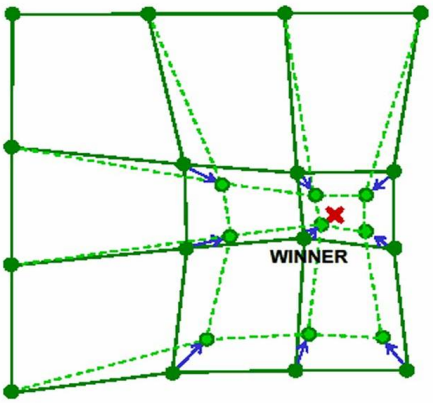

Figure 7.12: Adjusting weights of nodes

In our case, a set of stimuli described by the stimulus vector v, drives the model to adapt its feature vectors W_r by an iterative sequence of steps. At the beginning of each step a feature vector v is chosen at random according to the probability distribution $P(v)$. Using a distance measure d which in our simulations $d(\overrightarrow{v}, \overrightarrow{W}) = \left|\overrightarrow{v} - \overrightarrow{W}\right|^2$, the cell $\vec{s}$ whose feature vector $\vec{W}_s$ is closest to vector $\vec{v}$ is determined by

$$\overrightarrow{s} = \min\ d(\overrightarrow{v}, \overrightarrow{W_r}) \tag{7.5}$$

And the attached feature vectors are updated according to the SOFM rule.

$$\overrightarrow{W_r}(t+1) = \overrightarrow{W_r}(t) + \varepsilon(t)h(\overrightarrow{r}, \overrightarrow{s}, t)(\overrightarrow{v} - \overrightarrow{W_r}(t)) \tag{7.6}$$

An essential element of the equation is the presence of the 'kernel' $h(r, s, t)$ that correlates the changes of the cells at neighboring positions r, s and is given by

$$h(\overrightarrow{r}, \overrightarrow{s}, t) = exp\left(-\frac{(r_1 - s_1)^2}{\sigma_{h1}^2(t)} - \frac{(r_2 - s_2)^2}{\sigma_{h2}^2(t)}\right) \tag{7.7}$$

The equations have been shown to lead under a broad variety of feature vectors that can be characterized to the spatial fields of the feature vector W_r which can be characterized by following two conditions (i). the variation of W_r with the cell position r is as continuous as possible, and (ii). the resulting vectors W_r span the range over which the feature combinations vary in the set of input patterns. These two complementary requirements which are, favours uniformity and demands diversity for the feature vectors. The quantities σ_{h1} and σ_{h2} parametrize the shape of the kernel h and therefore, determine the range over which response properties of the cells are kept correlated. σ_{h1} and σ_{h2} are the major and axes of the neighborhood function or they are the principal and orthogonal standard deviations of the neighborhood function. During the training phase, various orientations are used to generate the matured orientation map of the brain [73]. Initial orientation preferences of the neurons are as shown in fig.7.13(a). Once the brain learns to recognize orientation patterns from the visual space, the orientation preferences of brain cells looks like shown in fig.7.13(b).

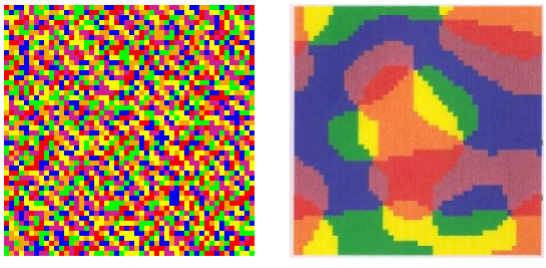

Figure 7.13: (a) Initial orientation preferences of the neurons (b) orientation preferences of the neurons in a matured brain

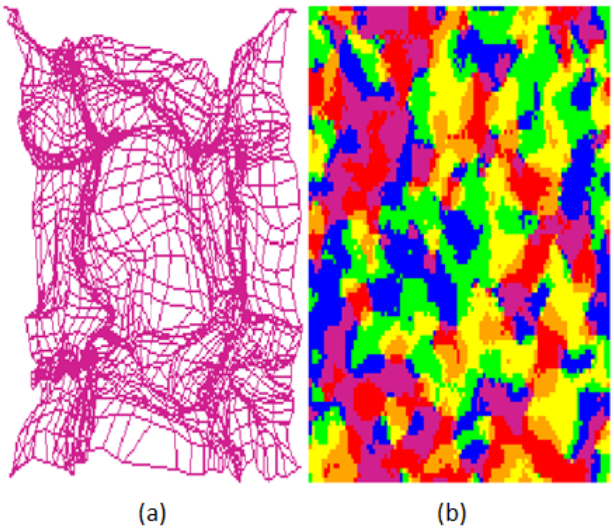

Figure 7.14: Outputs (a) Locations of the receptive field centers (b) Distribution of orientation preference

7.7 Experimental Results

Various pinwheels observed in the orientation map correspond to folding in space as observed in the network shown in Figure 7.14. Figure 7.15(a) shows the topographical representation of visual space based on the SOFM algorithm. The diagram presents the locations (x_r, y_r) of the receptive field centers in the visual space of all the cells in the network layer. Receptive field centers of the neighboring field's cells were connected by lines. An ideal topographical representation of the visual space to the network layer would give rise to a square lattice with equal mesh size in the figure. These distortions are due to constraint map of more than two dimensional feature space on a two dimensional cortical surface. Figure 7.15(b) shows the final distribution of orientation preference ϕ_r (color) and specificity q_r (saturation) along the network layer for a map generated in the regime above the threshold with appropriate neighborhood function. Outputs generated for same expression (Disgust) of different peoples is shown in Figure 7.15. Outputs generated for different expressions of same person is shown in Figure 7.16. Outputs generated for happy, surprise, and sad expressions of 10 peoples are shown in Figure 7.17, 7.18, and 7.19 respectively. For various expressions, the network folding pattern will be different. During the training phase unique folding patterns or structures will be generated for various expressions. During the testing phase, pattern having the maximum similarity with the stored pattern will be the winner, and the recognized expression will be the expression corresponding to the stored pattern. So we have used multilayer perceptron artificial neural network to classify these patterns. The neural network architecture used in the present study

consists of three layers (i) an input (ii) a hidden and (iii) an output layers. The number of neurons in input layer is 2500. By varying the number of neurons in the hidden layer, the optimum result is obtained. The variations are 800, 1200, 2000 and 3000 neurons plus one bias. The last layer is output layer and it has 7 neurons each corresponding to one facial expressions. The activation function is sigmoid bipolar for hidden layer and sigmoid for output layer. This is because the expected output value is binary i. e., 0 or 1. Learning rate is also varied (0.25 and 0.5). The target of error and maximum epoch are 0.0001 and 1000 respectively. Confusion matrix and accuracy of facial expression recognition is shown in Table 7.1. Four sets containing 25% of the data for each class, chosen randomly, were created. One set containing 25% of the samples for each class is used for the test set, while the remaining sets form the training set. After the classification procedure is performed, the samples forming the testing set are incorporated into the current training set, and a new set of samples (25% of the samples for each class) is extracted to form the new test set. The remaining samples create the new training set. This procedure is repeated four times. The average classification accuracy is the mean value of the percentages of the correctly classified facial expressions. It is 91.42%.

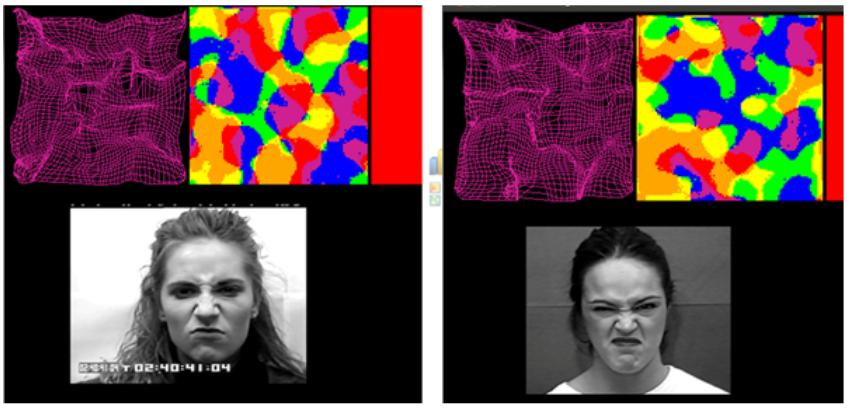

Figure 7.15: Outputs generated for same expression of different people

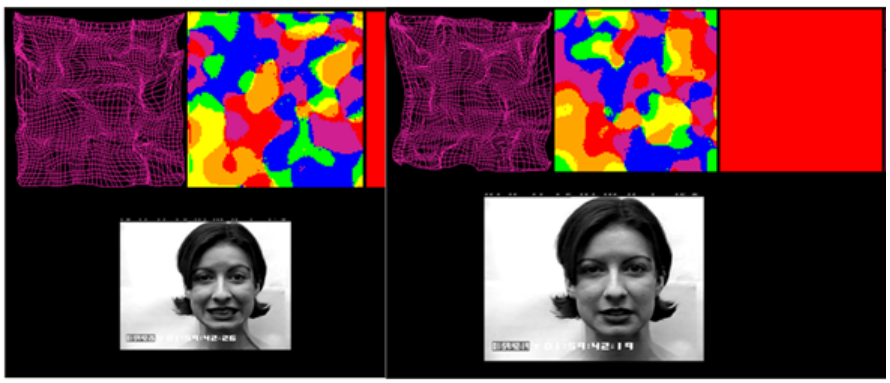

Figure 7.16: Outputs generated for different expressions of the same person

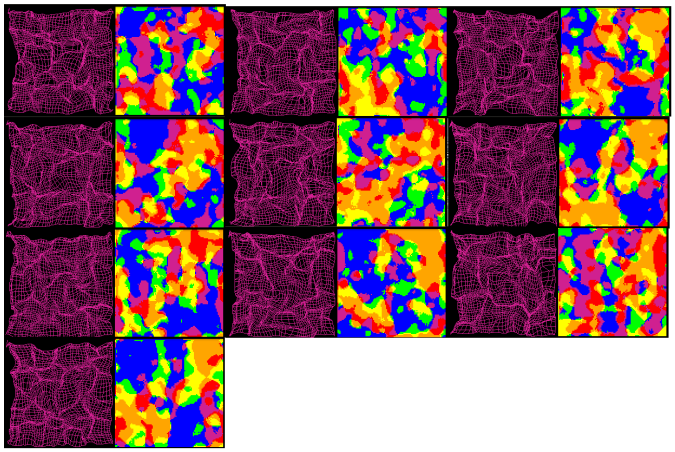

Figure 7.17: Outputs generated for happy expression of different persons

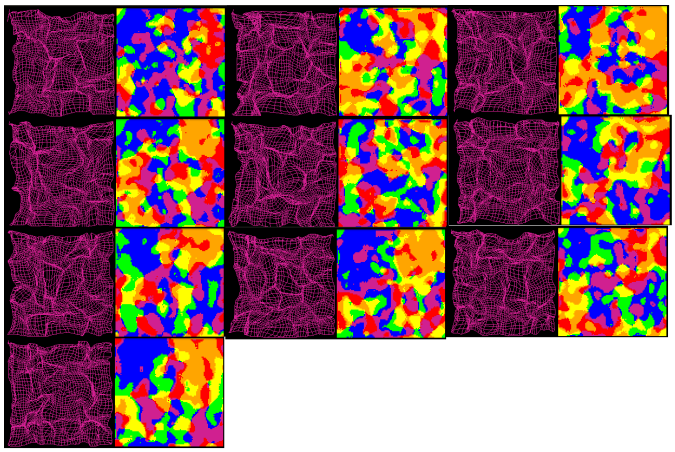

Figure 7.18: Outputs generated for surprise expression of different persons

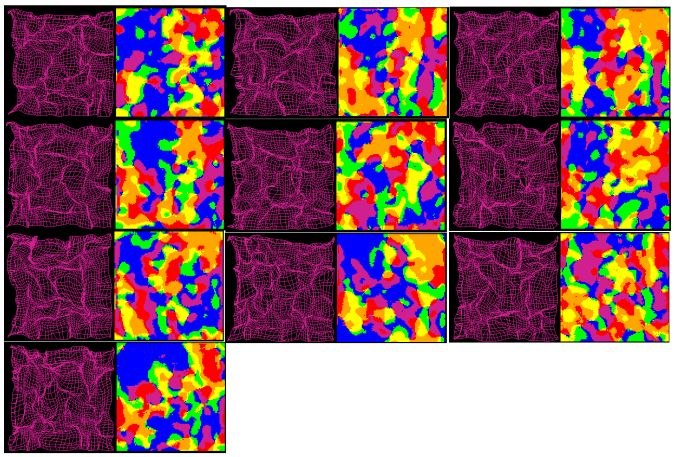

Figure 7.19: Outputs generated for sad expression of different persons

Table 7.1: Confusion Matrix and accuracy of facial expression recognition

Expression	*Happy %*	*Surprise %*	*Sad %*	*Anger %*	*Disgust %*	*Fear %*	*Neutral %*
Happy	89	2	0	0	0	2	3
Surprise	0	92	0	0	0	4	0
Sad	0	0	90	0	0	3	2
Anger	4	2	4	91	4	0	0
Disgust	0	0	0	6	94	0	0
Fear	7	4	4	3	0	91	2
Neutral	0	0	2	0	0	0	93

Chapter 8

Conclusions and Future Work

The aim of this thesis has been to explore and propose efficient algorithm for facial expression recognition and illustrate low latency verion implementation (FPGAs). Beginning with the motivation for facial behavior analysis, we reviewed this field of science, which has been extensively studied in terms of application and automation. Earlier, facial expression analysis was done manually by psychologists. Now, it will be done by suitable computer software. Extensive research is going on in this field and variety of image processing techniques is developed for the facial expression recognition. However, there are still many challenges and problems to solve in such systems, especially in the area of their performance and applicability improvement.

Ours has been one more step further in this direction. Large variety of faces, pose, and illumination are the parameters which makes these system more complicated. Apart from literature review, this work provides the design and implementation of Facial Expression Recognition System. Proposed system is developed to process the video or image sequence and recognize facial expressions. Major strengths of the system are full automation and recognize expressions from frontal as well as tilted faces with respect to y axis. Even though the system cannot handle occlusions, the head shifts are allowed. Additionally, the recognition results are quite promising with regard to the fact that only geometrical information is given to the classifier.

To summarize, we have used a wire frame model, Active appearance algorithm and Support vector machine for facial expression recognition. Our system recognize facial expressions from image sequence of frontal as well as tilted faces. We have designed algorithm for facial feature point extraction. We have developed algorithm for fitting wire frame model automatically on first frame of image sequence. In subsequent frames facial feature points are tracked using Active Appearance Approach. Here we removed the necessity of the first frame to be of neutral facial expression. As a classifier we use multiclass SVM. For classifier,

only geometrical information is given, no texture information is required. With hardware implementation we made the system real time. Execution time is considerably reduced with hardware implementation. Power saving is increased as partial reconfiguration is used.

We have used our approach to recognize nine facial expressions of Indian classical dance style "Bharatnatyam". These expressions are shringar (Erotic), hasya (Happy), karuna (Sad), raudra (Anger), veera (Heroic), bhayanak (Fearful), bibhatsa (Disgust), adbuta (Surprise), shanta (Neutral). We have created our own database. We have designed multiple kernel sparse representation classifier and showed that it outperforms the other classifiers.

Also, work is undergoing to improve the time efficiency of our system in order to make it appropriate to use in different real time (real life) applications such as patient monitoring, computerized tutor system, and human computer interaction. In AAM, for computing synthesized images, we have performed principal component analysis (PCA) on training images. For accurate model fitting on variety of images, training set should consists of as many variety of images as possible. To achieve this, in future we propose to use incremental PCA. So in real time, when model fits on new image, that image will be added in training set automatically and a new matrix of eigen face images will be generated. This will save training time and reduce the off line computations with increase in variety of images.

We have developed a mathematical model using self organizing feature map, analogous to mapping of image in the visual cortex of the brain, for facial expression recognition. We got different patterns for different expressions of different persons. We have classified these patterns using ANN and got 91% recognition accuracy. This technique is used for recognition of facial expressions from still images.

Bibliography

[1] B. Fasel and J. Luettin, "Automatic facial expression analysis: A survey," *Pattern Recognition*, vol. 36, no. 1, pp. 259–275, 2003.

[2] I. Pandzic and R. Forchheimer, "MPEG--4 Facial animation: The standard, implementation and applications," *John Wiley and sons*, 2002.

[3] K. Obermeyer and H. Ritter, "A model for the development of the spatial structure of retinotopic maps and orientation columns," *IEICE Transactions Fundamentals*, vol. 75, pp. 537–545, May 1992.

[4] M. Pantic and L. J. M. Rothkrantz, "Automatic analysis of facial expressions: The state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1424–1445, Dec. 2000.

[5] P. Ekman and W. V. Friesen, "Facial action coding system manual," *Palo Alto consulting Psychologists Press*, 1978.

[6] A. Samal and P. Iyenger, "Automatic recognition of human faces and facial expressions: A survey," *Pattern Recognition*, vol. 25, no. 1, pp. 65–77, 1992.

[7] M. Bartlett, G. Littlewort, I. Fasel, and J. R. Movellan, "Real time face detection and facial expression recognition: Development and applications to human computer interaction," *Proc. Conf. Computer Vision and Pattern Recognition Workshop, Madison, WI*, vol. 5, pp. 53–58, Jun 2003.

[8] K. Anderson and W. Peter, "A real time automated system for the recognition of human facial expressions," *IEEE Trans. on systems, Man, and Cybernetics Part B Cybernetics*, vol. 36, pp. 96–105, Feb 2006.

[9] G. Abrantes and F. Pereira, "MPEG-4 facial animation technology: survey, implementation, and results," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 9, pp. 290–305, Mar 1999.

[10] Y. Yacoob and L. Davis, "Recognizing human facial expressions from long image sequences using optical flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 636–642, June 1996.

[11] M. J. Black and Y. Yacoob, "Recognizing facial expressions in image sequences using local parameterized models of image motion," *lnt'l J. Computer Vision*, vol. 25, no. 1, pp. 23–48, 1997.

[12] S. Kimura and M. Yachida, "Facial expression recognition and its degree estimation," *Proc. Computer Vision and Pattern Recognition*, pp. 295–300, 1997.

[13] H. Wu, T. Yokoyama, D. Pramadihanto, and M. Yachinda, "Face and facial feature extraction from color image," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 343–350, 1996.

[14] I. A. Essa and A. P. Pentland, "Coding, analysis, interpretation, and recognition of facial expressionsl," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 757–763, July 1997.

[15] A. Pentland, B. Moghaddam, and T. starner, "View based and modular eigenspaces for face recognition," *Proc. Computer vision and Pattern Recognition*, pp. 84–91, 1994.

[16] E. Simoncelli, "Distributed representation and analysis of visual motion," *PhD thesis, Massachusetts Inst. of Technology*, 1993.

[17] J. F. Cohn, A. J. Ziochower, J. J. lien, and T. kanade, "Feature point tracking by optical flow discriminates subtle differences in facial expression," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 396–401, 1998.

[18] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proc. Joint Conf. Artificial Intelligence*, pp. 674–680, 1981.

[19] M. Wang, Y. Iwai, and M. Yachindai, "Expression recognition from time-sequential facial images by use of expression change model," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 324–329, 1998.

[20] J. Buhmann, J. Lange, and C. von der Malsburg, "Distortion invariant object recognition matching hierarchically labelled graphs," *Proc. Int'l Joint Conf. Neural Networks*, pp. 155–159, 1989.

[21] Y. Tian, T. Kanade, and J. Cohn, "Recognizing action units for facial expression analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 97–115, Feb 2001.

[22] I. Cohen, N. Sebe, S. Garg, L. S. Chen, and T. S. Huanga, "Facial expression recognition from video sequences: temporal and static modelling," *Comput. Vis. Image Understand*, vol. 91, no. 5, pp. 160–187, 2003.

[23] T. Hai and T. Huang, "Connected vibrations: a modal analysis approach for non-rigid motion tracking," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 735 –740, June 1998.

[24] P. Michel and R. Kaliouby, "Real time facial expression recognition in video using support vector machines," *Proc. 5th Int. Conf. Multimodal interfaces, Vancouver, BC, Canada*, pp. 258–264, 2003.

[25] M. Valstar, I. Patras, and M. Pantic, "Facial action unit recognition using temporal templates," *13th IEEE International Workshop on Robot and Human Interactive Communication*, pp. 253–258, 2004.

[26] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, 2001.

[27] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *International Journal of Computer Vision*, 1998.

[28] M. Pantic and I. Patras, "Detecting facial actions and their temporal segments in nearly frontal-view face image sequences," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 4, pp. 3358–3363, Oct 2005.

[29] W. Zheng, X. Zhou, C. Zou, and L. Zhao, "Facial expression recognition using kernel canonical correlation analysis (kcca)," *Neural Networks, IEEE Transactions on*, vol. 17, pp. 233–238, Jan 2006.

[30] I. Kotsia and I. Pitas, "Facial expression recognition in image sequences using geometric deformation features and support vector machines," *IEEE Transactions on Image Processing*, vol. 16, pp. 172–187, Jan. 2007.

[31] J. Y. Bouguet, "Pyramidal implementation of the Lucas-Kanade feature tracker," tech. rep., Intel Corporation, Microprocessor Research Labs, 1999.

[32] I. Kotsia, I. Buciu, and I. Pitas, "An analysis of facial expression recognition under partial facial image occlusion," *Image and Vision Computing*, vol. 26, no. 7, pp. 1052 – 1067, 2008.

[33] S. Lajevardi and M. Lech, "Facial expression recognition from image sequences using optimized feature selection," *23rd International Conference on Image and Vision Computing New Zealand*, pp. 1–6, 2008.

[34] P. Viola and M. Jones, "Robust real-time object detection," tech. rep., Cambridge Research Laboratory Technical report series, February 2001.

[35] Z. Ligang and D. Tjondronegoro, "Facial expression recognition using facial movement features," *Affective Computing, IEEE Transactions on*, vol. 2, pp. 219–229, Oct 2011.

[36] M. Valstar, M. Mehu, B. Jiang, M. Pantic, and K. Scherer, "Meta-analysis of the first facial expression recognition challenge," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, pp. 966–979, Aug 2012.

[37] Y. Li, S. Wang, Y. Zhao, and J. Qiang, "Simultaneous facial feature tracking and facial expression recognition," *Image Processing, IEEE Transactions on*, vol. 22, pp. 2559–2573, July 2013.

[38] R. A. Khan, A. Meyer, H. Konik, and S. Bouakaz, "Framework for reliable, real-time facial expression recognition for low resolution images," *Pattern Recognition Letters*, vol. 34, no. 10, pp. 1159 – 1168, 2013.

[39] H. Fang, N. M. Parthalain, A. J. Aubrey, G. K. Tam, R. Borgo, P. L. Rosin, P. W. Grant, D. Marshall, and M. Chen, "Facial expression recognition in dynamic sequences: An integrated approach," *Pattern Recognition*, vol. 47, no. 3, pp. 1271 – 1281, 2014.

[40] S. Canavan, X. Zhang, and L. Yin, "Fitting and tracking 3d/4d facial data using a temporal deformable shape model," in *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pp. 1–6, July 2013.

[41] T. Wu, M. Bartlett, and J. R. Movellan, "Facial expression recognition using gabor motion energy filters," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pp. 42–47, June 2010.

[42] J. Whitehill, M. Bartlett, and J. Movellan, "Automatic facial expression recognition for intelligent tutoring systems," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pp. 1–6, June 2008.

[43] N. Sebe, M. Lew, I. Cohen, Y. Sun, T. Gevers, and T. Huang, "Authentic facial expression analysis," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pp. 517–522, May 2004.

[44] M. Valstar, M. Mehu, B. Jiang, M. Pantic, and K. Scherer, "Meta-analysis of the first facial expression recognition challenge," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, pp. 966–979, Aug 2012.

[45] J. Sung, S. Lee, and D. Kim, "A real-time facial expression recognition using the staam," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1, pp. 275–278, 2006.

[46] G. Littlewort, J. Whitehill, T. Wu, I. Fasel, M. Frank, J. Movellan, and M. Bartlett, "The computer expression recognition toolbox (cert)," in *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 298–305, March 2011.

[47] S. Koelstra, M. Pantic, and I. Patras, "A dynamic texture-based approach to recognition of facial actions and their temporal models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 1940–1954, Nov 2010.

[48] M. Valstar and M. Pantic, "Fully automatic recognition of the temporal phases of facial actions," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, pp. 28–43, Feb 2012.

[49] M. H. Nguyen, T. Simon, F. De la Torre, and J. Cohn, "Action unit detection with segment-based SVMs," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[50] J. Cohn, A. Zlochower, J. Lien, and T. Kanade, "Feature-point tracking by optical flow discriminates subtle differences in facial expression," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 396–401, 1998.

[51] Y. Li, S. Wang, Y. Zhao, and Q. Ji, "Simultaneous facial feature tracking and facial expression recognition," *Image Processing, IEEE Transactions on*, vol. 22, pp. 2559–2573, July 2013.

[52] D. Anguita, A. Boni, and S. Ridella, "A digital architecture for support vector machines: theory, algorithm, and FPGA implementation," *Neural Networks, IEEE Transactions on*, vol. 14, pp. 993–1009, Sept 2003.

[53] I. Biasi, A. Boni, and A. Zorat, "A reconfigurable parallel architecture for SVM classification," in *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 5, pp. 2867–2872, July 2005.

[54] A. Boni, F. Pianegiani, and D. Petri, "Low-power and low-cost implementation of SVMs for smart sensors," *Instrumentation and Measurement, IEEE Transactions on*, vol. 56, pp. 39–44, Feb 2007.

[55] C. Kyrkou and T. Theocharides, "Scope: Towards a systolic array for SVM object detection," *Embedded Systems Letters, IEEE*, vol. 1, pp. 46 –49, aug. 2009.

[56] D. Mahmoodi, A. Soleimani, H. Khosravi, and M. Taghizadeh, "FPGA simulation of linear and nonlinear support vector machine," *Journal of Software Engineering and Applications*, vol. 4, pp. 320–328, Feb 2011.

[57] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition, Proceedings of the 2001 IEEE Computer Society Conference on*, pp. 511–518, February 2001.

[58] T. F. Cootes, G. Edwards, and C. J. Taylor, "Active appearance models," *Proc. 5th European Conference on Computer vision*, pp. 484–498, 1998.

[59] T. Kanade, J. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," *Proc.IEEE Int. Conf. Face and Gesture Recognition*, pp. 46–53, March 2000.

[60] M. M. Nordstrøm, M. Larsen, J. Sierakowski, and M. B. Stegmann, "The IMM face database - an annotated dataset of 240 face images," may 2004.

[61] D. Vukadinovic and M. Pantic, "Fully automatic facial feature point detection using gabor feature based boosted classifiers," in *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, vol. 2, pp. 1692 – 1698 Vol. 2, oct. 2005.

[62] J. Ahlberg, "Candide-3 an updated parameterized face.," Tech. Rep. Report No. LiTH-ISY-R-2326, Dept. of EE, Linkoping University, 2001.

[63] J. Ahlberg, "Wincandide 1.3 user's manual," Tech. Rep. Report No. LiTH-ISY-R-2344, Dept. of EE, Linkoping University, 2001.

[64] J. Ahlberg, "Fast image warping for active models," Tech. Rep. Report No. LiTH-ISY-R-2355, Dept. of EE, Linkoping University, 2001.

[65] E. W. Weisstein, "Barycentric coordinates," *MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/BarycentricCoordinates.html.*

[66] J. Ahlberg, "An active model for facial feature tracking," *EURASIP Journal on Applied Signal processing*, vol. 2002, pp. 566–571, 2001.

[67] D. Michie, D. Spiegelhalter, and C. C. Taylor, *Machine Learning, neural and statistical classification.* New York Ellis Horwoodl, 1994.

[68] L. Cheng, J. Zhang, J. Yang, and J. Ma, "An improved hieraarchical multiclass support vector machine with binary tree architecture," *International conference on internet computing in science and engineering*, pp. 106–109, Jan 2008.

[69] A. Zeineddini and K. Gaj, "Secure partial reconfiguration of FPGAs," in *IEEE International Conference on Field-Programmable Technology*, pp. 155 –162, Dec. 2005.

[70] L. Zhang, W. D. Zhou, P. C. Chang, J. Liu, Z. Yan, T. Wang, and F. Z. Li, "Kernel sparse representation-based classifier," *IEEE Transactions on Signal Processing*, vol. 60, pp. 1684–1695, April 2012.

[71] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 210–227, Feb 2009.

[72] A. Shrivastava, V. M. Patel, and R. Chellappa, "Multiple kernel learning for sparse representation-based classification," *IEEE Transactions on Image Processing*, vol. 23, pp. 3013–3024, July 2014.

[73] K. Obrmayer, H. Ritter, and K. Schulten, "A principle for the formation of spatial structure of cortical feature maps," *Proc. Natural ACD Science*, vol. 87, pp. 8345–8349, Nov. 1990.

[74] G. Guo and C. R. Dyer, "Learning from examples in the small sample case: Face expression recognition," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 35, pp. 477–488, Jun 2005.

[75] K. Anderson and P. W. McOwan, "A real-time automated system for the recognition of human facial expressions," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, pp. 96–105, Feb 2006.

[76] I. Cohen, N. Sebe, A. Garg, L. S. Chen, and T. S. Huang, "Facial expression recognition from video sequences: Temporal and static modeling," *Computer Vision and Image Understanding*, vol. 91, pp. 160–187, 2003.

[77] B. Welsh, "Model based coding of images," *Ph.D. dessertation, British Telecom Research Lab*, Jan 1991.

[78] R. Patil, V. Sahula, and A. Mandal, "Facial expression recognition in image sequences using active shape model and svm," *Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS)*, pp. 168–173, Sept. 2011.

[79] R. Patil, V. Sahula, and A. Mandal, "Automatic detection of facial feature points in image sequences," *International conference on Image information processing ICIIP*, Nov. 2011.

[80] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy, "SVM and kernel methods matlab toolbox," *Perception Systems Information, INSA de Rouen, France*, 2005.

[81] H. Liu, Y. wei Huang, and D. Liu, "Multi-class surface EMG classification using support vector machines and wavelet transform," in *8th World Congress on Intelligent Control and Automation*, pp. 2963 –2967, July 2010.

[82] A. Mathur and G. Foody, "Multiclass and binary SVM classification: Implications for training and classification users," *Geoscience and Remote Sensing Letters, IEEE*, vol. 5, pp. 241 –245, april 2008.

[83] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw*, vol. 13, pp. 415–425, March 2002.

[84] J. Weston and C. Watkins, "Multi-class support vector machines," Tech. Rep. CSD-TR-98-04, 2004.

[85] J. Strom, F. Davoine, and J. Ahlberg, "Very low bit rate facial texture coding," *Proc. Int. Workshop on Synthetic/Natural Hybrid Coding and 3D Imaging*, pp. 237–240, September 1997.

[86] R. C. Gonzalez and R. E. Woods, "Digital image processing,"

[87] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," *National Taiwan University, Taipei 106, Taiwan.*

[88] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Disc*, vol. 2, no. 2, 1998.

[89] G. Edwards, T. Cootes, and C. Taylor, "Face recognition using active appearance models," *Proc. European Conf. Computer Vision*, vol. 2, pp. 581–695, 1998.

[90] C. L. Huang and Y. M. Huang, "Facial expression recognition using model based feature extraction and action parameters classification," *J. Visual Comm. and Image Representation*, vol. 8, no. 3, pp. 278–290, 1997.

[91] H. Kobayashi and F. Hara, "Facial interaction between animated 3d face robot and human beings," *J. Visual Comm. and Image Representation*, pp. 3723–3737, 1997.

[92] M. Iain and S. Baker, "Active appearance model revisited," *Int'l J. Computer Vision*, vol. 60, no. 2, pp. 135–164, 2004.

[93] T. Otsuka and J. Ohya, "Spotting segments displaying facial expression from image sequences using hidden markov model," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 442–447, 1998.

[94] M. Yoneyama, Y. Iwano, A. Ohtake, and K. Shiraia, "Facial expressions recognition using discrete hopfield neural networks," *Proc. Int'l Conf. Information Processing*, vol. 3, pp. 117–120, 1997.

[95] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models-training and application," *Computer Vision Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[96] M. Pantic and L. M. Rothkrantz, "Expert system for automatic analysis of facial expression," *Image and vision computing*, vol. 18, no. 11, pp. 881–905, 2000.

[97] H. Hong, H.Neven, and C. von der Malsburg, "Online facial expression recognition based on personalized gallaries," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 354–359, 1998.

[98] I. Steffens, E. Elagin, and H. Neven, "Personspotter-fast and robust system for human detection, tracking and recognition," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 516–521, 1998.

[99] L. Wiskott, "Labelled graphs and dynamic link matching for face recognition and scene analysis," *Relihe Physik*, vol. 53, 1995.

[100] C. Padgett and G. W. Cottrell, "Representing face images for emotion classification," *Pro. Conf. Advances in Neural Information Processing Systems*, pp. 894–900, 1996.

[101] B. Horn and B. Schunck, "Determing optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[102] Z. Zhang, M. Lyons, M. Schuster, and S. Akamatsu, "Comparision between geometry based and gabor wavelets based facial expression recognition using multilayer perceptron," *Proc. Int'l Conf. Automatic face and Gesture Recognition*, pp. 454–459, 1998.

[103] M. J. Lyons, J. Budynek, and S. Akamatsu, "Automatic classification of single facial images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 1357–1362, 1999.

[104] M. J. Black and Y. Yacoob, "Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motions," *Int'l J. Computer Vision*, pp. 374–381, 1995.

[105] T. Otsuka and J. Ohya, "Recognition of facial expressions using hmm with continuous output probabilities," *Proc. Int'l Workshop Robot and Human Comm*, pp. 323–328, 1996.

[106] M. Riedmiller and H.Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," *Proc.Int'l Conf. Neural Networks*, pp. 586–591, 1993.

[107] I. Lien, T. Kanade, J. F. Cohn, and C. C. Li, "Automated facial expression recognition based on facs action units," *Proc. Int'l Conf. Automatic Face and Gesture Recognition*, pp. 390–395, 1998.

[108] Y. Zhang and Q. Ji, "Active and dynamic information fusion for facial expression understanding from image sequences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 699–714, May 2005.

[109] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," *Proc. 3rd IEEE Int. Conf. Automatic Face and Gesture Recognition*, pp. 200–205, 1998.

[110] L. Wiskott, J. Fellous, N. Kruger, and C. V. D. Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 775–779, July 1997.

[111] Y. Gao, M. Leung, S. Hui, and M. Tananda, "Facial expression recognition from line-based caricatures," *IEEE Trans. Syst., Man, Cybern. A: Syst. Humans*, vol. 33, pp. 407–412, May 2003.

[112] B. Abboud, F. Davoine, and M. Dang, "Facial expression recognition and synthesis based on an appearance model," *Signal Process.: Image Commun.*, vol. 19, no. 8, pp. 723–740, 2004.

[113] K. R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, pp. 181–201, March 2001.

[114] S. B. Gokturk, C. Tomasi, B. Girod, and J.-Y. Bouguet, "Model-based face tracking for view-independent facial expression recognition," *Proc. 5th IEEE Int. Conf. Automatic Face and Gesture Recognition.*, pp. 287–293, May 2002.

[115] M. Pontil and A. Verri, "Support vector machines for 3d object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 637–646, Jun 1998.

[116] H. Drucker, W. Donghui, and V. Vapniki, "Support vector machines for spam categorization," *IEEE Trans. Neural Netw.*, vol. 10, pp. 1048–1054, Sept 1999.

[117] A. Tefas, C. Kotropoulos, and I. Pitas, "Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 735–746, July 2001.

[118] I. Mpiperis, S. Malassiotis, and M. Strintzis, "Bilinear models for 3d face and facial expression recognition," *IEEE Trans. Information forensics and security*, vol. 3, pp. 498–511, Sept 2008.

[119] Y. Ming-Hsuan, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, 2002.

[120] L. Guanming, L. Xiaonan, and L. Haibo, "Facial expression recognition for neonatal pain assessment," *IEEE Int. Conf Neural Networks and Signal Processing*, pp. 456–460, Jun 2008.

[121] C. Shan, G. Shaogang, and W. Peter, "Robust facial expression recognition using local binary patterns," *IEEE Int. Conf. Image Processing*, vol. 2, pp. 11–14, Sept 2005.

[122] A. Lanitis, C. Taylor, and T. Cootes, "Automatic interpretation and coding of face images using flexible models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 743–756, 1997.

[123] G. Fanelli and M. Fratarcangeli, "A non-invasive approach for driving virtual talking heads from real facial movements," *3DTV Conference*, pp. 1–4, 2007.

[124] J. Mihalik and M. Kasar, "Human face and facial feature tracking by using geometric and texture models," *Journal of Electrical Engineering*, vol. 59, no. 5, pp. 266–271, 2008.

[125] J. Mihalik and M. Kasar, "Basis of eigenfaces for tracking of human head," *Journal of Electrical Engineering*, vol. 58, no. 3, pp. 134–139, 2007.

[126] W. J., B. M., and J. Movellan, "Automatic facial expression recognition for intelligent tutoring systems," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08*, pp. 1–6, June 2008.

[127] A. Geetha, V. Ramalingam, S. Palanivel, and B. Palaniappan, "Facial expression recognition a real time approach," *International journal on Expert systems with applications*, vol. 36, pp. 303–308, Jan 2009.

[128] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image and vision computing*, vol. 27, pp. 803–816, May 2009.

[129] C. Kyrkou and T. Theocharides, "Scope towards a systolic array for svm object detection," *IEEE Embedded Systems Letters*, vol. 1, pp. 46–49, Aug 2009.

[130] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 23–38, 1998.

[131] C. J. Lien, T. Kanade, J. F. Cohn, and C. Li, "Detection, tracking, and classification of action units in facial expressions," *J. Robot Auton. Sys.*, July 1999.

[132] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," *Neural Network.*, vol. 16, pp. 555–559, July 2003.

[133] L. Ma and K. Khorasani, "Facial expression recognition using constructive feedforward neural networks," *IEEE Trans. Syst. Man Cybern. B. Cybern.*, vol. 34, pp. 1588–1595, Jun 2004.

[134] Intel, "Open cv: Open Source Computer Vision Library," *http://www.intel.com /research/mrl/research/opencv/*.

[135] H. Li, J. Buenaposada, and L. Baumela, "Real time facial expression recognition with illumination corrected image sequences," *Automatic Face Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on*, pp. 1–6, Sept. 2008.

Appendix

Candide wire frame model

VERTEX LIST: 113

(x y z) starting with vertex number 0 to 112

0.000000 1.061000 -0.371000
0.174000 0.800000 -0.024000
0.000000 0.539000 0.085000
0.000000 0.278000 0.107000
0.000000 0.213000 0.085000
0.000000 -0.222000 0.210000
0.000000 -0.265000 0.124000
0.000000 -0.417000 0.142000
0.000000 -0.526000 0.150000
0.000000 -0.591000 0.107000
0.000000 -0.852000 0.063000
0.217000 1.039000 -0.371000
0.457000 0.909000 -0.328000
0.435000 0.626000 -0.111000
0.610000 0.539000 -0.328000
0.522000 0.278000 -0.111000
0.391000 0.374000 0.0300000
0.130000 0.278000 0.107000
0.391000 0.322000 0.0300000
0.304000 0.225000 -0.002000
0.470000 0.148000 -0.111000
0.304000 0.204000 -0.000000
0.304000 0.122000 -0.000000
0.130000 0.148000 -0.000000
0.304000 0.104000 -0.000000
0.109000 -0.157000 0.037000

0.174000 -0.244000 0.037000
0.387000 -0.100000 -0.045000
0.550000 -0.250000 -0.328000
0.609000 0.148000 -0.328000
0.470000 -0.600000 -0.328000
0.246000 -0.461000 -0.000000
0.174000 -0.809000 0.000000
0.043000 -0.396000 0.150000
-0.174000 0.800000 -0.024000
0.000000 0.539000 0.085000
0.000000 0.278000 0.107000
0.000000 0.213000 0.085000
0.000000 -0.222000 0.210000
0.000000 -0.265000 0.124000
0.000000 -0.461000 0.124000
0.000000 -0.526000 0.150000
0.000000 -0.591000 0.107000
0.000000 -0.852000 0.063000
-0.217000 1.039000 -0.371000
-0.457000 0.909000 -0.328000
-0.435000 0.626000 -0.111000
-0.610000 0.539000 -0.328000
-0.522000 0.278000 -0.111000
-0.391000 0.374000 0.030000
-0.130000 0.278000 0.107000
-0.391000 0.322000 0.030000
-0.304000 0.225000 -0.002000
-0.470000 0.148000 -0.111000
-0.304000 0.204000 -0.000000
-0.304000 0.122000 -0.000000
-0.130000 0.148000 -0.000000
-0.304000 0.104000 -0.000000
-0.109000 -0.157000 0.037000
-0.174000 -0.244000 0.037000
-0.387000 -0.100000 -0.045000
-0.550000 -0.250000 -0.328000
-0.609000 0.148000 -0.328000
-0.470000 -0.600000 -0.328000
-0.246000 -0.461000 -0.000000

-0.174000 -0.809000 0.000000
-0.043000 -0.396000 0.150000
0.348000 0.200000 -0.0300000
0.348000 0.115000 -0.0300000
-0.348000 0.200000 -0.0300000
-0.348000 0.115000 -0.0300000
0.265000 0.200000 -0.0300000
0.265000 0.115000 -0.0300000
-0.265000 0.200000 -0.0300000
-0.265000 0.115000 -0.0300000
0.080000 -0.220000 0.150000
-0.080000 -0.220000 0.150000
0.022000 0.213000 0.063000
-0.022000 0.213000 0.063000
0.123000 -0.410000 0.063000
-0.123000 -0.410000 0.063000
0.100000 -0.461000 0.050000
-0.100000 -0.461000 0.050000
0.100000 -0.461000 0.050000
-0.100000 -0.461000 0.050000
0.123000 -0.508000 0.063000
-0.123000 -0.508000 0.063000
0.000000 -0.461000 0.124000
0.200000 -0.461000 -0.024000
-0.200000 -0.461000 -0.024000
0.357000 -0.461000 -0.050000
-0.357000 -0.461000 -0.050000
0.065000 0.028000 0.050000
-0.065000 0.028000 0.050000
0.000000 0.068000 0.100000
0.387000 0.201000 -0.056000
-0.387000 0.201000 -0.056000
0.387000 0.186000 -0.056000
-0.387000 0.186000 -0.056000
0.387000 0.126000 -0.056000
-0.387000 0.126000 -0.056000
0.387000 0.117000 -0.067000
-0.387000 0.117000 -0.067000
0.217000 0.201000 -0.013000

-0.217000 0.201000 -0.013000
0.217000 0.186000 -0.013000
-0.217000 0.186000 -0.013000
0.217000 0.126000 -0.013000
-0.217000 0.126000 -0.013000
0.217000 0.117000 -0.024000
-0.217000 0.117000 -0.024000
0.120000 -0.265000 0.100000
-0.120000 -0.265000 0.100000

FACE LIST: 184

(vertex1, vertex2, vertex3) Triangular face is formed between vertex1, vertex2, vertex3.

0 11 1
0 1 34
0 34 44
11 12 1
44 34 45
1 12 13
1 13 2
1 2 34
2 46 34
34 46 45
12 14 13
13 14 15
13 15 16
2 13 16
2 16 17
2 17 3
2 3 50
2 50 49
2 49 46
46 49 48
46 48 47
45 46 47
14 29 15
15 29 20
18 15 19

18 16 15
16 18 17
17 18 19
17 23 77
3 17 77
3 78 50
78 56 50
50 52 51
49 50 51
48 49 51
48 51 52
48 53 62
47 48 62
29 28 27
20 29 27
24 26 25
57 58 59
53 60 62
62 60 61
111 26 33
75 26 111
75 25 26
76 59 58
76 112 59
112 66 59
6 33 7
6 7 66
9 32 10
9 10 65
6 76 5
6 5 75
3 77 78
7 33 79
7 79 81
7 81 87
7 80 66
7 82 80
7 87 82
80 82 89

80 89 64
79 88 81
79 31 88
26 79 33
26 31 79
59 66 80
59 80 64
88 83 85
88 85 31
83 8 85
83 40 8
8 40 84
8 84 86
86 89 84
86 64 89
9 85 8
9 8 86
32 85 31
32 9 85
65 86 64
65 9 86
27 26 24
90 30 32
90 32 31
90 30 28
90 26 31
90 27 26
90 28 27
60 59 57
91 65 63
91 64 65
91 61 63
91 64 59
91 59 60
91 60 61
92 77 23
92 23 25
92 25 75
93 56 78

93 76 58
93 58 56
94 77 92
94 92 75
94 75 5
94 5 76
94 76 93
94 93 78
94 78 77
20 95 15
20 97 95
20 101 99
20 27 101
95 19 15
95 21 19
95 97 21
101 27 24
101 24 22
101 22 99
23 103 17
23 105 103
23 109 107
23 25 109
103 17 19
103 19 21
103 21 105
109 107 22
109 22 24
109 24 25
56 104 50
56 106 104
56 110 108
56 58 110
104 52 50
104 54 52
104 106 54
110 55 108
110 57 55
110 58 57

53 48 96
53 98 96
53 100 102
53 102 60
96 48 52
96 52 54
96 54 98
102 100 55
102 55 57
102 57 60
111 6 75
111 33 6
112 76 6
112 6 66
73 74 70
73 70 69
67 68 72
67 72 71
53 69 70
56 74 73
23 71 72
20 68 67
98 69 53
54 69 98
54 73 69
54 106 73
106 56 73
56 108 74
55 74 108
55 70 74
55 100 70
100 53 70
20 67 97
97 67 21
21 67 71
21 71 105
105 71 23
20 99 68
68 99 22

22 72 68

22 107 72

107 23 72

ANIMATION UNITS LIST: 65

vertex number (x, y, z)

AUV0 Upper lip raiser (AU10)

10

7 0.000000 0.086957 0.021739

33 0.000000 0.065217 0.021739

66 0.000000 0.065217 0.021739

79 0.000000 0.050000 0.021739

80 0.000000 0.050000 0.021739

81 0.000000 0.050000 0.021739

82 0.000000 0.050000 0.021739

87 0.000000 0.065217 0.021739

88 0.000000 0.020000 0.000000

89 0.000000 0.020000 0.000000

AUV11 Jaw drop (AU26/27)

12

40 0.000000 -0.260000 -0.050000

8 0.000000 -0.260000 -0.050000

9 0.000000 -0.260000 -0.100000

10 0.000000 -0.130000 -0.150000

32 0.000000 -0.150000 -0.130000

65 0.000000 -0.150000 -0.130000

83 0.000000 -0.200000 -0.050000

84 0.000000 -0.200000 -0.050000

85 0.000000 -0.200000 -0.050000

86 0.000000 -0.200000 -0.050000

88 0.000000 -0.020000 0.000000

89 0.000000 -0.020000 0.000000

AUV2 Lip stretcher (AU20)

18

31 0.090000 0.000000 -0.090000

8 0.000000 0.032500 -0.017391

33 0.000000 -0.022000 -0.025500

7 0.000000 -0.022000 -0.010000

64 -0.090000 0.000000 -0.090000
66 0.000000 -0.022000 -0.025500
79 0.045000 -0.020000 -0.020000
80 -0.045000 -0.020000 -0.020000
81 0.040000 0.000000 -0.020000
82 -0.040000 0.000000 -0.020000
83 0.040000 0.000000 -0.020000
84 -0.040000 0.000000 -0.020000
85 0.045000 0.023000 -0.020000
86 -0.045000 0.023000 -0.020000
88 0.080000 0.000000 -0.080000
89 -0.080000 0.000000 -0.080000
90 0.040000 0.000000 -0.040000
91 -0.040000 0.000000 -0.040000
AUV3 Brow lowerer (AU4)
14
17 -0.130435 -0.130435 0.000000
16 -0.086957 -0.130435 0.017391
18 -0.086957 -0.130435 0.017391
15 0.000000 -0.065217 0.000000
50 0.130435 -0.130435 0.000000
49 0.086957 -0.130435 0.017391
51 0.086957 -0.130435 0.017391
48 0.000000 -0.065217 0.000000
21 0.000000 -0.034783 0.000000
54 0.000000 -0.034783 0.000000
67 0.000000 -0.026087 0.000000
69 0.000000 -0.026087 0.000000
71 0.000000 -0.026087 0.000000
73 0.000000 -0.026087 0.000000
AUV14 Lip corner depressor (AU13/15)
14
31 0.000000 -0.140000 -0.010000
64 0.000000 -0.140000 -0.010000
88 0.000000 -0.100000 -0.008000
89 0.000000 -0.100000 -0.008000
79 0.000000 -0.030000 -0.020000
80 0.000000 -0.030000 -0.020000
81 0.000000 -0.030000 -0.020000

82 0.000000 -0.030000 -0.020000
83 0.000000 -0.030000 -0.020000
84 0.000000 -0.030000 -0.020000
85 0.000000 -0.040000 -0.020000
86 0.000000 -0.040000 -0.020000
90 0.000000 -0.040000 -0.000000
91 0.000000 -0.040000 -0.000000
AUV5 Outer brow raiser (AU2)
8
15 0.021739 0.173913 -0.021739
16 0.000000 0.152174 -0.021739
17 0.000000 0.021739 0.000000
18 0.000000 0.152174 -0.021739
48 -0.021739 0.173913 -0.021739
49 0.000000 0.152174 -0.021739
50 0.000000 0.021739 0.000000
51 0.000000 0.152174 -0.021739
AUV6 Eyes closed (AU42/43/44/45)
12
21 0.000000 -0.062000 0.010000
22 0.000000 0.020000 0.010000
54 0.000000 -0.062000 0.010000
55 0.000000 0.020000 0.010000
97 0.000000 -0.045000 0.007000
98 0.000000 -0.045000 0.007000
99 0.000000 0.015000 0.007000
100 0.000000 0.015000 0.007000
105 0.000000 -0.045000 0.007000
106 0.000000 -0.045000 0.007000
107 0.000000 0.015000 0.007000
108 0.000000 0.015000 0.007000
AUV7 Lid tightener (AU7)
12
21 0.000000 -0.056000 0.010000
22 0.000000 0.026000 0.010000
54 0.000000 -0.056000 0.010000
55 0.000000 0.026000 0.010000
97 0.000000 -0.038000 0.007000
98 0.000000 -0.038000 0.007000

99 0.000000 0.022000 0.007000
100 0.000000 0.022000 0.007000
105 0.000000 -0.038000 0.007000
106 0.000000 -0.038000 0.007000
107 0.000000 0.022000 0.007000
108 0.000000 0.022000 0.007000
AUV8 Nose wrinkler (AU9)
23
2 0.000000 -0.086957 0.013043
3 0.000000 -0.043478 0.000000
5 0.000000 0.086957 0.000000
26 0.000000 0.043478 -0.017391
25 0.000000 0.043478 -0.008696
24 0.000000 0.017391 0.000000
22 0.000000 0.017391 0.000000
59 0.000000 0.043478 -0.017391
58 0.000000 0.043478 -0.008696
57 0.000000 0.017391 0.000000
55 0.000000 0.017391 0.000000
68 0.000000 0.008696 0.000000
70 0.000000 0.008696 0.000000
72 0.000000 0.008696 0.000000
74 0.000000 0.008696 0.000000
99 0.000000 0.017391 0.000000
100 0.000000 0.017391 0.000000
101 0.000000 0.017391 0.000000
102 0.000000 0.017391 0.000000
107 0.000000 0.017391 0.000000
108 0.000000 0.017391 0.000000
109 0.000000 0.017391 0.000000
110 0.000000 0.017391 0.000000
AUV9 Lip presser (AU23/24)
8
8 0.000000 0.032500 0.000000
33 0.000000 -0.020000 0.000000
66 0.000000 -0.020000 0.000000
7 0.000000 -0.021000 0.000000
79 0.000000 -0.020000 0.000000
80 0.000000 -0.020000 0.000000

85 0.000000 0.023000 0.000000

86 0.000000 0.023000 0.000000

AUV10 Upper lid raiser (AU5)

6

21 0.000000 0.030000 -0.010000

54 0.000000 0.030000 -0.010000

97 0.000000 0.015000 -0.007000

98 0.000000 0.015000 -0.007000

105 0.000000 0.015000 -0.007000

106 0.000000 0.015000 -0.007000

SHAPE UNITS LIST: 14

vertex number (x, y, z)

Head height

16

0 0.000000 0.200000 0.000000

1 0.000000 0.200000 0.000000

11 0.000000 0.200000 0.000000

12 0.000000 0.200000 0.000000

13 0.000000 0.200000 0.000000

14 0.000000 0.200000 0.000000

34 0.000000 0.200000 0.000000

44 0.000000 0.200000 0.000000

45 0.000000 0.200000 0.000000

46 0.000000 0.200000 0.000000

47 0.000000 0.200000 0.000000

10 0.000000 -0.200000 0.000000

30 0.000000 -0.100000 0.000000

63 0.000000 -0.100000 0.000000

32 0.000000 -0.200000 0.000000

65 0.000000 -0.200000 0.000000

Eyebrows vertical position

8

15 0.000000 0.100000 0.000000

16 0.000000 0.100000 0.000000

17 0.000000 0.100000 0.000000

18 0.000000 0.100000 0.000000

48 0.000000 0.100000 0.000000

49 0.000000 0.100000 0.000000
50 0.000000 0.100000 0.000000
51 0.000000 0.100000 0.000000
Eyes vertical position
36
19 0.000000 0.100000 0.000000
20 0.000000 0.100000 0.000000
21 0.000000 0.100000 0.000000
22 0.000000 0.100000 0.000000
23 0.000000 0.100000 0.000000
24 0.000000 0.100000 0.000000
52 0.000000 0.100000 0.000000
53 0.000000 0.100000 0.000000
54 0.000000 0.100000 0.000000
55 0.000000 0.100000 0.000000
56 0.000000 0.100000 0.000000
57 0.000000 0.100000 0.000000
67 0.000000 0.100000 0.000000
68 0.000000 0.100000 0.000000
69 0.000000 0.100000 0.000000
70 0.000000 0.100000 0.000000
71 0.000000 0.100000 0.000000
72 0.000000 0.100000 0.000000
73 0.000000 0.100000 0.000000
74 0.000000 0.100000 0.000000
95 0.000000 0.100000 0.000000
96 0.000000 0.100000 0.000000
97 0.000000 0.100000 0.000000
98 0.000000 0.100000 0.000000
99 0.000000 0.100000 0.000000
100 0.000000 0.100000 0.000000
101 0.000000 0.100000 0.000000
102 0.000000 0.100000 0.000000
103 0.000000 0.100000 0.000000
104 0.000000 0.100000 0.000000
105 0.000000 0.100000 0.000000
106 0.000000 0.100000 0.000000
107 0.000000 0.100000 0.000000
108 0.000000 0.100000 0.000000

109 0.000000 0.100000 0.000000
110 0.000000 0.100000 0.000000
Eyes, width
20
20 0.100000 0.000000 0.000000
95 0.050000 0.000000 0.000000
97 0.050000 0.000000 0.000000
99 0.050000 0.000000 0.000000
101 0.050000 0.000000 0.000000
103 -0.050000 0.000000 0.000000
105 -0.050000 0.000000 0.000000
107 -0.050000 0.000000 0.000000
109 -0.050000 0.000000 0.000000
23 -0.100000 0.000000 0.000000
53 -0.100000 0.000000 0.000000
96 -0.050000 0.000000 0.000000
98 -0.050000 0.000000 0.000000
100 -0.050000 0.000000 0.000000
102 -0.050000 0.000000 0.000000
104 0.050000 0.000000 0.000000
106 0.050000 0.000000 0.000000
108 0.050000 0.000000 0.000000
110 0.050000 0.000000 0.000000
56 0.100000 0.000000 0.000000
Eyes, height
24
19 0.000000 0.100000 0.000000
21 0.000000 0.050000 0.000000
22 0.000000 -0.050000 0.000000
24 0.000000 -0.100000 0.000000
52 0.000000 0.100000 0.000000
54 0.000000 0.050000 0.000000
55 0.000000 -0.050000 0.000000
57 0.000000 -0.100000 0.000000
95 0.000000 0.070000 0.000000
96 0.000000 0.070000 0.000000
103 0.000000 0.070000 0.000000
104 0.000000 0.070000 0.000000
101 0.000000 -0.070000 0.000000

102 0.000000 -0.070000 0.000000
109 0.000000 -0.070000 0.000000
110 0.000000 -0.070000 0.000000
97 0.000000 0.035000 0.000000
98 0.000000 0.035000 0.000000
105 0.000000 0.035000 0.000000
106 0.000000 0.035000 0.000000
99 0.000000 -0.035000 0.000000
100 0.000000 -0.035000 0.000000
107 0.000000 -0.035000 0.000000
108 0.000000 -0.035000 0.000000
Eye separation distance
36
19 0.100000 0.000000 0.000000
20 0.100000 0.000000 0.000000
21 0.100000 0.000000 0.000000
22 0.100000 0.000000 0.000000
23 0.100000 0.000000 0.000000
24 0.100000 0.000000 0.000000
52 -0.100000 0.000000 0.000000
53 -0.100000 0.000000 0.000000
54 -0.100000 0.000000 0.000000
55 -0.100000 0.000000 0.000000
56 -0.100000 0.000000 0.000000
57 -0.100000 0.000000 0.000000
67 0.100000 0.000000 0.000000
68 0.100000 0.000000 0.000000
69 -0.100000 0.000000 0.000000
70 -0.100000 0.000000 0.000000
71 0.100000 0.000000 0.000000
72 0.100000 0.000000 0.000000
73 -0.100000 0.000000 0.000000
74 -0.100000 0.000000 0.000000
95 0.100000 0.000000 0.000000
96 -0.100000 0.000000 0.000000
97 0.100000 0.000000 0.000000
98 -0.100000 0.000000 0.000000
99 0.100000 0.000000 0.000000
100 -0.100000 0.000000 0.000000

101 0.100000 0.000000 0.000000
102 -0.100000 0.000000 0.000000
103 0.100000 0.000000 0.000000
104 -0.100000 0.000000 0.000000
105 0.100000 0.000000 0.000000
106 -0.100000 0.000000 0.000000
107 0.100000 0.000000 0.000000
108 -0.100000 0.000000 0.000000
109 0.100000 0.000000 0.000000
110 -0.100000 0.000000 0.000000
Cheeks z
2
27 0.000000 0.000000 0.100000
60 0.000000 0.000000 0.100000
Nose z-extension
6
5 0.000000 0.000000 0.100000
75 0.000000 0.000000 0.070000
76 0.000000 0.000000 0.070000
92 0.000000 0.000000 0.050000
93 0.000000 0.000000 0.050000
94 0.000000 0.000000 0.050000
Nose vertical position
17
3 0.000000 0.100000 0.000000
4 0.000000 0.100000 0.000000
5 0.000000 0.100000 0.000000
6 0.000000 0.100000 0.000000
25 0.000000 0.100000 0.000000
26 0.000000 0.100000 0.000000
58 0.000000 0.100000 0.000000
59 0.000000 0.100000 0.000000
75 0.000000 0.100000 0.000000
76 0.000000 0.100000 0.000000
77 0.000000 0.100000 0.000000
78 0.000000 0.100000 0.000000
92 0.000000 0.100000 0.000000
93 0.000000 0.100000 0.000000
94 0.000000 0.100000 0.000000

111 0.000000 0.100000 0.000000
112 0.000000 0.100000 0.000000
Nose, pointing up
3
5 0.000000 0.050000 0.000000
75 0.000000 0.050000 0.000000
76 0.000000 0.050000 0.000000
Mouth vertical position
21
7 0.000000 0.100000 0.000000
8 0.000000 0.100000 0.000000
9 0.000000 0.100000 0.000000
31 0.000000 0.100000 0.000000
33 0.000000 0.100000 0.000000
40 0.000000 0.100000 0.000000
64 0.000000 0.100000 0.000000
66 0.000000 0.100000 0.000000
79 0.000000 0.100000 0.000000
80 0.000000 0.100000 0.000000
81 0.000000 0.100000 0.000000
82 0.000000 0.100000 0.000000
83 0.000000 0.100000 0.000000
84 0.000000 0.100000 0.000000
85 0.000000 0.100000 0.000000
86 0.000000 0.100000 0.000000
87 0.000000 0.100000 0.000000
88 0.000000 0.100000 0.000000
89 0.000000 0.100000 0.000000
90 0.000000 0.100000 0.000000
91 0.000000 0.100000 0.000000
Mouth width
14
31 0.100000 0.000000 0.000000
64 -0.100000 0.000000 0.000000
88 0.100000 0.000000 0.000000
89 -0.100000 0.000000 0.000000
79 0.050000 0.000000 0.000000
80 -0.050000 0.000000 0.000000
81 0.050000 0.000000 0.000000

82 -0.050000 0.000000 0.000000
83 0.050000 0.000000 0.000000
84 -0.050000 0.000000 0.000000
85 0.050000 0.000000 0.000000
86 -0.050000 0.000000 0.000000
90 0.050000 0.000000 0.000000
91 -0.050000 0.000000 0.000000
Eyes vertical difference
36
19 0.000000 -0.100000 0.000000
20 0.000000 -0.100000 0.000000
21 0.000000 -0.100000 0.000000
22 0.000000 -0.100000 0.000000
23 0.000000 -0.100000 0.000000
24 0.000000 -0.100000 0.000000
52 0.000000 0.100000 0.000000
53 0.000000 0.100000 0.000000
54 0.000000 0.100000 0.000000
55 0.000000 0.100000 0.000000
56 0.000000 0.100000 0.000000
57 0.000000 0.100000 0.000000
67 0.000000 -0.100000 0.000000
68 0.000000 -0.100000 0.000000
69 0.000000 0.100000 0.000000
70 0.000000 0.100000 0.000000
71 0.000000 -0.100000 0.000000
72 0.000000 -0.100000 0.000000
73 0.000000 0.100000 0.000000
74 0.000000 0.100000 0.000000
95 0.000000 -0.100000 0.000000
96 0.000000 0.100000 0.000000
97 0.000000 -0.100000 0.000000
98 0.000000 0.100000 0.000000
99 0.000000 -0.100000 0.000000
100 0.000000 0.100000 0.000000
101 0.000000 -0.100000 0.000000
102 0.000000 0.100000 0.000000
103 0.000000 -0.100000 0.000000
104 0.000000 0.100000 0.000000

105 0.000000 -0.100000 0.000000
106 0.000000 0.100000 0.000000
107 0.000000 -0.100000 0.000000
108 0.000000 0.100000 0.000000
109 0.000000 -0.100000 0.000000
110 0.000000 0.100000 0.000000
Chin width
2
30 0.100000 0.000000 0.000000
63 -0.100000 0.000000 0.000000