

REAL-TIME FACE TRACKING AND HEAD POSE ESTIMATION FOR AUGMENTED REALITY IN MOBILE PLATFORMS

K.P.U.Anuruddhi (120025B)

N.I. Nimalsiri (120427J)

K.A.Welivita (120702A)

T.R.Wickramasinghe (120706N)

Supervisor: Dr. Chandana Gamage

Degree of Bachelor of Science of Engineering (Hons)

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2017

REAL-TIME FACE TRACKING AND HEAD POSE ESTIMATION FOR AUGMENTED REALITY IN MOBILE PLATFORMS

K.P.U.Anuruddhi (120025B)

N.I. Nimalsiri (120427J)

K.A.Welivita (120702A)

T.R.Wickramasinghe (120706N)

Supervisor: Dr. Chandana Gamage

Final Year Project Report submitted in partial fulfillment of the
requirements for the degree Bachelor of Science of Engineering (Hons)
in Computer Science and Engineering

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2017

DECLARATION

“We declare that this is our own work and this report does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of our knowledge and believe it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, we hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute our report, in whole or in part in print, electronic or other medium. We retain the right to use this content in whole or part in future works (such as articles or books).”

Candidate: K.P.U.Anuruddhi (120025B)

Signature:.....

Date:

Candidate: N.I.Nimalsiri (120427J)

Signature:.....

Date:

Candidate: K.A.Welivita (120702A)

Signature:.....

Date:

Candidate: T.R.Wickramasinghe (120706N)

Signature:.....

Date:

The above candidates have carried out research for B.Sc. Final Year Research Project under my supervision.

Name of the supervisor: Dr. Chandana Gamage

Signature of the supervisor:..... Date:

Abstract

The proliferation of mobile devices such as tablets and mobile phones with enhanced processing capabilities, cameras and sensors has created an opportunity for “Augmented Reality (AR)” to gain a perfect medium to reach a wider user base and is becoming the next key technology to drive technical innovations and content creation. Mobile AR has further opened up opportunities to seamlessly improve user experience in areas of retail, e-commerce, marketing and advertising. One such trend is allowing users to virtually visualize models of facial accessories such as eyewear and earrings before actually purchasing them. This builds up a better customer engagement model which is accessible to the customers to use at their convenience through their personal mobile devices.

For developing such applications, it requires a robust, efficient and real-time face detection, tracking and head pose estimation system that is optimized for the use in mobile devices. The 3D rendering software at the backend of the application would then be able to render virtual content on top of human heads in the correct location and in the correct orientation. This report discusses the development of a real-time face tracking and head pose estimation plugin that can be integrated into Unity 3D rendering software to build up a mobile application that is capable of augmenting human faces with virtual models of eyewear.

Keywords: Face detection, Face tracking, Head pose estimation, Augmented reality

ACKNOWLEDGEMENT

We would like to thank our supervisor, Dr. Chandana Gamage for his untiring efforts in reviewing our work and providing us immense support and guidance throughout the project. And thanks are also due to Mr. Sameera Nilupul, Director, Engineering at LiveRoom (Pvt) Ltd. for providing us technical support and guidance to carry out this project.

Our gratitude is extended to Eng. Nalin Karunasinghe for giving us valuable advice during project evaluations and encouraging us to achieve more and Dr. Charith Chitraranjan for his guidance on preparing the reports and coordinating all the work with respect to the final year project module. We also like to thank Dr. Chathura De Silva, Head of the Department of Computer Science and Engineering for providing us knowledge on Computer Vision field.

The facilities provided by the Department of Computer Science and Engineering, University of Moratuwa for conducting this research is greatly appreciated.

TABLE OF CONTENTS

DECLARATION	i
Abstract	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	x
LIST OF ABBREVIATIONS	xi
LIST OF APPENDICES	xiii
1. INTRODUCTION	1
1.1. Background	1
1.2 Problem statement	1
1.3 Motivation	22
1.4. Research objectives	2
1.5. Requirements specification	3
2. LITERATURE REVIEW	6
2.1. Introduction	6
2.2 Face detection and feature extraction	7
2.3 Face tracking	22
2.4. Head Pose Estimation	36
2.5. Related APIs and SDKs	60
2.6. Related Commercial Applications	65
2.7. Results	67
2.8. Discussion	67
2.9. Conclusion	68
3. RESEARCH METHODOLOGY	69
4. DESIGN DOCUMENT	71
4.1. Architectural representation	71
4.2. Architectural Goals And Constraints	73
4.3. Use-Case Realization	73
4.5. Process View	74

4.6. Physical view	75
5. EVALUATION OF FACE DETECTION AND FACE TRACKING ALGORITHMS	76
5.1 Evaluation Methodology	76
5.2 Evaluation criteria	77
5.2.1 Evaluation criteria for face detection algorithms	78
5.2.2 Evaluation criteria for face tracking algorithms	78
5.3 Evaluation and results of face detection algorithms	80
5.4 Evaluation and results of face tracking algorithms	80
5.4.1 KLT algorithm	81
5.4.2 AAM algorithm	82
5.4.3 CLM algorithm	84
5.5 Discussion	86
5.5.1 Face detection algorithms	86
5.5.2 Face tracking algorithms	86
5.6 Selection of the optimum approaches	88
6. IMPLEMENTATION	89
6.1 Tools and libraries used	89
6.2 Face Detection	91
6.3 Facial Feature Detection and Tracking	92
6.3.1. Facial feature extraction using Constrained Local Model	93
6.4 Head Pose Estimation	96
6.5.1 Histogram Equalization	97
6.5.2 Divergence Check	99
6.6 Graphical user interface of the mobile application	100
6.7 Connection to the server and back end	102
7. EVALUATION AND RESULTS	104
8. DISCUSSION AND CONCLUSION	107
REFERENCES	108
APPENDIX - A: SUMMARY OF FACE DETECTION TECHNIQUES	118
APPENDIX - B: SUMMARY OF FACE TRACKING TECHNIQUES	120
APPENDIX - C: SUMMARY OF HEAD POSE ESTIMATION TECHNIQUES	121

LIST OF FIGURES

		Page
Figure 1.1	Three degrees of rotational freedom termed Pitch, Yaw and Roll	3
Figure 2.1	Example Haar-like features shown relative to the detection window	8
Figure 2.2	Examples where the Haar-like features can be applied to differentiate certain unique features in the human face	8
Figure 2.3	Representation of the face space	10
Figure 2.4	Results captured by the SVM face detection system, detecting non-frontal faces	11
Figure 2.5	The facial model suggested by R. Cipolla	17
Figure 2.6	Interesting points found by use of spatial filtering with second derivative Gaussian filter	17
Figure 2.7	Result of applying grayscale erosion followed by an extremum sharpening operation	18
Figure 2.8	y-projection of a facial image	19
Figure 2.9	A face with correctly positioned landmarks by improved ASM model fitting	25
Figure 2.10	Average rate of convergence for Generic AAMs computed using original and refitted labels	27
Figure 2.11	Average rate of convergence for Generic AAMs using the SIC and PO algorithms using the refit labels	28
Figure 2.12	Histograms of errors after search from displaced positions for face data	29
Figure 2.13	The MSE of neighboring warped frames of a video sequence	30
Figure 2.14	Fitting results with zoom in facial area using SICOV	30
Figure 2.15	The number of iterations in fitting each frame of a video sequence	31
Figure 2.16	CLM Search	32
Figure 2.17	Experimental results obtained for generic face model for	33

	images	
Figure 2.18	Number of iterations for POSIT as a function of distance to camera	37
Figure 2.19	Examples from the training sets for the models	39
Figure 2.20	Comparison of angle derived from AAM tracking with actual angle	40
Figure 2.21	Overview of patch based pose estimation	41
Figure 2.22	System block diagram	42
Figure 2.23	The facial normal	44
Figure 2.24	Lengths between facial features	44
Figure 2.25	Examples of near-frame rate face detection, tracking and pose estimation 43 using a multiview composite SVM	47
Figure 2.26	An example labelled head image set	48
Figure 2.27	An example labelled head image set	50
Figure 2.28	Multi-view face detection and recognition	52
Figure 2.29	An overview of the proposed Head Pose Estimation System	53
Figure 2.30	Flow chart of the pose estimation system	53
Figure 2.31	Manifold mapping-Training Estimation and Recognition and Pose	56
Figure 2.32	Architecture of convolutional network used for training	57
Figure 2.33	First stage pose estimation	58
Figure 2.34	Second stage refinement	59
Figure 2.35	Example facial landmarks identified and tracked by Google Mobile Vision API	61
Figure 2.36	Pose angles determined by Google Mobile Vision API	61
Figure 2.37	Pose angles determined by Google Mobile Vision API	63
Figure 4.1	Architectural Representation	72
Figure 4.2	Setup in Unity3D	72
Figure 4.3	Use-case realization	74

Figure 4.4	SpecuLAR Sequence Diagram	75
Figure 4.5	SpecuLAR Deployment Diagram	75
Figure 5.1	Representation of annotated ground truth points and corresponding estimated points in a facial image.	79
Figure 5.2	Comparison between point to point RMS errors of the two approaches tracking facial features (eyes, nose tip and center of mouth) vs. tracking SIFT features in the facial region using the KLT point tracker.	81
Figure 5.3	19, 15 and 7 facial landmark points used for training the AAM.	82
Figure 5.4	The shape model and appearance model of the AAM obtained by using 19 facial landmark points.	82
Figure 5.5	Comparison between point to point RMS errors for AAM fitting using 10 iterations and point to point RMS errors for AAM fitting with optical flow initialization using 10 iterations.	83
Figure 5.6	Comparison between point to point RMS errors for AAM fitting using 5 iterations and point to point RMS errors for AAM fitting with optical flow initialization using 5 iterations.	84
Figure 5.7	Comparison between RMS point to point errors for AAM fitting and AAM fitting with optical flow initialization, using 10 and 5 iterations each for video no. 2.	84
Figure 5.8	Comparison of point to point RMS error values of CLM based face tracking with 5 iterations in a 16×16 search area vs. KLT tracking with CLM initialization.	85
Figure 6.1	Face detection and facial feature detection outputs from Accord.NET framework integrated with Unity.	90
Figure 6.2	Face detection outputs from EmguCV	91
Figure 6.3	Process of facial feature extraction	93
Figure 6.4	Procrustes distance based alignment	94
Figure 6.5	First four modes of variation	94
Figure 6.6	A linear image patch for left outer eye corner	95
Figure 6.7	The facial model	96

Figure 6.8	Global Histogram Equalization vs Contrast Limited Adaptive Histogram Equalization [93].	98
Figure 6.9	The point is inside the polygon	100
Figure 6.10	The point is outside the polygon	100
Figure 6.11	Assets of the 3D model	101
Figure 6.12	Pivot position of an imported 3D model	102
Figure 6.13	Correct pivot position after adding the 3D model to dummy parent object	102
Figure 6.14	User interface for uploading 3D Models to the server	103
Figure 7.1	Sample video frames taken from the database.	104
Figure 7.2	Point to point RMS errors between the ground truth and the estimated facial feature points	105

LIST OF TABLES

		Page
Table 3.1	Research methodology	70
Table 5.1	Test video suit specifications	77
Table 5.2	Summary of results for face detection algorithms	80
Table 5.3	Average FPS values for the 2 described approaches used with the KLT tracker	81
Table 5.4	FPS for variations of the AAM algorithm	83
Table 5.5	FPS for CLM algorithm and CLM initialization + KLT tracking with 5 iterations and 16 x 16 search area	85

LIST OF ABBREVIATIONS

Abbreviation	Description
AAM	Active Appearance Model
API	Application program interface
AR	Augmented Reality
ASM	Active Shape Model
AVAM	Adaptive View-Based Appearance Model
CLM	Constrained Local Models
DRMF	Discriminative Response Map Fitting
GWN	Gabor Wavelet Network
HSV	Hue-Saturation-Value
KLT	Kanade-Lucas-Tomasi
MSE	Mean Squared Error
NCC	Normalized Color Coordinate
PCA	Principal Component Analysis
POSIT	Pose from Orthography and Scaling with Iterations
PO	Project Out
RANSAC	Random Sample Consensus
RGB	Red-Green-Blue
RMS	Root mean square
RVM	Relevance Vector Machine
SDK	Software development kit
SFM	Structure from motion
SIC	Simultaneous Inverse Compositional
SICOV	SIC for Video
SMAT	Simultaneous Modelling and Tracking
SRM	Structural Risk Minimisation

SVC	Support Vector Classification
SVM	Support Vector Machine
SVR	Support Vector Regression

LIST OF APPENDICES

	Page
Appendix - A Summary of face detection techniques	118
Appendix - B Summary of face tracking techniques	120
Appendix - C Summary of head pose estimation techniques	121

1. INTRODUCTION

1.1. Background

Augmented Reality is a technology that superimposes computer-generated images and graphics onto real world environments. This enhances the user's perception of reality by combining real and virtual elements. From early 2000 onwards, with the proliferation of mobile devices such as tablets and mobile phones, AR has gained a perfect medium to reach people and is becoming the next big thing in technology. Increasingly, companies are using AR technology to reach out to customers to market their products by allowing users to virtualize product models of jewellery, eyewear etc. on top of themselves. This we define as "self augmentation". Products like furniture, bathware etc. can be virtualized against real world environments which we call "environment augmentation". For both self and environment augmentation, the 3D rendering software behind the AR application needs to have some method to get the correct location and the correct orientation according to which content should be rendered. In "environment augmentation" this can be achieved with very high accuracy by using marker based approaches. But in "self augmentation" scenarios, we need some efficient algorithms to robustly detect and track the human face and determine the pose of the head in real-time, in order to augment content on top of human heads. For this we need to examine efficient face detection, tracking and head pose estimation algorithms that can work efficiently meeting the real-time requirements.

Mobile AR has further opened up opportunities to seamlessly improve user experience in areas of retail, e-commerce, marketing and advertising. LiveRoom [1] is one such application that allows users to augment the environment and try out various products before actually purchasing them. This builds up a better customer engagement model which is accessible to the customers 24x7 right through their personal mobile devices. An emerging trend in this field is allowing users to augment themselves and virtually try on accessories like eyewear and earrings.

1.2. Problem statement

LiveRoom currently supports environment augmentation using a marker based approach. It renders virtual 3D objects on top of the real world video feed captured using the back facing camera of a mobile device. But it does not support allowing users to augment themselves using the front facing camera because of the impracticability in placing markers on a person's face. Hence it is required to have an approach that would allow tracking the face and estimating the head pose so that virtual accessories can be realistically placed on top of the human face.

1.3. Motivation

Solving the above problem would allow a diverse range of enhancements to be integrated into the existing AR experience provided by the LiveRoom application. This would make a remarkable impact on the e-commerce business sector by bringing in novel means of interaction with the customers.

Analyzing all the possible approaches on face tracking and head pose estimation that exist in the literature would not be sufficient to come up with the most appropriate solution to the problem at hand. That is because most of those solutions are either generic approaches or approaches specifically designed for applications different from ours. Therefore, this would involve a lot of coding for actually implementing the algorithms which would enhance our knowledge and skills on areas of Computer Vision, Machine Learning, Image Processing and Mathematics.

In order to bring in a seamless experience for the users, the application needs to have a real-time face tracking and head pose estimation algorithm with a considerable level of accuracy. As stated before, a lot of research has already been done in general to address the challenges in this field. Most of the algorithms proposed by these researches perform reasonably well on PC platforms with relatively high processing power and large memory but lag when run on mobile platforms. Therefore, we need a more accurate and concise algorithm for meeting the requirements of the stated problem.

1.4. Research objectives

Our research objectives include designing a fast and robust algorithm optimized for the front facing camera of a mobile device that is capable of accurately tracking a face and estimating the head pose (i.e. forward/backward, up/down, left/right motion and rotational movements along the 3 axes termed pitch, yaw and roll as depicted in Figure 1.1) and developing a plugin for the Unity game engine that performs the aforementioned task so that it can be easily integrated into the application of interest.

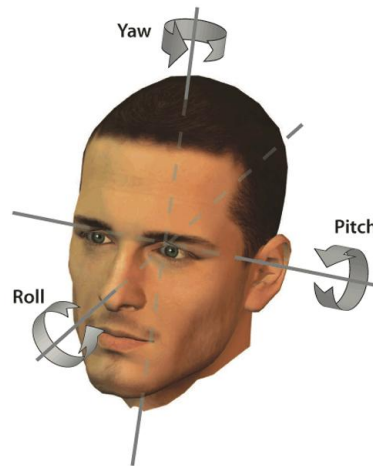


Figure 1.1: Three degrees of rotational freedom termed Pitch, Yaw and Roll

1.5. Requirements specification

We propose an AR based demo mobile application named ‘SpecuLAR’ which allows users to try on virtual models of eyewear by augmenting human faces. This should have the capability of tracking translational movement of the head along the three axes x,y and z and detecting the rotational movement of the head along these three axes. Hence it should have the ability to render virtual models of eyewear on human faces in the correct location and orientation, giving the users seamless, real-time experience. Eyewear dealers can use this application as a way of allowing their customers to visualize latest models of eyewear without having to visit their showrooms and purchase them through mobile phones.

1.5.1 Functionality requirements

The functionality requirements of the proposed application includes,

- The ability to view the augmented model of the eyewear or accessory on oneself when viewed from different perspectives.
- The ability to select a desired model.
- The ability to update the models.
- The ability to make an order.
- The ability to take a snapshot or record a video and share it on social networks.
- The ability to switch back to rear cam when necessary.

1.5.2 Usability requirements

Usability is the extent to which the product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. SpecuLAR application should have an intuitive design and user interfaces which are aesthetically pleasing to the user. There should be easy navigation in the application, which would make the users able to easily find the essential features quickly. While the users are using the application, they should not have trouble in finding exactly what they are looking for. The application should work in different mobile platforms having different specifications. Ease of learning should be a key usability requirement in the application as it would be used by people having different levels of technical abilities. The training time taken for a normal user should be less than 5 minutes and a user should be able to become productive at using a certain operation of the application within less than 1 minute.

1.5.3. Reliability requirements

Reliability is usually defined as the probability that a product will operate without failure for a specified number of uses for a specified period of time. The proposed application should operate without failure for any number of uses at any time. Moreover, the product needs to exceed customer expectations. It should be more reliable than its competitors.

Reliability can be stated as an average period of operation before failure. Any piece of hardware will eventually wear out and fail and as such hardware product

reliability can be stated and measured in terms of time to failure or time between failures. Hence SpecuLAR requires high degree of reliability. The level of reliability can be dependant on the type of mobile platform as well.

1.5.4 Performance requirements

The performance requirements of the proposed application includes,

- Response Time
 - Quick response time is very important in a real-time application. The head pose has to be calculated real-time and the speed of interaction should be high.
- Resource utilization
 - Memory utilization by the application should be low due to the memory limitations in mobile devices.

1.5.5. Supportability requirements

Supportability refers to the application's ability to be easily modified or maintained to accommodate typical usage or change scenarios. The proposed application should support its functionality in different kinds of environments like indoor, outdoor etc. and under different lighting conditions. The application should also support different device and camera specifications of mobile phones.

1.5.6. Design constraints

The major design constraints for a real-time application that runs on a mobile platform are the low processing power and memory limitations in mobile devices. Sometimes the quality of the video stream captured by the front facing camera of mobile phones is lower when compared to the back facing camera. These video streams may also consist of different types of noise and they can be taken under different illumination conditions with mobile phones having different device and camera specifications. The application should have the capability to handle all these design constraints.

2. LITERATURE REVIEW

2.1. Introduction

In order to come up with an algorithm optimized for tracking the face and estimating the head pose on a mobile platform, it is required to have knowledge on the existing algorithms in the literature which perform these tasks. Hence a comprehensive review on the algorithms, APIs and SDKs that exist for achieving the tasks of face detection, tracking and head pose estimation was carried out and is presented in this chapter.

Face detection refers to locating all the faces inside a particular image, regardless of their position, facial gestures and variations in scale and orientation. This is used to identify human faces in digital images. Face tracking is used to calculate the displacement of the detected face between different frames.

Head pose estimation refers to inferring the orientation of the head with respect to three degrees of freedom indicated by pitch, yaw and roll. Given an image of a person, an ideal head pose estimator should have the ability to analyse the orientation of that person's head accurately with respect to the aforementioned angles. This capability can then be extended for video streams for which the head pose estimator should have the ability to estimate the head pose in real-time. An ideal head pose estimator should also be robust against the speed of motion of the person and variations in illumination, skin colour or background. Infact, many computer vision applications require a reliable, fast, and accurate face trackers and head pose estimators. We intend to look at such approaches in the following sections.

The existing approaches for face detection and facial feature extraction are examined first. Next, face tracking algorithms which are categorized under model based, feature based and appearance based depending on the technique these algorithms follow to track the face, are discussed. Then several head pose estimation algorithms which also are categorized under model based, feature based, appearance based and hybrid approaches are examined. Thereafter some APIs and SDKs which are available to achieve the same target are discussed. And finally several commercial

level, successful AR applications that have utilized these algorithms to provide seamless AR experience for the users are presented.

2.2 Face detection and feature extraction

Many algorithms have been developed for detecting faces in digital images. The most widely used method for real-time face detection is the Viola-Jones algorithm. This lies as the foundation for most of the other algorithms. Face detection can be categorized into two different categories as feature based and training based approaches.

Feature extraction refers to identifying specific features in a feature image. This concept goes hand in hand with face detection but still has the potential to give significant results independently as well. The following section describes some prominent algorithms that are used in the area of face detection and facial feature extraction.

2.2.1 Feature based face detection

2.2.1.1 Viola Jones object detection framework

Paul Viola and Michael J. Jones introduced a face detection framework that has the capability of processing images rapidly [2]. This is a machine learning approach for detecting faces given the upfront frontal facial images and is based on a cascade of weak classifiers that can detect separate Haar-like features identified in a facial image.

In this approach, a set of classifiers is trained using supervised learning to detect examples of a particular class, in this case, the class of faces. By inputting a set of facial and non-facial images, the algorithm trains a set of weak classifiers to identify a new facial image. Figure 2.1 shows some examples of Haar-like features shown relative to the enclosing detection window, which are used by the Viola-Jones face detection algorithm to identify upright, frontal faces.

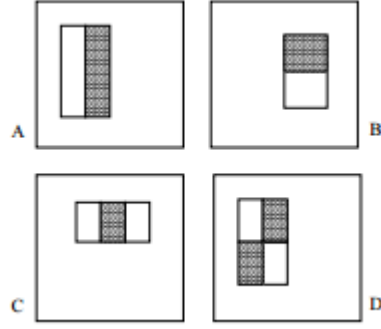


Figure 2.1: Example Haar-like features shown relative to the detection window [2]

Figure 2.1 shows some examples where these Haar-like features can be applied to differentiate certain unique features in the human face. The first feature measures the difference in intensity between the region of the eyes and the region across the upper cheeks. The second feature measures the intensity difference between the bridge of the nose and the regions either side of it.

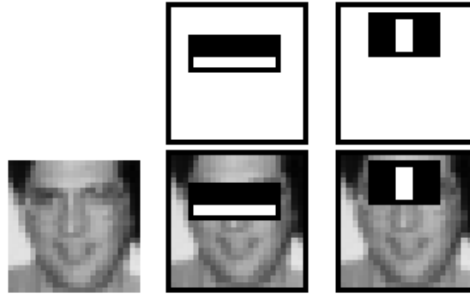


Figure 2.2: Examples where the Haar-like features can be applied to differentiate certain unique features in the human face [2]

Given the set of Haar-like features and a facial and non-facial training image data set, any machine learning algorithm can be used to learn the classification function and train a set of weak classifiers to identify faces and non-faces based on the Haar-like features. Within an enclosing object detection window of base resolution 24x24, there are about 160,000+ Haar-like features and using which we can form 160,000+ weak classifiers [2]. A variant of the Adaboost algorithm [3] is used to select a small set of features out of these and use them in a cascade of weak classifiers for identifying unseen images.

Then in the phase of object detection, a detection window starting at base resolution

24x24 is scanned through the new image. It keeps on iteratively increasing the window resolution while it finds a location of a face based on the value each of the Haar-like feature yields in each of the window scans. The value of any given feature is calculated as the sum of pixel values under the black rectangle, subtracted from the sum of pixel values under the white rectangle.

Experiments conducted by Viola and Jones have yielded a detection rate of 95% with a false positive rate of 1 in 14084 for a frontal face classifier constructed from 200 features. And in terms of computation time, this classifier required only 0.7 seconds to scan a 384 by 288 pixel image which is significantly faster than any other face detection algorithm [2]. This makes it possible to use the algorithm even with mobile platforms having low computational power. However, this presents a trade off between the detection accuracy and the computation time because the most straightforward method for improving the detection rate is introducing more features which then increases the computation time. And another disadvantage of using this algorithm is, it only has the ability to detect frontal faces and it can hardly detect even faces rotated 45^0 around the vertical and horizontal axes. As such, if we use this algorithm in mobile AR applications, it might enforce the constraint that the user will always have to place his face directly facing the camera in order to get his face captured by the application.

2.2.1.2 Face detection using Eigenfaces

M. Turk and A. Pentland proposed a face detection mechanism using eigenfaces as described in [4]. The motivation behind Eigenfaces is that the previous work had ignored the question of which features are important for classification, and which are not. Eigenfaces tries to answer this by using Principle Component Analysis (PCA) of the images of the faces. This analysis reduces the dimensionality of the training set by leaving features that are only critical. This work has used frame differencing to track motion against a static background. The filtered image is thresholded to produce a number of motion blobs. Furthermore, this has used few heuristics such as the small blob of the large blob is a face and the face must all move contiguously to determine the location of the head. The size of this blob can be used to estimate the

size of the face and fit them to the face space which is represented by Figure 2.3. The size could also be used in a multi scale eigenfaces approach. If there are a number of blobs to select from, the face can be located by examining a fixed size subregion and determining the 'faceness' of that region. However, this type of brute force method can be quite time consuming and may not be practical for any real-time application. The authors present a derivation of this algorithm to suggest how faceness might be computed in a less computationally expensive way, by calculating some of the fixed terms ahead of time. The following steps summarizes the face detection algorithm.

Given an unknown image Γ ,

Step 1: Compute $\Phi = \Gamma - \Psi$; Ψ is the average face vector

Step 2: Compute $\Phi^{\wedge} = \sum_j^k w_j u_j$ ($w_j = u_j^T \Phi$)

Step 3: Compute $e_d = \|\Phi - \Phi^{\wedge}\|$

Step 4: If $e_d < T_d$, then Γ is a face. The distance e_d is called distance from face space and T_d is the threshold distance from face space.

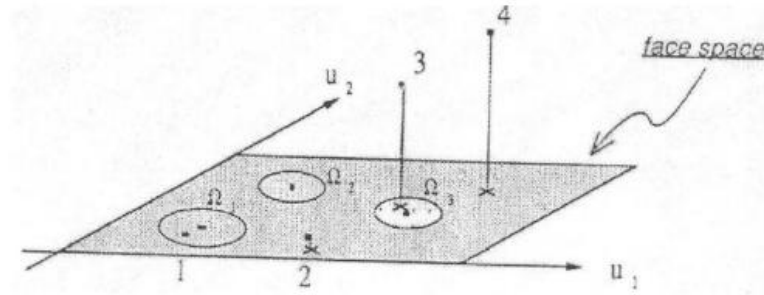


Figure 2.3: Representation of the face space [4]

With this approach, performance degrades with changes in light and changes in size of the head. Performance also decreases with changes in the orientation but not as fast as with changes in scale. Hence, it would be difficult to use the algorithm directly in real world mobile applications.

2.2.1.3 Skin based face detection

Various research had been carried out in using human skin as a mode of detecting a

human face and its enclosing features. One such approach was put forward in [5] where a method was incorporated with skin detection along with grayscale morphology. A geometrical facial model is used in roughly detecting the facial region and extracting its features like eyes, mouth, nostrils and eyebrows. This approach has been able to yield successful results with near frontal facial images under proper lighting conditions. Authors state that their method had enough performance to be run on low computational power devices like mobile platforms. Butn still it lacks robustness to perform accurately in varying lighting conditions and in the presence of face occlusions.

2.2.2 Training based face detection

2.2.2.1 Neural Networks for face detection

Neural networks based face detection as proposed by Henry A. Rowley, Shumeet Baluja, and Takeo Kanade [6] is a method for recognizing faces in grayscale images utilizing neural networks. They have designed an upright frontal face identification framework. This framework utilizes a retinally associated neural network system to inspect small windows of a picture and choose whether each window contains a face. The framework additionally mediates between multiple networks to enhance execution over a solitary network. This paper introduces a direct strategy for training positive face examples. To gather negative faces, they utilized a bootstrap calculation, which included false discoveries into the training set while going through the training stage. This would then take out the troublesome work of manually selecting non face examples. This proposed system works as two stages. First it applies a set of neural network based filters to an image, and then uses an arbitrator to combine the outputs. The arbitrator then merges detections from individual filters and eliminates overlapping detections.

Stage 1: The principal component of the framework is a filter that gets a 20×20 pixel area of the image as input, and produces a result of 1 to -1, meaning the presence or absence of a face in that area. This filter is applied at every location in the image. The image is then repeatedly diminished in size and filter is added to detect faces bigger than the window size, which is called subsampling.

First, a preprocessing step is applied to a window of the image to correct lighting effects. This would equalize the intensity values across the window. Then the intensity values would be mapped non-linearly to increase the scope of intensities in the window which is called histogram equalization. This preprocessed window is then applied to the neural network which has retinal connections to its input layer. There are three types of hidden units: 4 which look at 10×10 pixel subregions, 16 which look at 5×5 pixel subregions, and 6 which look at overlapping 20×5 pixel horizontal stripes of pixels. For example, the horizontal stripes are used to identify features such as mouths or pairs of eyes, while the square stripes are used to detect features such as individual eyes, the nose, or corners of the mouth.

Stage 2: The system has used two methods to improve the reliability of the detector: merging overlapping detections from a single network and arbitrating among multiple networks.

The algorithm proposed by Henry A. Rowley, Shumeet Baluja and Takeo Kanade can detect between 77.9% and 90.3% of faces in a set of 130 test images, with an acceptable number of false detections [6]. Depending on the application, the system can be made more or less conservative by varying the arbitration heuristics or the thresholds used.

2.2.2.2 Support Vector Machines

E. Osuna et al. proposed a method of training a Support Vector Machine (SVM), which is a machine learning technique developed by V. Vapnik et al. at Bell Labs, for detecting faces [7]. The hyper plane or the decision surface of the SVM in this case is found by solving a linearly constrained quadratic programming problem. Because of its quadratic nature, this is a challenging optimization due to the fact that memory requirements can grow with the square of the number of data points.

But here they present a decomposition algorithm that can be used to train SVMs over very large datasets guaranteeing optimality. This decomposition algorithm solves subproblems iteratively while evaluating the optimality conditions. This helps in generating improved iterative values and establishing the stopping criteria for the

algorithm.

At the detection stage, the SVM algorithm exhaustively scans an image for faces at many possible scales by dividing the original image into overlapping sub images. These sub images are then classified using the SVM as a face or a non-face.

The feasibility of this approach has been established by them on a face detection problem which involves a dataset of 50,000. Their implementation has the ability to deal with about 2,500 support vectors on a machine with 128MB of RAM. In order to test the runtime system, two sets of images have been used by them, one comprising of 313 high-quality images with one face per image and the other comprising of 23 images of mixed quality with a total of 155 faces. These two sets have been tested using their system and has reached a detection rate of 97.1% in the first test set and a detection rate of 74.2% in the second test set. In order explain the different detection rates obtained for the two sets of images, it is important to state that the first set involved 4,669,960 pattern windows, while the second set involved 5,383,682 pattern windows [7].

Unlike Viola Jones face detection algorithm, the SVM approach is able to capture non-frontal facial images up to a small degree as indicated in Figure 2.4, but can miss partially occluded faces. The detection accuracy of partially occluded faces can be improved by adding in a lot of training images with partially occluded facial images. Their system also has the ability to deal with some degree of rotation of the image plane because the database they have used for training has a number of virtual faces obtained by rotating some face examples up to 10^0 [7]. When relating this to the context of mobile AR applications dealing with video streams, this can be expected to perform better than the Viola Jones algorithm in detecting non-frontal faces and faces with small degrees of rotation, but its efficiency in detecting faces in real time should further be tested.



Figure 2.4: Results captured by SVM face detection system, detecting non-frontal faces [7]

2.2.3 Facial feature extraction

Facial feature extraction in general deals with identifying and locating interesting features in a facial image which mostly include physical features like eyes, eyebrows, nose and mouth. Yet also it is still possible to have other interesting facial features that do not correspond to any sensible human organ.

Locations of some features might help in narrowing down the search for other features. For example the location of eyes can tell a relative estimation of the area where the eyebrows might be and also the location of the mouth can tell where the nose might be [8]. The same way the initial search for the facial features can be drastically improved by means of a face detection algorithm that provides a coarse estimation on the region the human head might exist in an image [8].

A large number of research attempts have been carried out so far in this field that, it is quite difficult to go through each and every one of them in detail. However in general, feature based facial feature extraction methods can be put under two broad categories as low level methods and template based methods. Some important facial feature detection methods that belongs to each of these categories are discussed below [5].

2.2.3.1 Low level methods

Low level methods make use of traits such as the color or the grey levels in the image in finding some points of interest. Once these points of interest are found, these can be interpreted as facial features, based on some additional knowledge on the structure and positioning of the face and its physical organs.

2.2.3.1.1 Color based feature extraction

Skin color analysis is the basic approach that can be taken under color based facial feature extraction methods and this method is a good candidate for systems that require real time performance [5] making them suitable for mobile AR applications of our interest. Once the skin area is detected it would be possible to figure out face-like regions and proceed with specific facial feature mining.

To start with, the images need to be converted into a color space that enhances the human skin present in an image. There are several image spaces like Red-Green-Blue (RGB), Normalized Color Coordinate (NCC), Hue-Saturation-Value (HSV) and YCrCb and out of them, as Soriano et al. [9] have found, NCC is the most suitable one for the purpose. RGB color coordinates can be converted to NCC color coordinates by equations 2.1, 2.2 and 2.3. It should be noted that only two chromaticities are needed since the other can always be derived from the given two.

$$I = R + G + B \rightarrow (2.1)$$

$$r = R/I \rightarrow (2.2)$$

$$b = B/I \rightarrow (2.3)$$

Once the image is converted to the NCC color space, a histogram is drawn based on the chromaticities r and b . The skin color found in the image will occupy a small cluster in that histogram and the small cluster can be generalized and identified by training with images containing skin areas in various lighting conditions. Once the cluster is identified, given r and b values, it can be checked if the point specified lies inside the learnt cluster and if so, it can be considered a skin pixel. This skin detection algorithm is presented in detail in the paper by Soriano et al. [9]. It was

observed in their paper, that the authors have focused more in achieving better accuracy and robustness than performance. Hence there is not enough evidence to come to any conclusion regarding the suitability of this method to run on mobile platforms.

For facial feature extraction, R.L. Hsu et al. proposed a method that involves constructing feature maps for eyes, mouth and face boundary [10]. For instance, to detect the eyes, two separate eye maps are built where one corresponds to the chrominance component and the other corresponds to the luminance component. The chroma map is based on the general observation that Cr and Cb values in an YCrCb image containing eyes tend to have high Cb values and low Cr values around the eyes. Similarly, the luma map makes use of the observable bright and dark pixels around the eye regions [10].

Authors claim that this method of feature extraction gives around 80% average accuracy but the average computational time ranges from around 10 - 40 seconds when run on a 860 MHz CPU. This makes it quite improbable for it to give real-time results when run on general mobile devices without high computational power.

2.2.3.1.2 Edge detection based feature extraction

Edge detection is a very commonly used technique in detecting features in image processing and computer vision fields. The situation is the same with facial feature extraction. There are numerous approaches put forward by various researches in handling this problem of edge detection. As Hjelmås and Low state in their survey [11], the Sobel operator is the most common edge detection approach used in facial feature extraction methods. Some other methods like Marr-Hildreth edge operator [12] and a variety of first and second derivatives of Gaussians have also been utilized in some other approaches [13].

K.C. Yow and R. Cipolla in their paper [13] talks about a method of detecting facial features using a simple facial model as shown in Figure 2.5. The model symbolizes facial features specifically eyes, nose and mouth.



Figure 2.5: The facial model suggested by R. Cipolla [13]

As such, for a given image to detect those facial features, they propose a way of first finding a list of interesting points in the image using spatial filtering. That is by smoothing the image and then filtering it with a second derivative Gaussian filter elongated at an aspect ratio of 3:1. Local maxima in the resulting image as shown in Figure 2.6 would mark the presence of such interesting points that agree to some extent with the aforementioned facial feature model. Next, this list of interesting points is analyzed and processed further by examining around their neighborhood and trying to link the edges based on their proximity. Finally, the edges that match the facial model are chosen as the specific facial features.

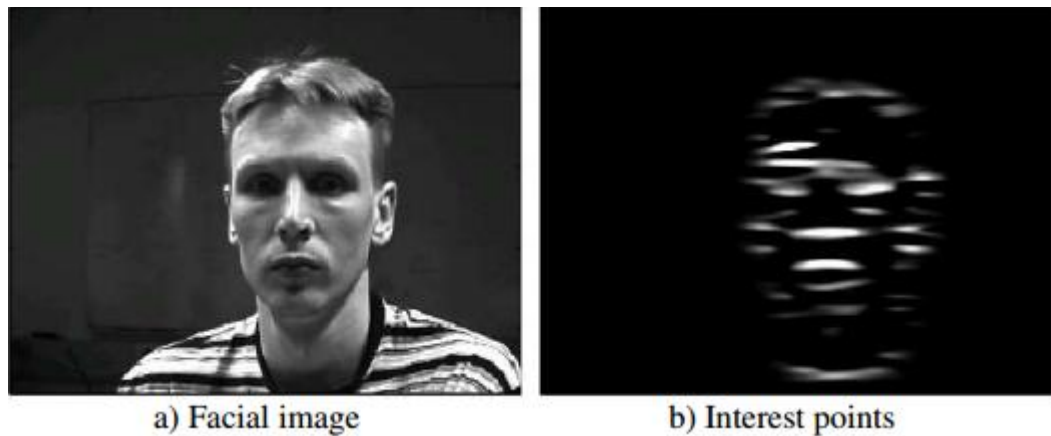


Figure 2.6: Interesting points found by use of spatial filtering with second derivative Gaussian filter [5]

It could be observed that the focus of the authors had been to have their method as universal as possible and their future work is also targeted at making the method more scale invariant and more robust. Hence there is not enough information to come to a conclusion on the suitability of this algorithm to run in mobile devices and give real-time results.

2.2.3.1.3 Projections

Owing to the various inherent positioning and color properties of our human parts, there is an observable difference in their brightness compared to their surrounding areas in a normal grayscale facial image. These traits can be used to identify each of those features by means of various projection methods. In general a projection method would project the image into some plot with respect to some property or quality in the image and would observe and analyze the plot to come up with interesting information. Sobottka and Pitas conducted one such research [14] on using grayscale intensities on an enhanced facial image to detect facial features.

Their method includes a preprocessing step where the dark regions of the detected facial area in an image are enhanced first by applying a grayscale erosion followed by an extremum sharpening operation. The results of enhancement on two different facial images are shown in Figure 2.7.



Figure 2.7: Result of applying grayscale erosion followed by an extremum sharpening operation [14]

Then the x-projection and the y-projection of the grey levels in the image are taken

by determining the mean grey level of each row or column in the detected facial region. The y-projection (taking means of the rows) would show significant minima for hair, eyes, nose, mouth and chin as shown in the Figure 2.8.

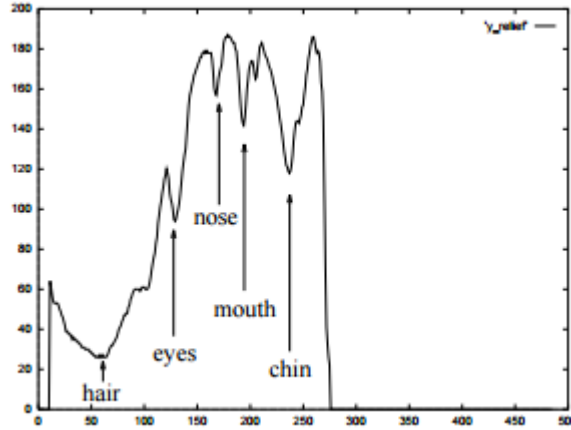


Figure 2.8: y-projection of a facial image [14]

With that, a rough list of candidate features can be identified and the exact feature identification and consolidation is done by analyzing the x-projection along the interested areas.

This method has shown a detection rate varying between 74% and 97% for different facial features and will show poor performance with images of people having different hairstyles and beards. This might become an issue if we are to employ this method in a mobile application since the user base of the application cannot be restricted. Performance wise the algorithm can be believed to consume relatively less computational power since feature extraction only involves a finite set of simple mathematical calculations like finding local minimums and comparisons. Still it is difficult to arrive at a strong conclusion about its suitability for mobile platforms without actually testing it on one.

2.2.3.1.4 Gabor filters

Gabor filter is a widely used linear filter in the field of image processing that is used specifically in edge detection applications. Image analysis with Gabor filters is considered to be quite similar to the perception in the human visual system. Sirohey

and Rosenfeld in their research on eye detection in face images [15] have used Gabor filters as one of the approaches.

The eye detection filter they have used was constructed of oriented Gabor wavelets that form an approximation to the human eye. The most important feature of this filter that makes it suitable for this purpose according to them is that they have a property of energy localization in both space and frequency domains. Their results have given 80% detection rate with Aberdeen database and a 95% detection rate on a certain video frame sequence [15]. Even though the accuracy of this method gives positive results, the computational power it requires makes it problematic to be used in real time applications and mobile platforms.

2.2.3.2 Template based methods

Template based methods try to solve the problem of facial feature extraction by means of a search looking for a known feature that adheres to a known template. Correlation and Hough transform are examples of such techniques.

2.2.3.2.1 Correlation

Correlation technique is considered as one of the simplest ways of feature extraction. This involves finding the minimum error between a certain template and a patch in the image. The simplest process in achieving this is to convolute the template with all possible positions in the image and find the locations that gives the maximum value.

Brunelli and Poggio in their research on face recognition [16] have used correlation as a step in detecting the facial features. First they normalize the given image as a preprocessing step. All the pre-registered facial images are kept in a database along with a set of four masks representing the eyes, nose, mouth and additionally the facial region from below the eyebrows. Given an unclassified image, the image is matched with all the masks in the database using the correlation technique and a vector containing a list of cumulative scores is returned. The face is classified as the one giving the highest cumulative score.

They also have found out that the results from this template matching correlation

technique gives better accuracy than low level feature based methods. One issue with this method is that variations in illumination and object orientation in the image can cause inaccurate results and hence would always require a preprocessing step on the images before applying this technique. Another downside of this classical method is the high computation time required since images usually contain a huge number of pixels and comparing each of them consumes time proportionally. Still various techniques like image pyramids and cascaded classifiers can significantly improve its performance. Though it might be hard to achieve real time results with this method, it is a strong candidate for one time feature extraction needs in mobile devices.

2.2.3.2.2 Hough transform

Hough transform is a well-known feature extraction procedure widely used in the fields of image analysis, computer vision and digital image processing. The classical Hough transform technique was focused on identifying lines in images. Later on, various extensions of the same technique came to existence that have the capability of identifying positions of arbitrary shapes like circles or ellipses.

G. Chow and X. Li in their research on automatic facial feature detection [17] have suggested a way of extracting the shapes and locations of eyes and mouth from a relatively unposed head and shoulder image. Their approach consists of 3 modules of which one is a context module that looks for possible locations of faces and the other two are eye and mouth modules that target at locating those specific features of interest. The eyes and mouth modules are more complex than the first and are based on a combined approach of the Hough transform and deformable template techniques. The eye module functions basically by attempting to locate the irises which are modeled as a pair of circles. An extension of the classic Hough transformation which is the circle Hough transformation is employed in detecting those irises. Once they are detected and located, a template of only the bounding parabolas is used to correctly orient and fill in the missing information regarding the eye pair.

Authors have evaluated their system separately both accuracy wise and performance wise with respect to each module. The eye module and mouth module have shown

around 55% 45% fits respectively with adequate accuracy. Performance wise the mouth detection has consumed an average time of 3.23 seconds and the eye detection has consumed an average time of 11.545 seconds. Considering the above results, it is safe to assume that we cannot expect real-time performance from this method when run on a mobile platform.

2.3 Face tracking

2.3.1 Feature based face tracking

2.3.1.1 Kanade-Lucas-Tomasi tracking algorithm

Dirk W. Wagener and Ben Herbst came up with a face tracker as described in [19]. The aim of this project was to develop a robust and effective face tracker based on the Kanade-Lucas-Tomasi (KLT) tracking algorithm. The project concentrates mainly on robust tracking despite excessive background clutter. The authors have also investigated the problem of disappearing and re-generation of features on a person's face. The system could also be used with two video cameras to track a person's face in three dimensions.

The authors have come up with three steps for the purpose of tracking the face. Those locate the face in an image stream, selecting features to track and tracking them. Good features are located by examining the minimum eigenvalue of each 2×2 gradient matrix, and features are tracked using Newton-Raphson method of minimizing the difference between the two windows [19]. When features are lost, the algorithm replaces the lost features by finding new features. The KLT algorithm detects a set of object points across the video frames. Once the algorithm detects the face, the next step detects feature points that can be constantly tracked.

The aim of the feature selection is to find the coordinates in the image which have a varying texture. Areas with a varying texture pattern are mostly unique in an image, while uniform or linear intensity areas are often common and not unique.

The technique for feature tracking can be summarized as follows.

1. First the features which can be tracked from image to image in a video stream

are selected based on texture (intensity information).

2. A number of fixed-sized feature windows are selected on the first image of a sequence.
3. These feature windows are tracked from one video frame to the next using the KLT algorithm.
4. Then the KLT feature point tracking algorithm uses Sum of Squared intensity Differences (SSD) of the window that needs to be tracked as the measurement criterion to realize the tracking of feature points. The target of KLT tracking algorithm is to calculate the translation distance $d = (\Delta x, \Delta y)$.

Processing an image with a high resolution is computationally expensive. As such, the authors tried to minimize the processing time in order to keep up with a moving person. Hence the KLT algorithm can be used in a multiresolution image pyramid for processing images. Since KLT algorithm is remarkably robust for tracking facial images, it could be used in mobile platforms with minimized processing time.

2.3.1 Model Based Face Tracking

2.3.1.1. Active Shape Models

Cootes et al. in 1995 proposed a method of Active Shape Models (ASM) to address the problem of image alignment. It is a statistical model of shape that captures the variability of a particular object class given an annotated set of training images [20]. It captures the variability of shape by constructing a mean shape and deriving main modes of shape variation in terms of eigenvectors. Each of these modes change the shape by moving landmarks along straight lines normal to the shape contours through mean landmark positions. New shapes are generated by modifying the mean shape with weighted sum of these modes.

ASM approach consists of two stages. First one is the model building stage and the second is the stage of image search by model fitting. The ASM consists of a shape model and a profile model. In the stage of model building, the training images are first applied using Procrustes Analysis to align the set of facial feature points marked

in the images, removing any translational and rotational movement. Then PCA is applied on those points to generate a set of eigenvectors and their corresponding eigenvalues which represent the main modes of variation in shape. The shape model is this mean shape plus a selected set of most significant modes of variation multiplied by a vector of shape parameters p . In the process of model fitting, it is this p set of parameters which is required to be found. The shape model described here is represented by eq. 2.4 where x is the mean shape, p is the matrix of main modes of variation and Q_s is the vector of shape parameters.

$$x = x' + Q_s p \rightarrow (2.4)$$

During the stage of training, for each landmark point, a profile model is constructed in order to locate approximate position of each landmark by template matching. These profile models form fixed length normalized gradient vectors or profiles along the lines orthogonal to the shape boundary (whiskers) at each landmark. Hence during training, a mean profile vector g' and a profile covariance matrix S_g is calculated at each landmark. And during searching it displaces landmarks along the whisker to the pixel whose profile g has the lowest Mahalanobis distance from the mean profile g' . The definition of the Mahalanobis distance is given by eq. 2.5.

$$\text{Mahalanobis distance} = (g - g')^T S_g^{-1} (g - g') \rightarrow (2.5)$$

ASM search is an iterative process of matching the model to the image. For this purpose we need to initialize the model by estimating the initial model position. And the iterative model fitting algorithm searches along the normal via each model point to find the best local match for the model of image appearance at that point. i.e. the strongest nearby edge and update its model parameters to best fit the model instance to the found points. This is repeated several times until convergence.

Lot of the earlier works on ASMs were focused on locating various objects specially in the domain of medical image analysis. Stephen Milborrow and Fred Nicolls proposed some improvements for locating facial features and tracking human faces with ASMs [21]. There they came up with 6 major extensions to the ASM approach which were (i) fitting more landmarks than actually needed (ii) selectively using two

instead of one dimensional landmark templates (along the normal and tangent directions to the landmark contours) (iii) adding noise to the training set (iv) relaxing the shape model where advantageous (v) trimming covariance matrices by setting most entries to zero (vi) stacking two ASMs in series. Incorporating all these improvements to cater to the problem of ASM model fitting in facial images, Figure 2.9 shows a face taken from the BioID [22] dataset with correctly located landmarks.



Figure 2.9: A face with correctly positioned landmarks by improved ASM model fitting [21]

When taking the above improvements into consideration, ASM can be considered to have the ability to robustly track human faces even in mobile platforms along with some more optimizations that would cater to reduce the computation time of the algorithm. This can be done in several ways like decreasing the number of landmark points, decreasing the length of the normal line along which search is carried out and decreasing the number of scales of resolution.

2.3.1.2. Active Appearance Models

Active Appearance Models (AAMs) as proposed by Cootes et al. [23] is a way of matching a statistical model of shape and appearance to facial images. This consists of a model building phase followed by a phase of model fitting. In the first phase, the model is built by combining a model of shape variation with a model of texture variation. AAMs also require a set of annotated training images of faces for building up the model. The shape model component of AAM is built the same way as it is built in the ASM by applying Procrustes Analysis to align the set of landmark points marked in the images and applying PCA to generate a set of eigenvectors and their corresponding eigenvalues. As mentioned above, these eigenvectors describe the modes of variation that are possible among different faces taken in different poses.

As in (4), the shape model used in AAM is described in terms of the addition of the mean shape with a linear combination of eigenvectors. Model fitting is a process of finding the coefficients of these eigenvectors (shape parameters).

In addition, the AAM consists of an appearance model that is trained by warping each training image to the mean shape and calculating a mean appearance model along with appearance eigenvectors that cater to the change in appearance. Hence in general, we can model the appearance as in eq. 2.6. Here g' is the mean appearance, Q_g is the matrix containing the modes of variation and c is the vector of appearance parameters.

$$g = g' + Q_g c \rightarrow (2.6)$$

This creates a model which is deformable to match for different instances of facial figures. In deformable model fitting what is estimated is p , which is the vector of shape parameters. Estimating c is only a byproduct of the fitting algorithm. For estimating p and c , various algorithms including regression, classification and nonlinear optimization methods have been proposed in the literature [24]. In the original paper of AAM proposed by Cootes et al. [23], it follows an iterative matching algorithm by learning the perturbations in the model parameters and the induced image errors to calculate the shape and appearance parameters.

R. Gross et al. have conducted a research on generic vs. person specific AAMs and have established the fact that performance of a person specific AAM which is built to model the variation in appearance of a single person across pose, illumination and expression is substantially better than the performance of a generic AAM. This models the variation in appearance across many faces that include unseen subjects who are not in the training set [24]. Moreover it is also stated that building a generic shape model is far easier than building a generic appearance model and the shape component is the main reason behind reduced fitting robustness of generic AAMs.

They have also proposed two refinements to improve the fitting performance of generic AAMs. One is a new refitting procedure to improve the quality of the ground-truth data used to train the AAM. The other is a new fitting algorithm, the

Simultaneous Inverse Compositional algorithm [26]. In the first refinement which is based on data refitting, they first construct an AAM with original hand annotated facial landmarks and then refit the AAM to original training images and use the vertex locations of the fitted shape as new landmark data. The major reason for doing this is the fact that due to the difficulty in hand annotating a lot of images, the manually labelled facial annotations can be most of the time erroneous and low in quality. Hence this data refitting algorithm can be utilized to improve the ground truth data. Figure 2.10 shows the average rate of convergence for Generic AAMs computed using original and refitted labels.

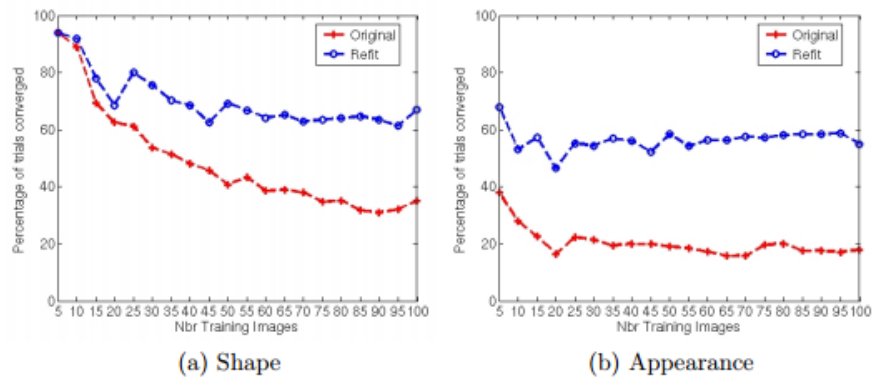


Figure 2.10: Average rate of convergence for Generic AAMs computed using original and refitted labels [25].

The goal of AAM fitting can be formulated as minimizing the sum of squared differences between the model instance and the input image which is warped on the base shape mesh. In the second refinement they have introduced the Simultaneous Inverse Compositional (SIC) algorithm that minimizes this sum of squared differences using Gauss Newton Gradient Descent optimization simultaneously on the warp and appearance parameters. Figure 2.11 shows the average rate of convergence for Generic AAMs using the SIC and Project Out (PO) algorithms using the refit labels. According to the results it can be inferred that SIC algorithm performs significantly better than the PO algorithm with Generic AAM model fitting.

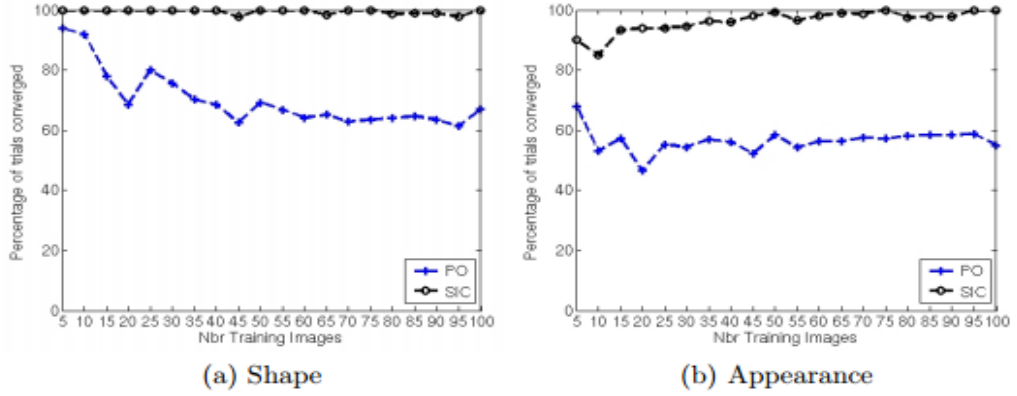


Figure 2.11: Average rate of convergence for Generic AAMs using the Simultaneous Inverse Compositional (SIC) and Project-Out (PO) algorithms using the refit labels [25].

The difference between previously described ASM and the AAM is that ASM only uses image texture information in small regions about each landmark point. It searches around the current position, typically along profiles normal to the boundary trying to minimize the distance between the model points and corresponding feature points in the image. In contrast, AAM uses an appearance model to model the entire facial region. And it only samples the image under the current position and tries to minimize the difference between the synthesized model image and the target image [27].

T.F. Cootes et al. in their work [27] have discovered that the ASM approach is faster than the AAM approach. The ASM approach has taken 190 ms/search while AAM has taken 640 ms/search [27]. The ASM approach has achieved more accurate feature point locations as well than the AAM approach with a failure rate of 1% whereas the AAM approach has shown a failure rate of 1.6% [27]. This is with each model having 133 points marked as facial landmarks and tested on a test set having 400 facial images. And the AAM has been built to represent 5000 pixels. Figure 2.12 shows the histogram for the resulting point-to-boundary errors (the distance from the located points to the associated boundary on the marked images) and the RMS texture error after search from displaced positions. But the fitting performance can be improved by changing the number of facial landmark points taken for model creation. The effect of the number of points against the fitting speed and accuracy is considerable in model based face tracking approaches.

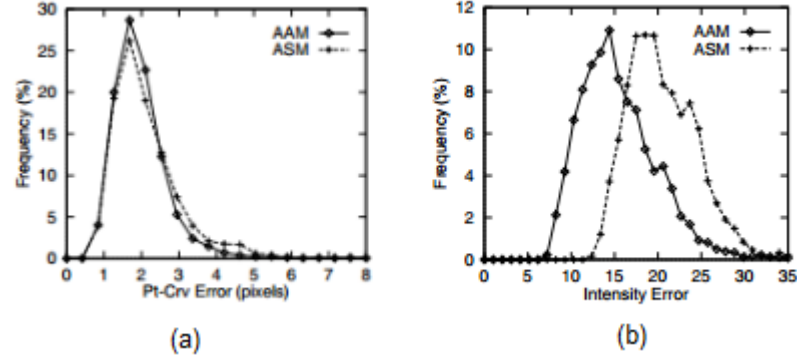


Figure 2.12: Histograms of errors after search from displaced positions for face data. (a) Histogram of point-boundary errors after search from displaced positions (b) Histogram of RMS texture errors after search from displaced positions [27]

However the ASM can be stated as less reliable than the AAM as ASMs only use the data around the model point and do not take into account the grey-level information available across the face. And ASM approach needs a relatively large number of landmarks in order to build a good model. Otherwise ASMs tend to become less robust. In contrast AAMs tend to provide a better match for the texture of a face having low intensity error. And this also has slightly larger capture range than the ASM [27].

Even though AAM based model fitting for facial images is quite common and a vastly researched area in computer vision, effectively fitting AAMs to video sequences still remains a challenging question especially when considering the varying quality of real world video content. L. Xiaoming et al. proposed a hybrid model in order to address this problem [28]. Both a Generic AAM and a subject specific model were simultaneously used in their proposed fitting algorithm, the SIC fOr Video (SICOV) algorithm.

They have tested their proposed method on outdoor surveillance video sequences and it has demonstrated improved image registration and faster fitting convergence against the SIC algorithm across video frame sequences. Figure 2.13 shows the Mean Squared Error (MSE) of neighboring warped frames of the video sequence. As indicated in green, the SICOV algorithm shows significantly lower levels of MSE indicating the improved frame-to-frame image registration accuracy. And Figure

2.14 shows the fitting results observed in video frames with zooming and low resolution where reliability in fitting using SICOV is proven even for a facial area of 15 pixels wide.

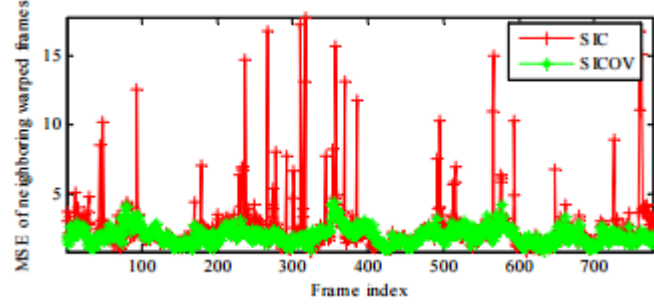


Figure 2.13: The MSE of neighboring warped frames of a video sequence [28]



Figure 2.14: Fitting results with zoom in facial area using SICOV [28]

Through the experiments they have also discovered that the proposed SICOV algorithm not only improves the fitting robustness and accuracy but also the fitting speed. Figure 2.15 indicates the number of iterations taken by each of the algorithms SIC and SICOV for converging onto facial images. The lower number of iterations taken by the SICOV algorithm is an indication that it converges much faster than the SIC algorithm.

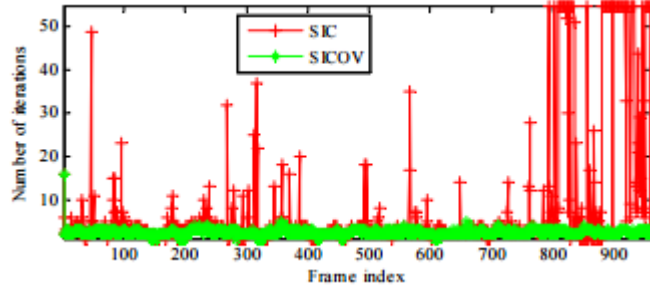


Figure 2.15: The number of iterations in fitting each frame of a video sequence [28]

T.F. Cootes et al. in their work have proposed a variant of AAMs called “View Based Active Appearance Models”. The principle they have utilized in their proposed method is that a small number of 2D statistical models are sufficient to capture the shape and appearance of a face from any viewpoint. They have shown how such set of models can be used to track faces through large angles of head rotation, estimate the head pose and synthesize faces from unseen viewpoints.

As described above, many authors have come up with so many improvements and variants of the AAM. Even though these models are not reasonably tested in mobile platforms, we can come to the conclusion that with proper optimization steps which specially involves reduction of the number of landmark points in the model, reduction of the number of modes of variation and the number of scales of resolution, the AAM can be optimized to be used in mobile applications to track human faces in real time.

2.3.1.3. Constrained Local Models

Constrained Local Models (CLMs) is a framework proposed by Cristinacce et al. [29] which uses a joint shape and texture appearance model to generate a set of region template detectors. It follows an iterative approach in fitting the model onto unseen images. The appearance model used here is quite similar to that used in AAM [23]. But instead of using a global appearance model built by triangulation of the shape model of the face, CLM uses a different approach by learning variation of appearance of a set of template regions or patches surrounding individual features. These image patches are assumed to be conditionally independent from each other which has contributed for greater performance when compared to other holistic

approaches of model fitting [30]. These image patch detectors are learned from annotated training images of faces using techniques such as SVMs [30].

In order to fit the model onto unseen facial images, the CLM requires a fairly accurate initialization of the facial feature points. Given this initialization, the algorithm uses the joint model of shape and appearance to generate a set of region templates around these points. Those templates are applied to the image which generate a set of response image patches for each feature point. Then the CLM search as shown in Figure 2.16 is done by finding the shape parameters so as to maximise this sum of responses.

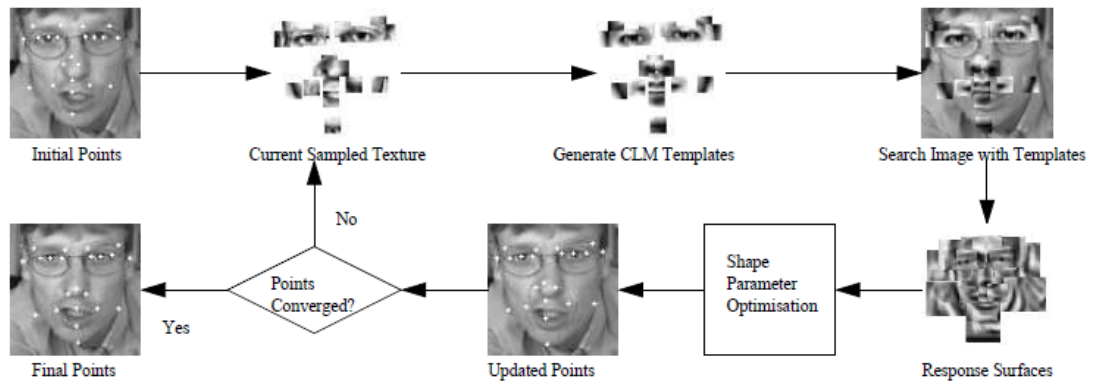


Figure 2.16: CLM Search [29]

This model was later extended by Saragih et al. in their seminal work [30] where they focus on the problem of ambiguity in using local detectors due to their small local support and large appearance variation in training. They introduced a method of cooperating individual response maps generated by the local detectors for an enhanced effect, limiting their effects of unambiguity.

Asthana et al. have proposed a method of robust Discriminative Response Map Fitting (DRMF) with CLMs [31] which is a regression based approach for fitting CLMs to new facial images. Their aim in the research was to use off-the-shelf regression techniques to learn robust functions from the response maps for updating shape parameters. This research has resulted in superior performance in scenarios of generic model fitting.

2.3.1.4 Deformable 3D face models

L. Unzeuta et al. in their research have found a robust and lightweight method for automatic fitting of generic, deformable 3D face models on facial images [32]. Unlike the above discussed model fitting approaches which use a set of facial images with manually labeled facial landmarks to train the model, this detects facial features based on local image gradient analysis and back project a deformable generic 3D face model through the optimization of its deformation parameters. This avoids disadvantages arising from training based approaches such as requiring the new images not to differ too much in illumination conditions, shape and appearance from the images that were used for training [32]. Therefore this has outperformed many other face fitting alternatives under challenging illumination conditions and on devices with low hardware specifications such as mobile phones and tablets and hence can be described as a better approach to be used in mobile AR applications of our interest. Figure 2.17 shows some experimental results obtained in detecting facial features and fitting the generic face model based on the detected features.



Figure 2.17: Experimental results obtained for generic face model for images [32]

2.3.1.5. Optical flow constraints on deformable models

A mechanism for face tracking with a deformable anthropomorphic 3D model is described in [34]. They have developed a dynamic system for treating optical flow information as a hard constraint on the motion of a deformable model. Their work also guarantees that the constraint remains satisfied when combined with edge

information in order to overcome error accumulation in tracking. They have used Kalman filters to relax the constraints and enable optical flow and edge information to be combined robustly. The use of a three dimensional detailed shape model had been very helpful to account for self-occlusion and track the head rotation and it also allows for accurate descriptions of facial shape. Finally, by designing the model with a separable shape and motion parameterization, they could separate the problem of estimating the shape of a face from estimating the motion. Through this work, they have tried to prove the idea that use of optical flow as a hard constraint and use of Kalman filters to accommodate these constraints are useful techniques to improve the expected output of the system.

But the system exhibits several limitations. One limitation is the idealization of the optical flow constraint equation, which had not accounted for the problems of photometric variation and self-shadowing. Another limitation is the inefficiency in tracking motions larger than the width of the derivative filters. Multiscale model based optical flow techniques could be used to tackle this problem when developing real time AR applications that operate in different platforms.

2.3.1.6. Dynamic Templates and Re-Registration Techniques

The work described in [35] presents a method to recover the full motion of the head from an input video using a cylindrical head model. Here the full motion refers to three rotations and three translations. This head model is created using an initial reference template of the head image and the corresponding head pose to recover the head motion automatically. To make this approach more robust, they have utilized three techniques. First, it uses the Iteratively Reweighted Least Squares (IRLS) technique in conjunction with the image gradient to accommodate non-rigid motion and occlusion. Second, the templates are dynamically updated while tracking to get rid of the effects of self-occlusion and gradual lighting changes and maintain accurate tracking even when the face moves out of view of the camera. And third, to minimize error accumulation in the use of dynamic templates as this system re-register images to a reference template.

The authors have tested the method as part of a facial expression recognition system

in spontaneous facial behavior with moderate head motion [35]. The 3D motion recovery was sufficiently accurate with 98-100% accuracy for blink recognition. A detailed geometry of the head can be reconstructed with the recovered 3D head poses and the tracked features. The fact that the system runs in real time and that the system is robust to full head occlusion and video sequences, provides a clue that it would perform well in mobile applications as well. Using this approach, it is possible to successfully stabilize face and eye images allowing for 98% accuracy in automatic blink recognition.

2.3.1.7. Evaluation of speed and accuracy in model based face tracking

For obtaining a quantitative evaluation of model fitting approaches, the most popular method used is testing the performance of the algorithms against hand annotated images of faces which are defined as ground truth parameters [23]. The model can be displaced by predetermined amounts of $(\pm) 10$, $(\pm)20$, $(\pm)30$ pixels etc. and the Root Mean Square (RMS) point to point error between the model points and the hand annotated feature points can be measured to obtain a figure for the accuracy of model fitting. This can be done several times by changing the number of iterations specified for model convergence and plot the RMS error against the number of iterations. By this we can calculate the average rate of convergence [24].

Another performance measure is the average frequency of convergence where it takes into account the number of iterations taken by each algorithm to converge [24]. The convergence when the RMS point to point error becomes less than a predefined number of pixels can be specified.

Face tracking using model based approaches as stated above, is a scenario of fitting a trained model onto facial images in consecutive video frames. In this case, a fairly reasonable instantiation of the model is required in the first frame which can be done by scaling the mean shape to the face detected by a face detection algorithm such as Viola Jones [2]. Thereafter, model fitting in the consequent frames can be done by using the shape parameters and location of facial landmarks obtained after fitting the model in the previous frame. Therefore, an evaluation of speed and tracking accuracy of the algorithms when used in a video sequence can be done by plotting the RMS

point to point error against the sequence of video frames in a video feed [24]. These evaluation results can be utilized to compare the pros and cons in selecting an algorithm for a face tracking scenario in mobile devices which requires real time performance with limited computational power and memory.

2.4. Head Pose Estimation

Head pose estimation is a matter of estimating the rotation around the three axes x, y and z which are termed as roll, yaw and pitch. There are so many algorithms which are proposed in the literature which achieves the target of estimating human head pose. These algorithms can also be categorised into different approaches such as model based, feature based, appearance based and hybrid approaches.

2.4.1. Model Based Head Pose Estimation

2.4.1.1. POSIT algorithm

POSIT (Pose from Orthography and Scaling with Iterations) algorithm is a widely used algorithm proposed by Dementhon et al. [36] for estimation of the 3D pose of an object from a 2D image. For detecting the pose, it requires four or more detected features in the 2D image with known relative geometry of the object which they represent. This method combines 2 algorithms, one is POS (Pose from Orthography and Scaling) and the other is POSIT (POS with Iterations). The first algorithm approximates the perspective projection with a scaled orthographic projection and finds the rotation matrix and the translation vector of the object by solving a linear system. The second uses the approximate pose found by POS in its iteration loop to compute better scaled orthographic projections of the feature points and then applies POS to these projections instead of the original image projections.

Experimental results have shown that POSIT converges to accurate pose measurements in a few iterations with low computational cost. Increased number of feature points contribute in increasing the pose estimation accuracy and compensate for measurement errors and image noise. One of the advantages of using POSIT is that it does not require starting from an initial guess. And as it computes the pose using fewer floating point operations, it can be considered as a useful approach for

real time pose estimation. This makes it a good candidate algorithm to be used in real time mobile AR applications. Figure 2.18 shows the number of iterations taken by the POSIT algorithm as a function of the distance to the camera.

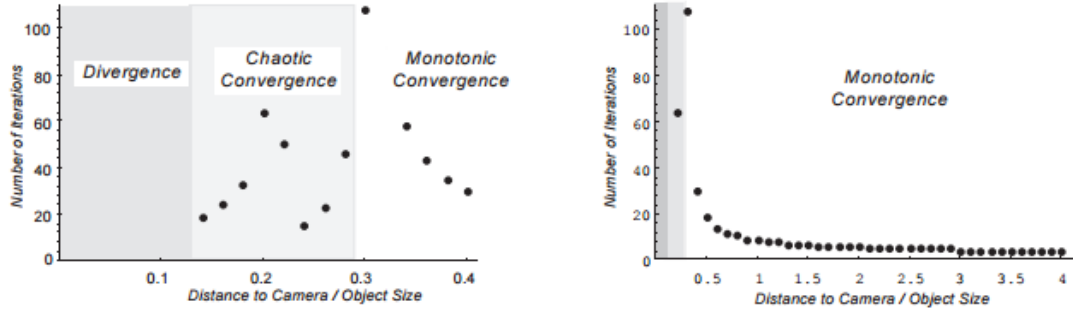


Figure 2.18: Number of iterations for POSIT as a function of distance to camera [36]

The major point to note here is that with the increase in distance-to-size ratios, the POSIT algorithm would converge in four or five iterations with significant increase in speed.

2.4.1.2. Non-rigid Structure from Motion and Point Correspondence

G. Zhenghui and Z. Chao in their work [37], proposed a 3D head pose estimation system using Non-rigid Structure from motion (SFM) and Point Correspondence. The model proposed by them basically addresses both the aspects of tracking and 3D pose estimation of human faces. The problem of estimating largely varying pose and expression changes in video sequences obtained from un-calibrated monocular cameras have been addressed by them. They have also pointed out some disadvantages in classical model based head pose estimation methods such as (i) always requiring a 3D head model or a reference frame to estimate the pose (ii) the camera r needing to be calibrated in advance and (iii) the difficulty of the models to deal with non-rigid motion of human faces. The method proposed by them has the ability overcome all the above disadvantages.

SFM is a popular method proposed in the literature [38] to successfully recover 3D shapes and estimate motion from 2D monocular videos. But the defect in them is that they are valid only for recovering 3D shape from static scenes and hence is not suitable to recover dynamically changing human head pose from video streams.

Hence research is being conducted to construct non-rigid SFM approaches in estimating human head pose and motion.

Just as the POSIT head pose estimation algorithm, this method also utilizes a 2D AAM in each video frame to track the changing pose and expressions of the human face in 2D image frames. Then they utilize their non-rigid SFM technique to recover the 3D shape from the 2D model. This can accurately recover the pose using robust statistics (RANSAC) and point correspondences. And they have also found that the resulting rotation matrix from traditional SFM techniques are not strictly orthogonal and hence lead to inaccurate rotation estimation and inaccurate results due to self-occlusion and inaccurate feature location. Therefore in their approach, they have proposed an adjustment step of using face symmetry before pose estimation, rather than directly using the rotation matrix resulting from traditional SFM.

They have tested their approach on representative frames covering large rotation and expression variations in video clips of people with frame counts varying from 300 to 500. They have obtained better results in yaw and pitch information when computed with the proposed approach using RANSAC and point correspondence than the original SFM algorithm.

2.4.1.3 View Based AAMs

Using “View Based AAMs” Cootes et al. have been able to capture the full 180^0 rotation using only 5 models, roughly centered at viewpoints at 90^0 , -45^0 , 0^0 , 45^0 and 90^0 [38]. They have used these models to track the face through wide changes in orientation and estimate the head pose. Figure 2.19 denotes the training sets used to train each of the models. $\pm 90^0$ and $\pm 45^0$ are simple reflections of each other and hence there needs to be only 3 distinct models of shape and appearance.

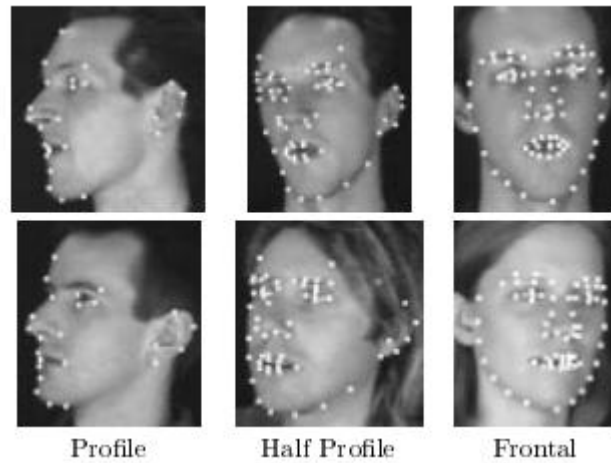


Figure 2.19: Examples from the training sets for the models [39]

By using the previously discussed AAM algorithm, we can match any of these individual models to new facial images quite rapidly. If we know the head pose in advance, we can quite easily select one of the models that is closest to that head pose and if not we can try each of these models on the new image and select what best fits the new image. Once the model is selected, we can easily estimate the head pose as well and thus track the face switching to a new model if the head pose varies significantly.

They have used this system to track 15 new video sequences of people in the training set each comprising of 20 to 30 video frames [39]. Figure 2.20 indicates the estimate of the angle from tracking against the actual angle. As it denotes, in most of the cases it has yielded good estimates of the head pose and only in one case the model has lost tracking and had been unable to recover itself back. This implies that this head pose estimator has reasonable pose estimation accuracy to be used in AR applications requiring accurate orientation details. Yet the system they discuss had been tested off-line, loading sequences from disk.

On a 450MHz Pentium III machine it had run about 3 frames per second with little work done to optimize it. Hence this has to be further optimized and tested on mobile platforms to see if it meets the real time requirements on such platforms.

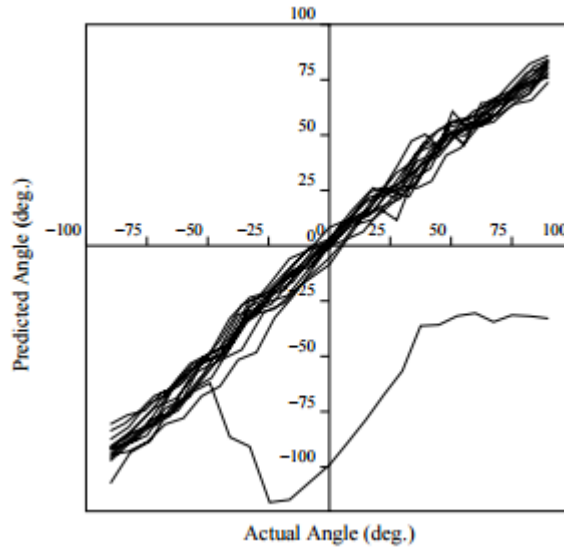


Figure 2.20: Comparison of angle derived from AAM tracking with actual angle (15 sequences) [39]

2.4.1.4. Patch based pose estimation with continuous regression

Most of the techniques related to head pose estimation attempt to estimate pose in a limited range. They treat pose as a classification problem by assigning the face to one of many discrete poses. Most of them have been tested on images taken in controlled environments. For example with solid or constant background, with small or no variation in illumination and expression. The work described in [40] addresses the problem of estimating the pose as a continuous regression problem on real world images with large variations in the background, illumination and expression which makes it robust against real world situations that can be experienced by mobile AR applications.

Their approach breaks the test image into a non-overlapping regular grid of patches where each is treated separately. They provide independent information about the true pose [40]. This specific algorithm contains a predefined library of object instances, which can be considered as a palette from which image patches can be taken. The test image patch is approximated by a patch from the library, which can be thought of as having a different affinity with each pose. These affinities are learned during the training process and then they are embodied in a set of parameters (W). The relative affinity of the selected patch from the library for each pose is used

to determine a posterior probability over the pose [40]. The overview of this algorithm is shown in Figure 2.21.

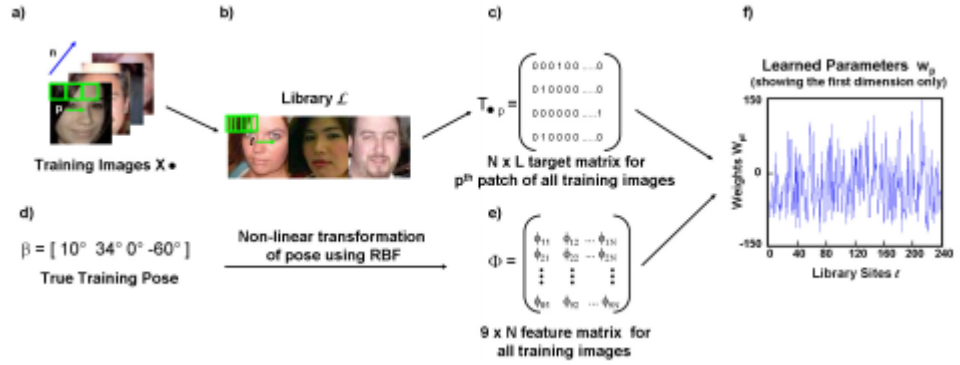


Figure 2.21: Overview of patch based pose estimation

(a) The model is trained from N training images, each of which is represented as a non-overlapping grid of patches indexed by p . (b) The library L is considered as a collection of patches L_l where l indexes the L possible sites. (c) Use 1-of- L coding scheme to represent the closest library patch \hat{l} and store them in a target matrix T_p for the Face Distribution p^{th} patch of all training images. (d) The true poses are stored in a $N \times 1$ vector β . (e) Columns of Φ represent ϕ_n : 9D radial basis functions of pose parameter β for the n^{th} training image. (f) Learn the parameter vector w_{pl} which represents the tendency for the library site l to be picked when considering patch p of an image with pose vector ϕ_n [40].

2.4.1.5. Automatic 3D model construction using SMAT, RANSAC and POSIT

An interesting work related to face tracking and pose estimation with automatic 3D model construction is presented in [41]. This paper describes a successful approach taken to robustly track and estimate the face pose of a person using stereo vision. This method is special because it is invariant to identity and it does not require previous training. A face model is automatically initialised and constructed on line. When the face comes frontal to the camera, a fixed point distribution is superimposed over the face, then several points close to those locations are selected for tracking. The 2D projections of the model points are tracked separately for left and right images using SMAT model refinement. Feature points are tracked using SMAT, incorrectly tracked points are rejected using RANSAC and finally the 3D

pose is recovered from the 2D points set using POSIT.

In this approach, 3D face pose is estimated from 2D projections. Matching poses may not succeed for all points and so it can result in errors or drifting for some of them. These errors degrade the accuracy of the estimated pose. Therefore, a robust method is required to estimate the best 3D face pose, so that points that are incorrectly tracked can be detected as outliers and safely discarded. Even the points that have been correctly tracked may have some random noise. RANSAC algorithm is used as a solution to overcome this problem. The complete system is outlined in Figure 2.22.

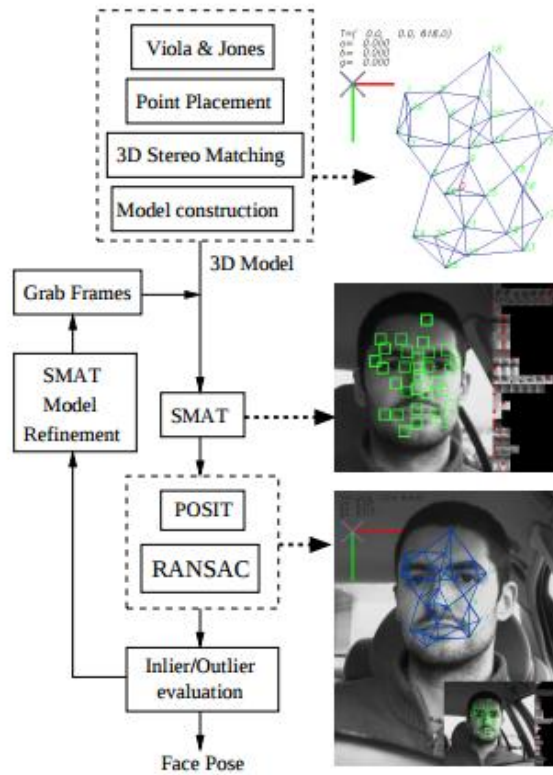


Figure 2.22: System block diagram [41]

In each RANSAC iteration, 7 points are randomly selected from the model and they are used to calculate the pose using POSIT. 3D original points of the model are projected over the image plane and Euclidean distance from the tracking point to the corresponding projected point is calculated. If the distance is less than a threshold, that point is considered correct and marked as an inlier.

Kalman filters have also been made use of to smooth the final results of the estimated pose. Points identified as outliers by RANSAC algorithm are moved to correct positions, so that they could be again tracked on the following frames, which is also known as feature point tracking failure detection and recovery. Patches corresponding to outlier points on the last frame are considered as outliers and removed from the SMAT model. When the magnitude of expressions of the person is small, the errors could be corrected using RANSAC and POSIT. A limitation that exists in this system is that it requires the user to look straight ahead for a few frames.

With this approach, head rotations up to $\pm 45^\circ$ are correctly estimated while serving in real time. This method has been used as a basis of a driver monitoring system and so it has been tested on sequences recorded in a moving car. This method can also be expected to give reasonable real time results when run on a mobile platform.

2.4.2. Feature based head pose estimation

Feature based head pose estimation as the name suggests refers to estimating the head pose or the gaze direction based on the geometric positioning of features extracted from a human face. This method can provide pose estimations of reasonable accuracy and the accuracy can be improved by increasing the number of features processed.

In the research done by A.H. Gee and R. Cipolla on estimating human gaze direction from a single monocular image [18], they attempt to find the facial normal of a given facial image by considering the geometric distribution of the facial features.

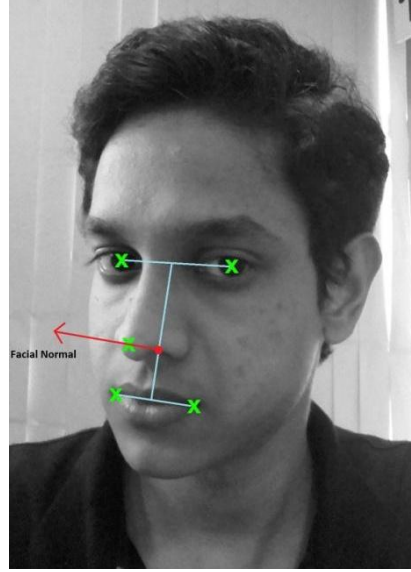


Figure 2.23: The facial normal

As shown in Figure 2.23, Cipolla and Gee suggests that once the positions of the two eyes, two corners of the mouth and the nose tip is correctly found, the facial normal can be easily computed from them. This method is based on the assumption that in every human being, the ratio between lengths l_m and l_f as shown in Figure 2.24, stays roughly the same.

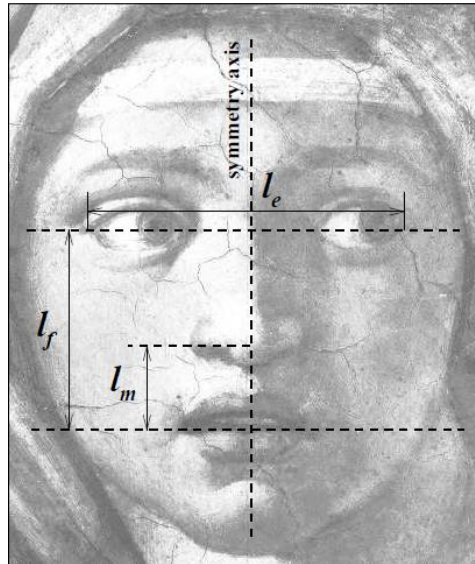


Figure 2.24: Lengths between facial features [18]

Let's consider $R_m = l_m/l_f$, point between the two eyes as A and the point between the two corners of the mouth as B. Because of the symmetry of the human face, the line

joining A and B can be considered as the symmetric axis. Now from the value of R_m , it is possible to find a point in line AB that corresponds to the base of the nose. Once the base of the nose is found, the direction from the base point to the tip of the nose as shown in Figure 2.23, gives a fair estimation of the gaze direction of the face. The individual values of the roll, yaw and pitch of the head can be inferred from 3D geometry by calculating the relevant angles of the direction vector with respect to the image plane.

One issue with this approach is the necessity of having both the eyes visible in the given image. Still the computation of the pose values pitch, yaw and roll are straightforward and consumes a minimum amount of computational power. Therefore this method can be expected to give reasonable real time results even when run on a mobile platform.

2.4.3. Appearance Based Head Pose Estimation

2.4.3.1. Template matching

2.4.3.1.1. Multi view face detection and pose estimation using SVMs

SVMs have shown greater potential in resolving different computer vision applications. The method described in [42] uses SVM classification algorithm to enable simultaneous multi view face detection and pose estimation at near frame rate. SVMs are used to generalize and transform a generic 2D facial appearance model across the view sphere, to investigate face detection at different views and to implement pose estimation efficiently. In this related work, they had used multi view SVM based face model to perform both multi-view face detection and pose estimation across views.

SVMs perform automatic feature extraction. They construct complex nonlinear decision boundaries for learning the distribution of a given data set. The learning process and the number of support vectors for a data set should be decided by the algorithm. That should depend on different parameters like Lagrange multipliers to distinguish between noisy data and the parameter for determining the effective range of Gaussian support vectors. This work adopts a semi iterative approach for

obtaining good examples of negative training data from a database of randomly selected scenery pictures. In this approach, they have tried to extend the training of a single frontal view SVM face model to the use of face images across the view sphere.

It is difficult to determine a function which best describes the underlying structure of the data distribution. Structural Risk Minimization (SRM) is used as a remedy to this problem and it provides a great quantitative measure for the capacity of a learned function to generalize over unknown test data. SRM provides a guaranteed minimal bound on the test error. For example, SVM defines a hyperplane and the decision boundaries of the two classes if it is a linearly separable two class recognition problem. The face pose distribution across the view sphere provides a basis for learning a generic face model based on multi view SVMs. Pose estimation had been automatically performed by Gaussian kernel functions used in the multi view SVMs. More accurate pose estimation can be achieved by using a better aligned training set.

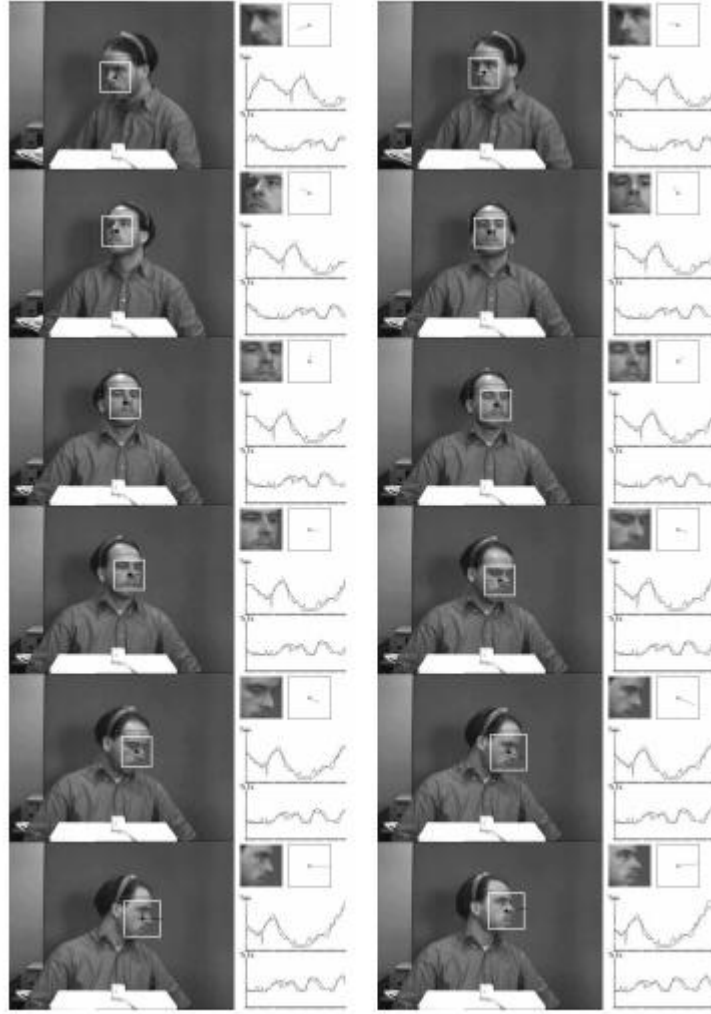


Figure 2.25: Examples of near-frame rate face detection, tracking and pose estimation using a multiview composite SVM. On the right of each picture are detected faces in each image frame, its pose estimated in a dial, and the estimated pose versus the ground-truth from the Polhemus sensor over time [42].

When we consider the real time performance, multi view SVM classification is still not computationally attractive for real-time use. As such this method could not be expected to give reasonable real time results when run on a mobile platform. However, image scanning using global optima search methods provides a clue for faster tracking.

2.4.3.1.2. Identity independent head pose estimation from prototype images

Jamie Sherrah, Shaogang Gong and Eng-Jon Ong have developed a view based

statistical learning technique based on the similarity of images to prototype real time identity independent head pose estimation from a single 2D view. Their work as described in [43], requires pose invariant face detection. The complexity of calculating real time head pose increases with changes in illumination, identity and facial position. Similarities can be more robustly calculated with PCA. A generic appearance model is obtained by estimating an average face template at each pose. In order to build the appearance model example, views labelled with 3D pose angles are required. Figure 2.26 shows a set of head images obtained by using a special camera and a sensor.



Figure 2.26: An example labelled head image set. The image of labelled views are from +90 to -90 in Yaw and +30 to -30 in tilt at 10° intervals [44].

Facial images must be transformed in such a way so as to emphasize differences in pose while suppressing differences in identity. They investigate appropriate transformations with similarity-to-prototypes. Orientation selective Gabor filters were used to detect features for pose discrimination. PCA had been used for dimensionality reduction and it was found to provide invariance to identity while accurately describing pose changes. PCA is also understandable through visualization and it is more computationally efficient, since similarities are calculated in the low dimensional space. They had also investigated the lowest angular separation at which pose differences can be robustly detected. A greatest lower bound of approximately 20 was determined, and the actual minimum resolution may be 10 or lower at some poses.

This approach uses second-order similarity to obtain robust similarity measures from

sparse data. It estimates head yaw on the range [0,180] and tilt (elevation) on the range [60,120]. The respective face prototype database consists of facial images from several different people at different poses in 10 increments. Given a novel face image and hypothesized head pose as the inputs, the distance of each prototype to the novel face is calculated using the algorithm. The novel face is then represented in similarity space as a vector of dissimilarities as follows.

$$s_i = [d_{i,1}, d_{i,2}, \dots, d_{i,N}]^T \rightarrow (2.7)$$

Here N is the number of human subjects in the database, and $d_{i,j}$ is the distance from the face i to subject j at the same hypothesized pose.

The principle that had been used here is that distances in the similarity space are smoother than in the original high dimensional image space. This whole approach relies on a pose similarity assumption that different people at the same pose look more similar than the same person at different poses. So they consider that pose is a stronger indicator of similarity than identity.

Some issues that can be found in this approach are,

- The optimal filter orientation at each pose is not necessarily unique. Gabor filters may not be the optimal filters for pose estimation. A more general approach could be taken by adapting the filter orientation locally within the facial images.
- The benefits of pose selective filters and PCA need to be combined. The main difficulty is creating PCA bases from images that have been filtered differently.
- PCA may not be the best linear projection for removal of identity information.
- This requires further improvements to facilitate good performance in real time, so that it could be extended to give reasonable real time results when run on a mobile.

2.4.3.2. Detection arrays

2.4.3.2.1. Face pose discrimination using SVMs

Face pose discrimination means that one can label the face image as one of several known poses. The system developed in [45] has also used SVMs mainly with the intention of coming up with a solution for face pose discrimination.

Face images are drawn from the standard FERET database. The training set includes 150 images approximately 33.75° rotated left and right respectively, and the test set consists of 450 images equally distributed among the three different types of poses. Some examples of the training and test image are shown in Figure 2.27.



Figure 2.27: Examples of (a) training and (b) test images [45]

This had achieved perfect accuracy of 100% discriminating between the three possible face poses on unseen test data. This has also proved that SVM provides robust discrimination and classification schemes.

2.4.3.3. Nonlinear regression methods

2.4.3.3.1. Support vector regression and classification based multi view face detection and recognition

Another very interesting work with SVM based multi view face detection and recognition framework is described in [46]. The face detection has been carried out by constructing several detectors in which each of them is specific for a particular

view. This work had made use of the symmetrical property of face images to simplify the complexity of modelling. The estimation of head pose has been achieved by using Support Vector Regression (SVR) technique which provides information for choosing the appropriate face detector. The complete approach can be summarized into following stages:

- Using SVR to construct pose estimators which provides robust and fast estimation of head pose.
- Training a set of Support Vector Classification (SVC) classifiers for multi-view face detection. The symmetrical property of the face images have been used to build the classifiers with only half of the possible views.
- Using pose information to guide the selection of face detectors to further improve the detection accuracy and to ease computation in detection.
- Using the pose change smoothing technique for further computational reduction.
- Activating a SVC multiclass classifier to perform face recognition when faces are in frontal view. As the appearance change of faces in frontal view is smoother than in other views, this method provides robust recognition performance.
- All the above issues were integrated in a SVM based framework.

The implementation of the system can be demonstrated as in Figure 2.28.

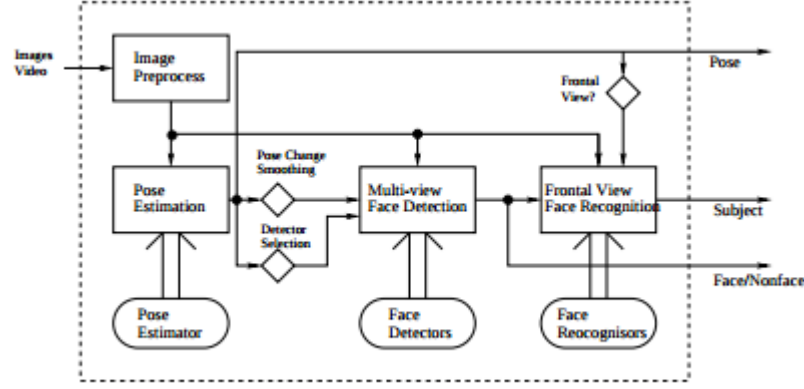


Figure 2.28: Multi-view face detection and recognition [46]

For video sequences, further improvements could be achieved by using a kind of pose change smoothing strategy. Test results from four video sequences have shown a detection rate of above 95%, recognition accuracy of above 90% and average pose estimation error around 10° . Therefore, with several more optimizations that cater to the performance in mobile platforms with low computational power and memory, this approach could be expected to perform reasonably well in real time mobile applications as well.

2.4.3.3.2. SVR with Localized Gradient Histograms

Erik Murphy-Chutorian, Anup Doshi, and Mohan Manubhai Trivedi have developed an identity and lighting invariant system to estimate a driver's head pose. This system, as described in [47], is capable of operating online in daytime and nighttime driving conditions by making use of a monocular video camera sensitive to visible and near infrared light. This system utilizes Localized Gradient Orientation histograms with SVM for regression. The algorithm estimates the orientation of the driver's head in two degrees of freedom.

In developing this system, they attempted to meet three special criteria; Monocular, Autonomous, and Invariant. They wanted the system to work regardless of the specific driver and operating conditions. This approach uses soft histograms of location specific gradient orientation as the input to a support vector regressor for each degree of freedom. The operation of the system consists of following steps.

- Facial regions are found by three cascaded Adaboost face detectors applied to the grayscale video images.
- The detected facial region is scale normalized to a fixed size and used to compute a Localized Gradient Orientation histogram.
- The histogram is passed to two SVRs trained for head pitch and yaw.

A graphical overview of the system is shown in Figure 2.29.

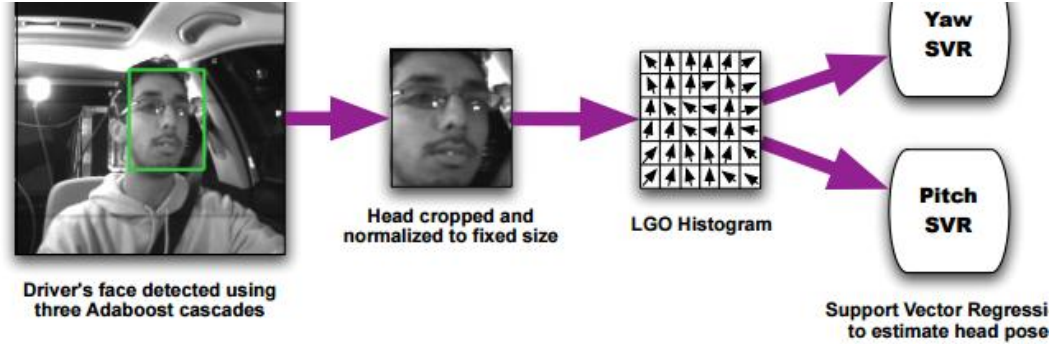


Figure 2.29: An overview of the proposed Head Pose Estimation System [47]

This specific method exhibits a significant improvement in estimation accuracy and it maintains performance in the variable automotive setting.

2.4.3.3.3. SVR with sparse representations

A very interesting work related to pose estimation is presented in [48]. This is about developing a high performance JGCF pose estimation system based on the sparse Bayesian regression technique, which is also known as Relevance Vector Machine (RVM) and sparse representation of facial patterns. Facial properties are represented using sparse features of 20 key localized facial points. RVM is used to find the relationship between the sparse representation and the yaw and the pitch angles.

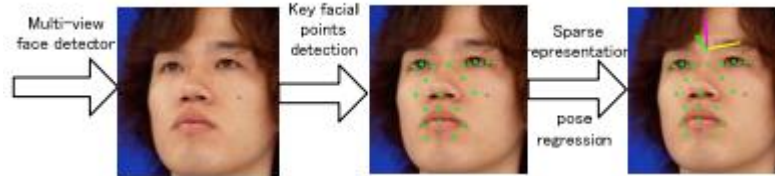


Figure 2.30: Flow chart of the pose estimation system [48]

RVM utilizes very few kernel functions, thereby guaranteeing better generalization,

faster speed and less memory. The performance of this system has been compared with several other conventional methods like CCA, Kernel CCA, SVR etc. The experimental results had shown that this system could estimate face pose more accurately, robustly and faster than those based on conventional methods. So the authors ended up with a conclusion that the face recognition system based on Gabor wavelet feature was very robust to facial points localization error.

A similar work of estimating facial pose from a sparse representation is described in [49]. This had taken into consideration the prominent facial features relative to the position of the head. They have developed a high dimensional, randomly sparse representation of a human face using a simplified facial feature model. This specific model converts a raw face image into a vector which represents how well the image matches large number of randomly posed head models. This conversion needs collecting salient features of the face image that is useful to estimate the pose, while removing any irrelevant variations of the face image. This system makes use of SVR to find the relationship between the sparse representation and the pose. The sparse representation combined with SVR had shown promising results in estimating the pose more quickly and accurately. This approach would definitely be useful when developing real time applications related to augmented reality.

2.4.3.3.4. Head pose estimation using neural networks

Neural networks technique could also be used to estimate the head pose. Several examples of this approach can be found in history.

One such example system that provided a coarse estimate of head pose is presented in [50]. This work had used two neural networks which have been trained to approximate the functions that map an image of a head to the orientation of the head. This system combines head tracking and localization with a neural network based head orientation estimation system. The head localization subsystem detects and extracts the image region which corresponds to a human head. This head region is then applied as an input to a neural network system for head orientation estimation. This work had proved that it is important to have an accurate head localization system in order to obtain better results from the head orientation estimation system.

Another significant work related to recognition of human head orientation based on Artificial Neural Networks was developed by Robert Rae and Helge J. Ritter [51]. They have used a neural network method based on three neural networks of the local linear map type which facilitates identification of the head orientation of a user by learning from examples. First network has been used for color segmentation, the second for localization of the face, and the third for the final recognition of the head orientation. The most significant feature of this system is that all recognition and identification steps like segmentation, face localization, and extraction of head orientation were entirely based on local linear map network modules that were synthesized from training data examples. The feature extraction stage uses a set of heuristically parameterized Gabor filters.

The system exhibits few limitations. With this approach the system is able to identify the face direction of a person in front of a complex laboratory background and the user's head is required to remain within the camera view area. This system has worked well for the persons included in the training data but it has not worked well for previously unseen persons as the generalization on new users is a restriction of this approach.

2.4.3.3.5. Gabor Wavelet Networks for head pose estimation

Gabor Wavelet Network (GWN) combine the advantages of continuous wavelet transform with RBF networks. The use of Gabor filters enables the coding of geometrical and textural object features. Gabor filters can be used as a model for object features to ensure data reduction and allow any desired precision of the object representation ranging from sparse to photo realistic representation.

Volker Krueger and Gerald Sommer have introduced the GWN as a model based approach for effective and efficient object representation [52]. They have developed a system for head pose estimation based on the GWN. The feature information has been encoded in the wavelet coefficients. Then an artificial neural network has been used to compute the head pose from the wavelet coefficients. In this approach, GWNs have been used successfully for wavelet based affine real time face tracking and pose invariant face recognition, while exploiting all the advantages of the GWN

for estimating the head pose.

2.4.3.3.6. Convolutional networks

A very robust and novel method for simultaneously detecting faces and estimating the head pose in real time is described in [53]. The technique that they have used is a convolutional network which integrates detection and pose estimation. This network is capable of mapping images of faces to points on a low dimensional manifold, parameterized by pose. The images of non faces have been mapped to points far away from that manifold. The network is trained by optimizing a loss function of three variables called image, pose, and face/non-face label. When these three variables match, the energy function is trained to have a small value. When these three variables do not match, it is trained to have a large value. In fact, the system has been designed to handle very large range of poses without retraining.

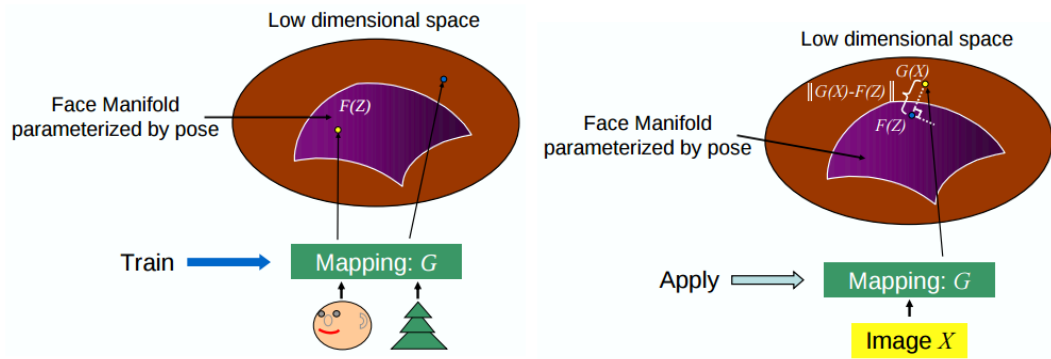


Figure 2.31: Manifold mapping-Training Estimation and Manifold Mapping-Recognition and Pose [53]

Convolutional network can be defined as an end to end trainable system that can operate on raw pixel images. They learn low level features and high level representation in an integrated fashion. Each layer in a convolutional network contains units organized in planes called feature maps. Each unit in a feature map takes inputs from a small neighborhood within the feature maps of the previous layer and computes a weighted sum of its inputs and passes the result through a sigmoid saturation function. Therefore, each feature map can be seen as convolving the feature maps of the previous layers and passing the sum of those convolutions through sigmoid functions.

Consider one slice of the network with a 32×32 input window. This slice includes all the elements that are necessary to compute a single output vector. The trained network is replicated over the full input image to produce one output vector for each 32×32 window stepped every 4 pixels horizontally and vertically. The process is repeated at multiple scales as demonstrated in the Figure 2.32.

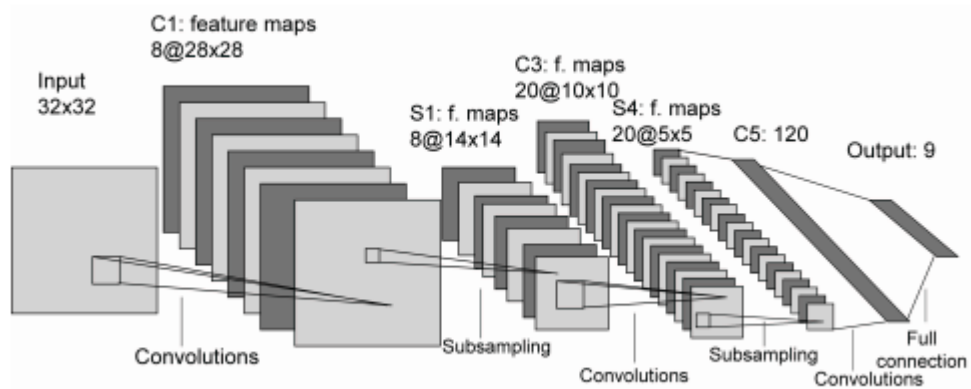


Figure 2.32: Architecture of convolutional network used for training [53]

This system possesses several significant features as listed below [53].

- The use of a convolutional network makes it fast. It can process 5 frames per second on a 2.4Ghz Pentium IV at typical webcam resolutions.
- It is robust to a wide range of poses, including variations in yaw up to $\pm 90^\circ$, rotation up to $\pm 45^\circ$, and pitch up to $\pm 60^\circ$. This has been verified by testing the system using three standard datasets specific for yaw, roll and pitch angles.
- Another significance of the system is that it produces estimates of pose at the same time that it detects faces.

This work has experimentally shown a higher state of accuracy at both pose estimation and face detection. It is fast, but seems to be computationally expensive which would make it difficult to be used for mobile applications unless doing any further optimizations.

2.4.3.4. Manifold embedding methods

2.4.3.4.1. Subspace analysis with the topography method

Subspace analysis is a widely used technique for head pose estimation. But, such techniques are usually sensitive to data alignment and background noise. Junwen Wu and Mohan M. Trivedi have proposed a two-stage approach to address this issue by combining subspace analysis together with the topography method. The first stage (Figure 2.33) is based on the subspace analysis of Gabor wavelet responses. The pose estimation is taken by nearest prototype matching with Euclidean distance [54].

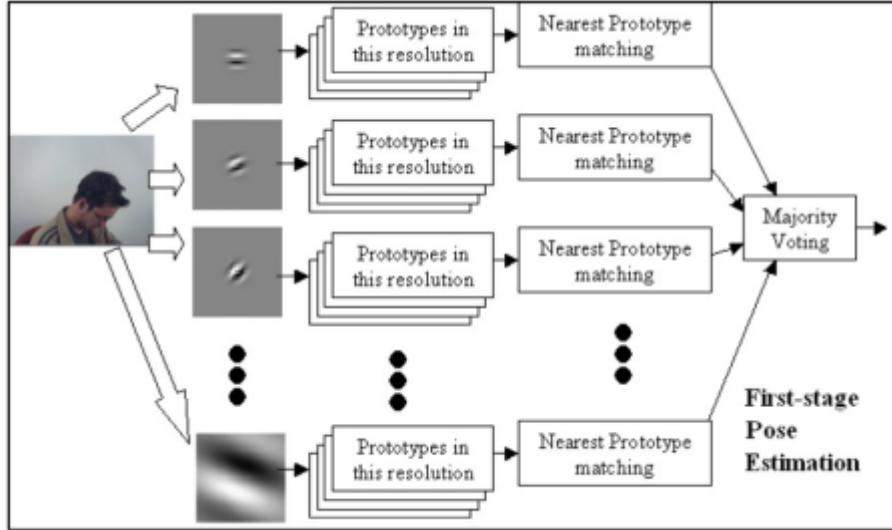


Figure 2.33: First stage pose estimation [54]

In the second stage (Figure 2.34), the pose estimate obtained from first stage is refined by analyzing finer geometrical structure details captured by bunch graphs.

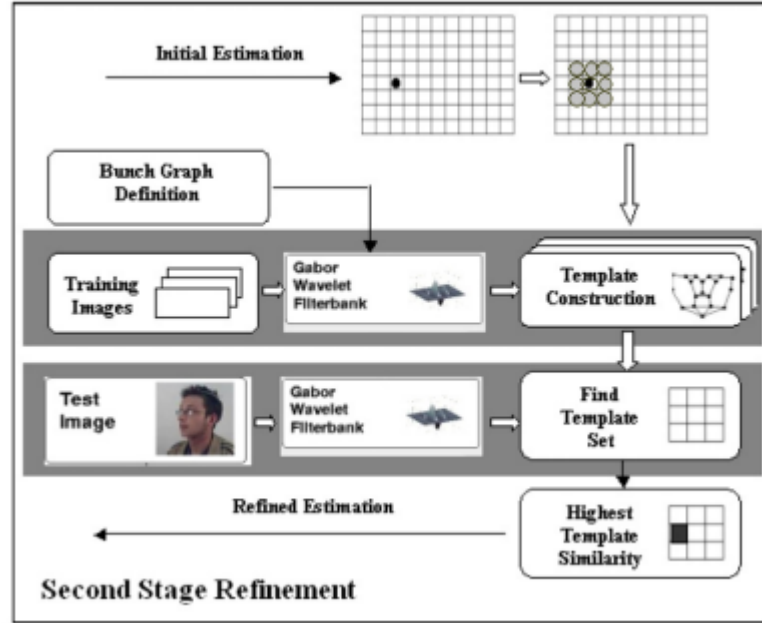


Figure 2.34: Second stage refinement [54]

According to experimental results, the proposed two stage algorithm helps to lower the computational cost and achieve a higher performance gain. More robustness is added to the algorithm by the combination of statistical analysis on features from entire images with the geometrical topography driven approach.

2.4.4. Hybrid approaches for head pose estimation

Yuxiao Hu et al. proposed a robust pose estimation approach combining facial appearance asymmetry and 3D geometry. In this approach, the rough face pose is first estimated by analyzing the asymmetry of the distribution of the facial component. Thereafter this rough face pose is refined to get the pose estimation by using a 3D-to-2D geometrical model. The proposed approach has the ability to track a face which has a cluttered background and to recover its pose robustly and accurately in real-time [55].

Junwen Wu et al. presented a two-level approach for estimating face pose from a single static image. In this approach, Gabor wavelets are used as the basic features. In the first level, an estimate of the pose is derived within some uncertainty. In the second level, the output from the first level is systematically processed to minimize its uncertainty by analyzing finer structural details captured by the bunch graphs.

Gabor wavelets which are joint spatial frequency domain representations are considered as good feature detectors. The classification results from different Gabor wavelets combined by majority voting [56].

Youding Zhu and Kikuo Fujimura came up with an adaptive head pose estimation method for driver attention monitoring. PCA and 3D motion estimation were used for head pose estimation from an image sequence. The algorithm performs accurate pose estimation by learning the subject appearance. Depth information is used effectively in this algorithm to segment the head region even in a cluttered background and to perform 3D head motion estimation based on optical flow constraints. The face pose angle is calculated incrementally and it is reset whenever the image is frontal to avoid drift [57].

Louis-Philippe Morency et al. presented a method for estimating the absolute pose of a rigid object based on intensity and depth view-based eigenspaces which are built across multiple views of example objects of the same class. Given an initial frame of an object with unknown pose, a prior model is reconstructed for all views represented in the eigenspaces. For each new frame, the pose-changes are computed between every view of the reconstructed prior model and the new frame. The resulting pose-changes are then combined and used in a Kalman filter update. This approach for pose estimation is very useful as it is user independent and the prior model can be initialized automatically from any viewpoint of the view-based eigenspaces [58]. They have integrated an Adaptive View-Based Appearance Model (AVAM) tracking framework which was presented in [59] to track the face more robustly.

2.5. Related APIs and SDKs

2.5.1. Google Mobile Vision API

Google Mobile Vision API was developed as a way to facilitate finding objects in photos and video, using real-time on-device technology [60]. This API has the capability to find human faces in photos, videos or live streams. It also has the ability to track facial landmarks such as the eyes, nose and the mouth as denoted in Figure

2.35. It first detects the entire face independently of the landmark points, and if landmark detection is optionally specified by the developer, it detects the facial landmark points as well, which takes an additional time [61]. This API can also estimate the pose of the head in terms of discrete angles of yaw and roll as denoted in Figure 2.36. It currently does not support the pitch angle. The API additionally provides information about the state of facial features such as, are the subjects smiling or are their eyes open etc.



Figure 2.35: Example facial landmarks identified and tracked by Google Mobile Vision API [61]

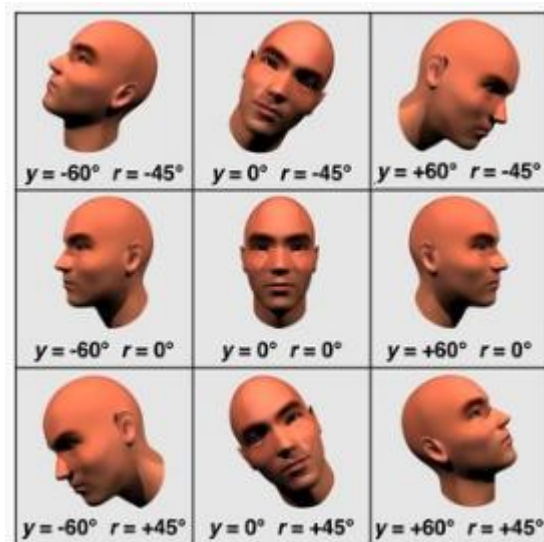


Figure 2.36: Pose angles determined by Google Mobile Vision API where y = yaw and r = roll [61]

This Vision API can be integrated to mobile applications running on Android 2.3 (Gingerbread) or higher [62]. And this can be used with less difficulty behind mobile AR applications to enhance input video feeds with various effects and decorations. It could also be used to render virtual content on top of the identified human faces in

the location and orientation, identified by the API. This opens up a whole new realm of possibilities in the domain of mobile augmented reality. Yet one of the major drawback of using this API is that it does not expose the underlying algorithms or the techniques which are utilized in detecting and tracking the face and estimating the head pose. Hence it does not provide the capability for the developers to tweak the code and modify it the way they want. Therefore it is quite difficult to modify and optimize the underlying algorithms to suit specific scenarios.

2.5.2. State of the art face tracking and analysis technology with Visage Technologies

visage|SDK™ is an excellent software package that could be utilized to develop applications that involve interactive animation, avatar control, sophisticated user interaction and other advanced features in games, arts and other applications [63]. Visage Technologie's mobile and adaptable technology is a very significant feature of this software. The software provides accurate tracking of head pose, gaze direction, facial actions and eye closure. Moreover, it could be used for development of various applications which requires monitoring drivers or operator attention and focus. The software has several components as described below.

FaceTrack [64]: visage|SDK™ FaceTrack package is a very powerful, fully configurable face tracking toolkit. This SDK can be used easily to develop solutions for different applications that require implementing features like face tracking. This is capable of operating in real time to track the face and facial features in video streams. Additionally, this provides excellent user experience through special features such as face masking, virtual makeup, virtual glasses try-on and many others. Some significant features of FaceTrack are listed as follows [64].

- For each processed video frame, this returns 2D and 3D head pose along with other important information such as [65],
 - 2D and 3D head pose (translation and rotation).
 - Facial feature coordinates in global 3D space relative to the head or in 2D image space.
 - 3D gaze direction (gaze vector) and eye closure.

- Screen-space gaze coordinates: the point where the user is looking on the screen.
- A set of Action Units (e.g. jaw drop, lips stretch, brow raise etc.) describing the current facial expression.
- Standard MPEG-4 FBA Face Animation Parameters
- 3D model of the face in current pose and expression, returned as a single textured 3D triangle mesh. This is useful in AR applications to correct occlusion of virtual objects by the head, cut out the face from the image, draw the 3D model from a different perspective than the one in the actual video or insert the face into another video or 3D scene.



Figure 2.37: 3D triangle mesh of current pose and expression [64]

- Tracks the mouth contour, chin pose, eyebrow contours, eye closure and eye rotation (gaze direction).
- Fully automatic operation.
- Instant initialization of face tracking when the face is visible.
- Robustly recovers from any losses due to occlusions, face turning away, tracked person entering and leaving etc.
- Automatically re-initializes if a new person appears in front of the camera.
- Lightweight technology enables high performance across a whole range of supported platforms. The FaceTrack SDK is compatible across operating systems like Windows, OS X, RedHat, Ubuntu, development platforms like Unity, HTML5, Flash, Xilinx and mobile platforms like Android and iOS.

HeadTrack [66]: visage|SDK™ HeadTrack engine can be used for applications that require comparatively high performance and robust 3D head pose tracking in a video sequence. This is same as FaceTrack, but this tracks and outputs only head translation and rotation.

FaceDetect [67]: visage|SDK™ FaceDetect engine performs face detection by identifying facial features in images which have one or more human faces. This returns the 2D and 3D head pose and 2D and 3D coordinates of facial feature points like chin tip, nose tip, lip corners etc. Detecting multiple faces and facial features in input images and fully automatic operation are some significant features of FaceDetect.

FaceAnalysis [68]: visage|SDK™ FaceAnalysis package utilizes different machine learning algorithms to provide gender estimation as to male or female, probability of various emotions and approximate age estimation. FaceAnalysis uses FaceDetect and FaceTrack to detect and track human faces in images or video. Then it uses state of the art techniques to estimate gender and six main emotions like happiness, sadness, surprise, anger, disgust and fear and some combinations of emotions.

One significant feature in visage|SDK™ is that its lightweight technology enables high performance across multiple platforms. It is compatible across operating systems like Windows, OS X, RedHat, Ubuntu, development platforms like Unity, HTML5, Flash, Xilinx and mobile platforms like Android and iOS [64].

Oriflame Makeup Wizard is an application that operates in Android and iOS mobile platforms and run on visage|SDK™. This enables virtual try on of makeup from the Oriflame makeup collection [64]. User can virtually apply lipsticks, mascaras, eye shadows, blush and eyeliners using the phone's camera in order to get a real time view of how they look on his or her face.

2.5.3. OpenCV (Open Source Computer Vision Library)

OpenCV is an open source computer vision and machine learning software library which is a BSD-licensed product. It provides interfaces for C++, C, Python, Java and MATLAB. Furthermore it supports Windows, Linux, Android and Mac OS [69].

OpenCV has expanded its support for mobile application development after it was ported to the Android environment in 2010 and to iOS in 2012 [70]. The computer vision and machine learning algorithms available in the library can be used for many applications such as detecting and recognizing faces, identifying objects, tracking camera movements, tracking moving objects and many more.

2.6. Related Commercial Applications

2.6.1 TryLive

Combining real world object tracking techniques such as face tracking with AR can bring about drastic changes in how humans pursue life. Inspiration can be drawn from applications such as TryLive which provides a platform for retailers and e-commerce merchants to take their showrooms to the customer's home by providing intuitive try-on experience for a range of products such as eyewear, apparel, watches and furniture [72]. This uses face pose tracking techniques optimized for web, mobile and in-store use.

TryLive is a 3D product visualization and a virtual try-on solution that provides the users with virtual shopping experience. It uses AR technology to provide immersive interaction with real time movements and also provides the ability to manually scale the rendered content. It has an inbuilt face tracking algorithm that detects the eyes and the mouth and has the ability to differentiate faces of children, adults, men or women. TryLive Online runs using Adobe Flash Player and supports Stage3D (Adobe Flash Player API for rendering interactive 3D graphics with GPU acceleration [73]) to display 3D content [72]. This also has a mobile version which is compatible with Android and iPhone applications.

2.6.2 Masquerade

Masquerade is a popular showcase mobile app for face tracking and 3D face placement [71]. The app is well known for its ability to track the position and the orientation of the human face and put on various types of 3D masks and enhance the user's appearance over live video. This has a proprietary self-learning face tracking algorithm built into it and a framework for creating special graphical effects. Even

Facebook has taken over Masquerade which shows the popularity of this technology in the modern society.

Masquerade is a mobile application which has gone an extra mile in detection and tracking human head pose and gestures. It allows the user to overlay animated filters over their faces using the mobile phone's front facing camera. It takes in live video feed from the camera and enhances the person's appearance over live video. For this, it uses highly optimized self-learning mathematical algorithms for face tracking that performs well in both modern mobile devices as well as in legacy ones [71]. Its algorithms are quite accurate even in low lighting conditions because they have been trained on libraries containing millions of people's faces and continuously enhanced over many years of development.

2.6.3 Snapchat

Snapchat is an image messaging software product created by Evan Spiegel, Bobby Murphy, and Reggie Brown and it consists of a feature to create special graphical effects over the user's face. Snapchat is primarily used for creating multimedia messages referred to as "snaps". These snaps can consist of a photo or short video, and can be edited to include filters and effects, text captions, and drawings. It also includes a special feature known as "Geofilters", which allows special graphical overlays to be available if the user is within a certain geographical location, such as a city, event, or destination. The "Selfie lens" feature allows users to add real-time effects using face detection into their snaps. They are activated by long-pressing on a face within the viewfinder [74].

The AR filters used in Snapchat tap into the large and rapidly growing field of Computer Vision. The first step they follow is face detection which is done using the Viola Jones face detection algorithm by looking for areas of contrast between light and dark parts of the image. Once the face is detected, it locates facial features inside the enclosing window of the face detector. It does this using the ASM algorithms described in section 2.2.1. This places a 3D mask on human faces that can rotate, move and scale along with the face as video data comes in for every frame. And they can deform the mask to change the user's face shape, change his eye color, put

virtual accessories and trigger animations when the user moves his jaw or raises his eyebrows [75].

2.7. Results

This study covers a basic set of approaches that can be used for face detection, face tracking and head pose estimation. A summary of those different techniques and their performance is outlined and included in Appendix - A, Appendix - B and Appendix - C.

2.8. Discussion

In this literature review, we presented a comprehensive overview of face detection, tracking and head pose estimation algorithms, related APIs and SDKs and mobile AR applications which have utilized such techniques in providing real time AR experience for the users. As different type of algorithms had been designed taking specific conditions and limitations into account, and as most of them are not optimized to work in a mobile environment, we cannot come to a specific conclusion on which algorithms are best suited to achieve real time face tracking and head pose estimation on mobile platforms. Yet as stated after each method discussed in the review, general conclusions can be drawn that some of these algorithms can be integrated with mobile AR applications to obtain real time performance. But for this, these algorithms require some further optimizations and to be tested in different mobile platforms under varying conditions of illumination, image quality etc. We have also discussed some related APIs and SDKs that can directly be used to achieve the same results, and listed down the limitations in each of them when trying to use them in real time mobile AR applications. It is up to the developers to evaluate this information and decide whether to proceed with an existing SDK or to build a system from scratch.

We also brought forth some commercial mobile AR applications that have been successful in using such techniques for real time face tracking and head pose estimation. By drawing inspiration from such applications, we have proven that achieving real time face tracking and pose estimation with existing algorithms is

possible.

Taking all this information into account, one would be able to have a general idea on which specific types of algorithms are suitable to be used in different mobile AR applications.

2.9. Conclusion

In most of widely used AR applications, the techniques mentioned above are utilized to create useful applications that can visualize content using the AR technology on top of human heads. Applications such as TryLive, Masquerade and Snapchat are some examples of such applications. There are various types of face detection, tracking and head pose estimation algorithms which are used behind those applications to render 3D content on top of real world scenes in the correct location and in the correct orientation. Still such commercial level applications also have some limitations when it comes to real time, robust face tracking and head pose estimation. And specially when it comes to the domain of mobile applications, we need to think of other constraints such as limited processing power and memory as well as the low quality in video streams captured by the front facing mobile cameras. Taking all these into consideration, developing an algorithm that is optimized for face tracking and head pose estimation using the front facing camera in mobile devices is a challenging task. For this we need to analyze the pros and cons of the available face detection, tracking and head pose estimation techniques, take the best out of them and integrate into one single framework.

3. RESEARCH METHODOLOGY

The aim of this project is to build a pluggable module for Unity3D, which can be used to build mobile AR applications that can provide better customer engagement in businesses by allowing customers to virtually try out facial accessories such as eyewear and earrings. The scope of this research include investigating various types of face detection, tracking and head pose estimation algorithms that exist in the literature, evaluating them to select the optimum algorithms that meets our requirements and implementing the selected algorithms using a low language such as C#. The following steps were carried out to meet these aims.

- A comprehensive literature survey was carried out to identify the existing face detection and face tracking algorithms, where 56 peer reviewed journals and conference papers spanning from 1992 to 2014 were reviewed.
- A subset of three face detection algorithms and three face tracking algorithms were selected from the literature. The face detection algorithms include the Viola-Jones algorithm, a neural network based algorithm and a state vector machine based algorithm. Face tracking algorithms were considered under two categories as feature based trackers and model based trackers. The feature based Kanade-Lucas-Tomasi (KLT) point tracker algorithm and the model based Active Appearance Model (AAM) and Constrained Local Model (CLM) approaches were implemented. The algorithms were chosen based on their popularity and frequency of appearance in the literature.
- An experimental implementation of hybrid approaches combining those approaches was also done using Matlab.
- A comprehensive evaluation of the implemented face detection approaches and face tracking approaches was then carried out. Face detection approaches were tested against an image sample of the LFPW image dataset. A test video suit containing four videos of facial movements was developed for testing the performance of face tracking algorithms. Those approaches were then tested with respect to evaluation criteria.

A summary of the research methodology that we followed is shown in Table 3.1.

Table 3.1: Research methodology

Proposed Stages	Percentage Completed (%)
1. Performing a comprehensive research on the work done so far in the field of head pose estimation and coming up with a detailed list of approaches that can possibly be followed. Once this step is completed, the breadth of the research will be frozen and focus will given on exploring the chosen approaches in depth.	100%
2. Going through all the possible approaches in several iterations and narrowing down the list to a reasonable amount of approaches considering the extent they agree with our requirements.	100%
3. Implementing the selected approaches to be run first on a PC environment and performing a comparison of the performance of each of the approaches based on different criteria like speed, accuracy and precision and further narrowing down the possible set of approaches to proceed with.	100%
4. Implementing the final set of candidate algorithms on a mobile platform to compare their performance and decide on the best candidate for the purpose.	100%
5. Optimizing the algorithm to achieve better results on a mobile platform.	100%
6. Developing the algorithm as a plugin for the Unity	100%

4. DESIGN DOCUMENT

This chapter discusses the design of the proposed mobile application ‘SpecularAR’. The idea behind this application is to realistically render virtual accessories on the human faces in a live video feed captured from the front facing camera of a mobile device. The application runs on the Unity3D platform and has a pluggable face tracking and head pose estimation module integrated in. This particular module acts as the core module of the application and contains implementations of selected algorithms for face detection, tracking and head pose estimation.

4.1. Architectural representation

Figure 4.1 illustrates the architecture of the proposed application. The application consists of three main components Application Controller, Camera Handler and the Head Pose Estimator. The Application controller performs all the coordination required for the proper execution of the application. The Camera Handler assists in reading image frames of 640×480 resolution from the physical camera device. The Application Controller forwards the images returned from the Camera Handler to the Head Pose Estimator for it to do the necessary processing and give back the estimated head pose values. The returned head pose values in terms of pitch, yaw, roll, position and scale are used by the Application Controller to set the pose, scale and the position of the virtual accessory so that it would look realistically superimposed on the user’s face.

The setup used in Unity3D includes a plane object and a camera object as shown in Figure 4.2. The video feed from the front facing camera of the mobile device is rendered as the texture of the plane object and the camera is kept pointed towards it. Different virtual accessories can then be placed in between them so that in the camera's point of view, the virtual accessory would be seen as if it appears on the video feed itself. The position and orientation of the virtual accessory is updated based on the position and pose of the human head tracked in the video feed.

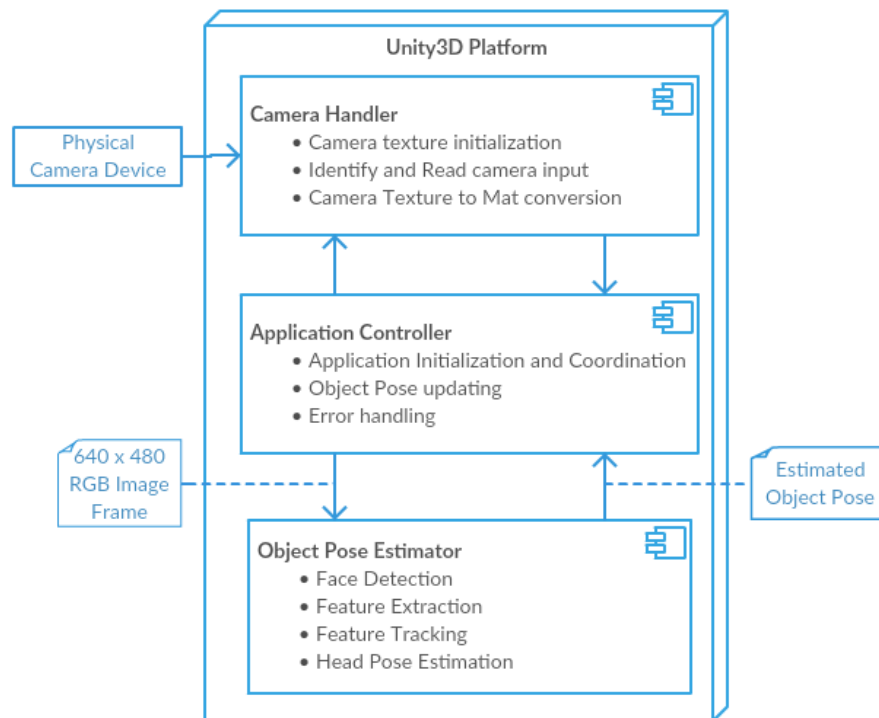


Figure 4.1: Architectural Representation

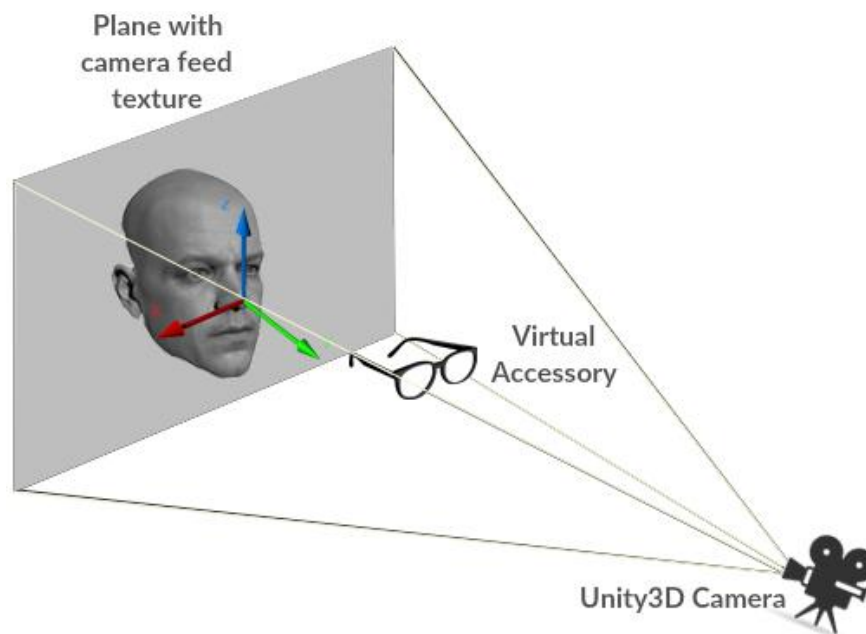


Figure 4.2: Setup in Unity3D

4.2. Architectural Goals And Constraints

4.2.1 Architectural goals

The following are the architectural goals of the SpecuLAR application.

- Developing the face tracker and head pose estimator as a pluggable module that can be easily integrated with the Unity game engine.
- The face tracker and head pose estimator should work in unconstrained environments with different lighting conditions.
- Testing potential paths.
- Building up a prototype of the proposed application.
- The application should be able to connect with an online retail platform to update the newest models and support ordering.

4.2.2 Architectural constraints

The following are the architectural constraints of the SpecuLAR application.

- The face tracking and head pose estimation component should be coded in a language compatible with the Unity game engine.
- The application should give real time performance on mobile devices above Samsung Galaxy S7.

4.3. Use-Case Realization

Figure 4.3 represents the use-case realization of the proposed application. The major functionalities of the application would include self augmentation with virtual models of eyewear and the capability to change the model which the users try on.

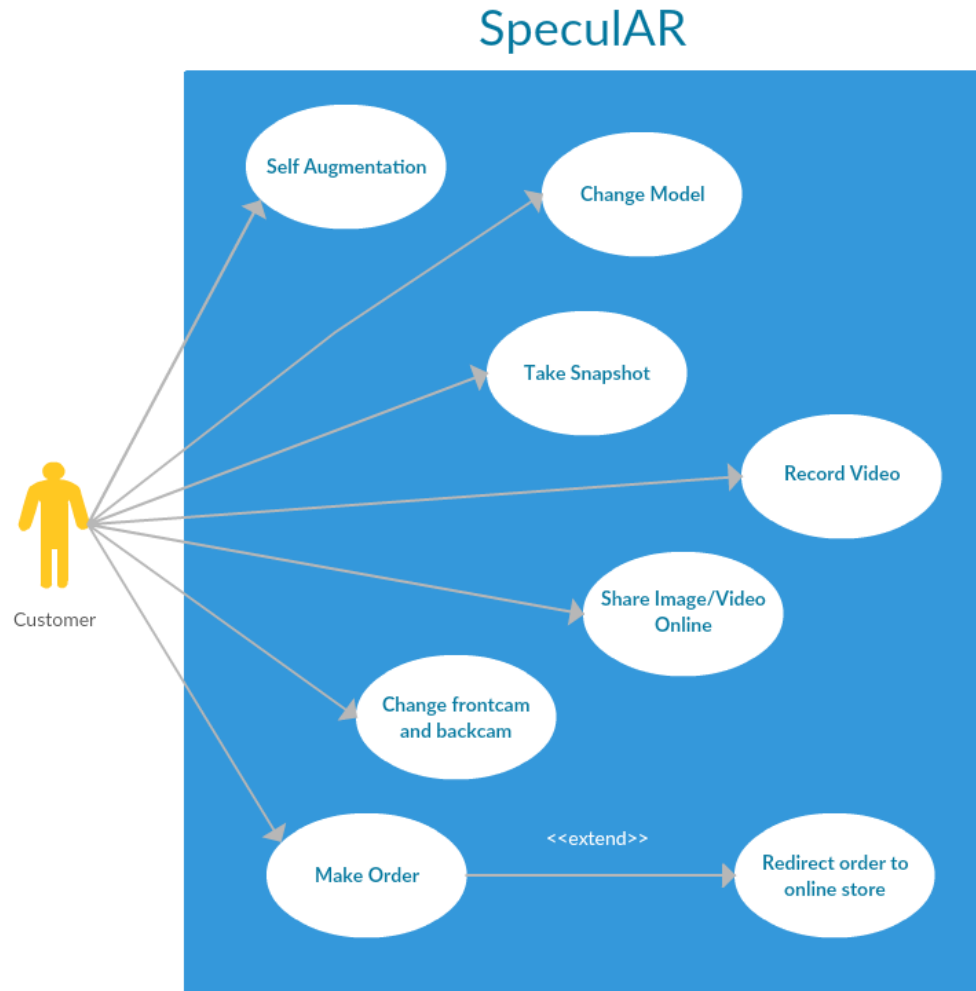


Figure 4.3: Use-case realization

4.5. Process View

The sequence diagram of SpecuLAR is depicted in Figure 4.4. When the user runs the application, the Unity3D platform initializes the Application Controller which would constantly read image frames from the Camera Handler and forward them to the Object Pose Estimator. Utilizing the output from the pose estimator, the Application Controller sets the orientation, position and scale of the virtual accessory. The changes are reflected real-time on the UI by the Unity Engine.

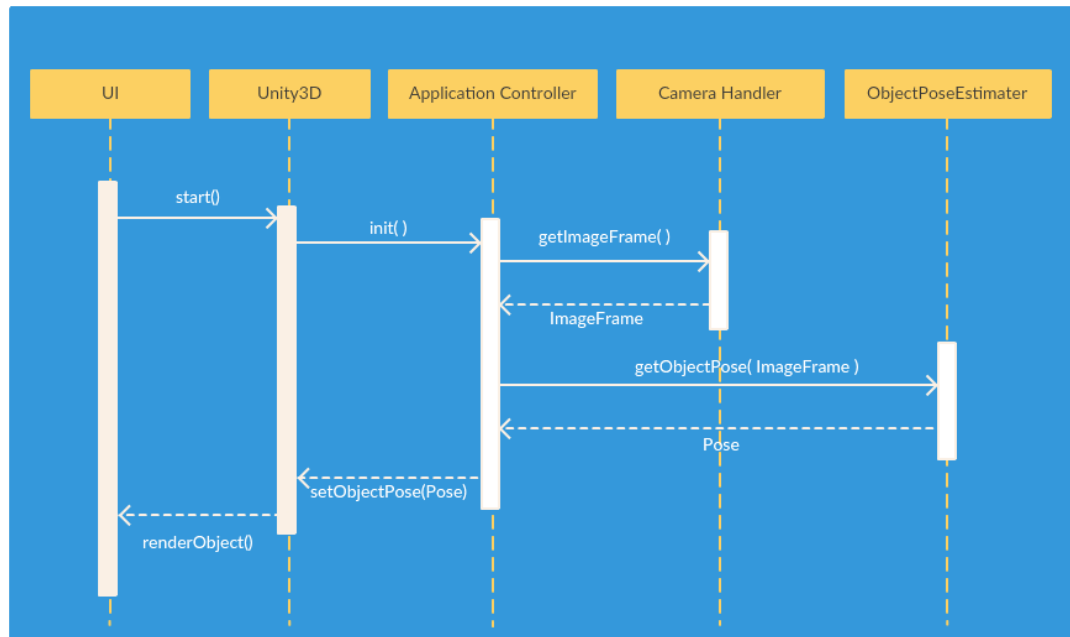


Figure 4.4: SpecularAR Sequence Diagram

4.6. Physical view

The deployment diagram of the application is shown in Figure 4.5. The application that runs on the mobile phone is connected to the business client's server from where the newest models of products can be downloaded and the application can be updated.

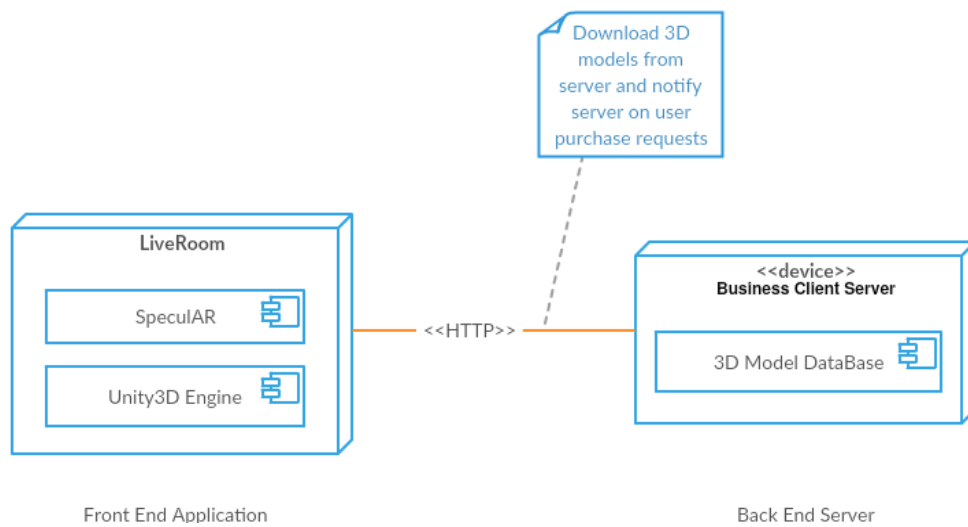


Figure 4.5: SpecularAR Deployment Diagram

5. EVALUATION OF FACE DETECTION AND FACE TRACKING ALGORITHMS

5.1 Evaluation Methodology

A subset of three face detection algorithms and three face tracking algorithms were selected from the literature to evaluate their suitability to find out the optimum algorithms that can be used in implementing the proposed application. The face detection algorithms subjected to evaluation are the Viola-Jones algorithm, a neural network based algorithm and a SVM based algorithm. Face tracking algorithms were considered under two categories as feature based trackers and model based trackers. The feature based KLT point tracker algorithm and the model based AAM and CLM approaches were also used in the evaluation. The algorithms were chosen based on their popularity and frequency of appearance in the literature. MATLAB implementations of the above algorithms were tested against a common dataset. MATLAB provides inbuilt implementations of the Viola-Jones face detector and the KLT feature tracker. Scale-Invariant Feature Transform (SIFT) feature extraction which was required for the KLT tracker was achieved using the VLFeat open source library [76]. Fast-SIC algorithm implementation for AAM fitting given in [77], and the implementation of the CLM algorithm given in [33] were also used in the evaluation. The face detection algorithms were tested against LFPW (Labeled Face Parts in the Wild) image dataset [78]. It includes images downloaded from the web that contain human faces with natural pose variations. A test video suite containing four videos captured in different environments and lighting conditions with different device and camera specifications was used for evaluating the face tracking algorithms. A summary of the specifications of the four test videos used is given in Table 5.1.

Table 5.1: Test video suit specifications

Video	Environment	Lighting condition	Device Specifications	Camera Specifications
1	Outdoor environment	High ambient light (Natural light)	Samsung Galaxy Grand Prime	5 MP, f/2.2, 1080p
2	Indoor environment	Low light (Diffused light)	Samsung Galaxy Core Prime	2 MP
3	Living room	High ambient light (Artificial light)	iPhone 6	1.2 MP. f/2.2, 720p video
4	Opticians Showroom	High ambient light (Artificial light)	Samsung Galaxy Grand Prime	5 MP, f/2.2, 1080p

Each of those videos contain human head motions in the three degrees of 300 freedom, pitch, yaw and roll. Once the videos were captured, 58 facial landmark points were manually annotated in each and every video frame so that they can be considered as ground truths in evaluating the face tracking algorithms. The test video suite along with the annotations has been made available at [79]. All the tests were run on a machine with an Intel Core i5 1.80GHz CPU and 4GB 305 memory.

5.2 Evaluation criteria

The detection rate, number of false positives, number of false negatives, precision, recall and F_1 score were used as accuracy measures and average detection time was used as a speed measure to evaluate the optimality of the face detection approaches. Point to point root-mean-square (RMS) error was used to measure the accuracy of the face tracking algorithms. It was obtained by computing the Euclidean distance between each of the estimated tracking points and the respective annotated ground truth points in each video frame. The Frames Per Second (FPS) value was used to measure the speed of the face tracking algorithms.

5.2.1 Evaluation criteria for face detection algorithms

This section describes the criteria used to evaluate the performance of the face detection approaches implemented in Matlab, in detail.

5.2.1.1 Accuracy

Accuracy measures such as the detection rate, number of false positives, number of false negatives, precision, recall and F_1 score were used to get a comparative measure of the accuracy of the implemented face detection algorithms. F_1 score is a commonly used measure for accuracy and always takes a value between 0 and 1. Higher the F_1 score, better the detection accuracy of an algorithm. The F_1 score is defined in eq. 5.1.

$$F_1 \text{ score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \rightarrow (5.1)$$

5.2.1.2 Speed

In order to compare the speed of the algorithms, the average time taken to detect a face in a facial image was recorded. The tests were run on a machine with an Intel Core i5 1.80GHz CPU and 4GB memory.

5.2.2 Evaluation criteria for face tracking algorithms

5.2.2.1 Subjective evaluation

This provided an indication of whether the values given by the evaluation metrics give a good representation for the human perception of what a good fit is. The method of evaluation is by human observation of the match between the estimated tracking points and the actual hand annotated facial landmark points of the four test videos.

5.2.2.2 RMS point to point error calculation

This method was employed to generate quantitative results on how good a certain algorithm is in correctly tracking the relevant facial features or fitting a facial model. This was done by computing the euclidean distance between each of the estimated

tracking points and the respective annotated ground truth points for each frame in all the four test videos. Finally for each frame, a Root Mean Square value was calculated using all the euclidean distances obtained from each point of interest.

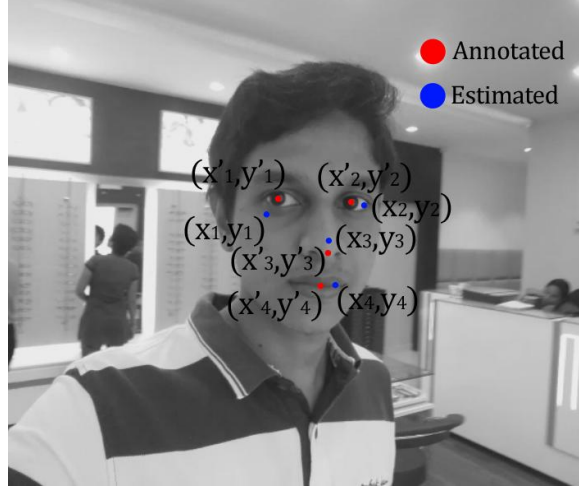


Figure 5.1: Representation of annotated ground truth points and corresponding estimated points in a facial image

Figure 5.1 shows a frame of a video containing a sequence of facial image frames. The Annotated ground truth points of interest are marked in red and the respective sample estimations of those points are marked in blue. Accordingly, the RMS error value for this frame could be calculated as follows.

Let the euclidean distances between each of the corresponding points be d_i and the feature count be n . Here $n = 4$.

$$d_i = \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2} \rightarrow (5.2)$$

Accordingly,

$$RMS = \sqrt{\frac{\sum_{i=1}^n d_i^2}{n}} \rightarrow (5.3)$$

The lesser the RMS value, the better the results given from a particular algorithm are.

5.2.2.3. Frames Per Second (FPS)

The FPS value stands as a good measure of how well an algorithm can support real time performance. A frame rate of around 20 frames per second gives smooth enough video playback for general applications. The higher the FPS value, the faster the algorithm is.

For all our approaches we estimated the frame rate supported by our algorithms by measuring the time it took to process all the four videos and dividing the total frame count from that value. This value plays an important role in deciding the suitability of any algorithm for applications that demand real time performance.

5.3 Evaluation and results of face detection algorithms

Table 5.2 gives a summary of the results obtained using the implementations of the three selected face detection algorithms.

Table 5.2: Summary of results for face detection algorithms

Approach	Detection rate	False Positives	False negatives	Precision	Recall	F₁ Score	Avg. running time (s)
Neural Network	64%	38	17	0.6275	0.7901	0.6995	4.2912
SVM	60%	36	13	0.6250	0.8219	0.7101	6.1807
Viola-Jones	98%	7	9	0.9334	0.9159	0.9246	1.1115

5.4 Evaluation and results of face tracking algorithms

The evaluation results comparing the point to point RMS error values for each of the face tracking approaches tested are presented in the form of line graphs. For comparison purposes, plots corresponding to different methods are shown in different colours in the same graph. The four rectangular areas shaded in different colours in each graph corresponds to the four test videos that were used in the evaluation.

5.4.1 KLT algorithm

Two approaches involving the KLT algorithm were tested in the evaluation. The first approach used four facial features (center points of the eyes, nose tip and the center of mouth) which were detected using a trained image cascade classifier. The second approach was built as an extension of the first approach where in addition, a set of SIFT features inside the face region were also extracted. Those features were then given as inputs to the KLT tracker and the positions of the four facial features were inferred from the geometric movements of the SIFT feature points. Figure 5.2 shows a comparison of the point to point RMS errors obtained from these two approaches. Table 5.3 shows the calculated average FPS values of each of the two approaches involving the KLT feature tracker.

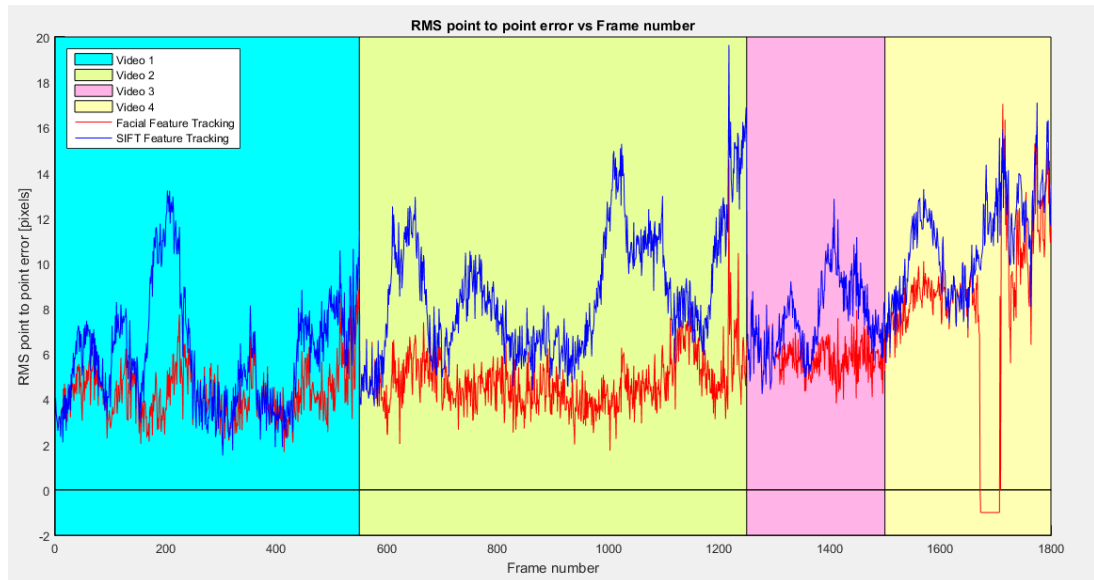


Figure 5.2. Comparison between point to point RMS errors of the two approaches tracking facial features (eyes, nose tip and center of mouth) vs. tracking SIFT features in the facial region using the KLT point tracker.

Table 5.3: Average FPS values for the 2 described approaches used with the KLT tracker.

KLT Tracker	
Tracking facial features (eyes, nose and mouth)	Tracking SIFT features
24.2995	15.4207

5.4.2 AAM algorithm

AAM with Fast-SIC algorithm implementation for model fitting given in [77] was tested in the evaluation. In the given implementation, the model was trained 16 using the LFPW dataset [78] for 68 facial landmark points. This implementation showed poor real-time performance. It focused mainly on the image fitting problem rather than on face tracking. Therefore it was modified by reducing the number of points, the number of iterations as well as the scales of resolution used for model fitting. Three separate models which used 19, 15 and 7 points were built as indicated in Figure 5.3. The shape and appearance models obtained by building up the model using 19 points is shown in Figure 5.4.

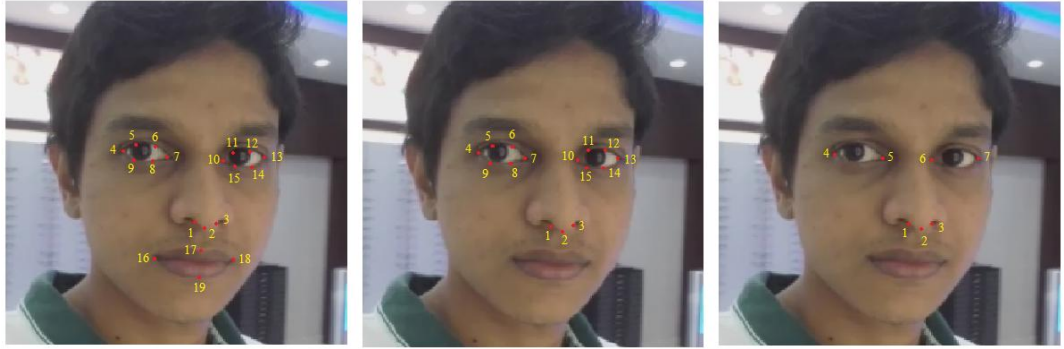


Figure 5.3: 19, 15 and 7 facial landmark points used for training the AAM.

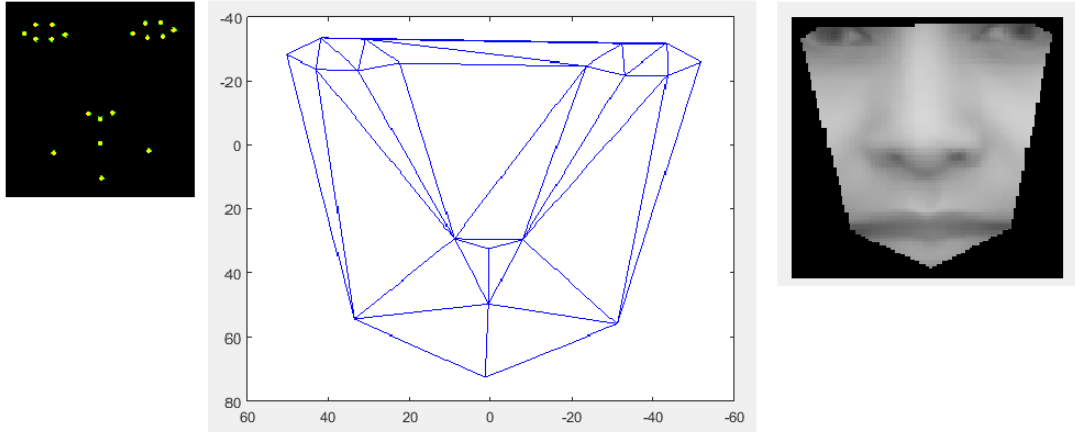


Figure 5.4: The shape model and appearance model of the AAM obtained by using 19 facial landmark points.

In order to make the tracking more robust in the presence of fast pose variations, we incorporated the optical flow estimated by the KLT algorithm for initializing the

AAM in subsequent video frames. The average FPS results obtained for these two approaches using the three models trained while changing the number of iterations used are indicated in Table 5.4.

Table 5.4: FPS for variations of the AAM algorithm

Iterations	AAM			AAM + Optical flow		
	19 pts	15 pts	7 pts	19 pts	15 pts	7 pts
10	5.6173	8.4450	13.2140	5.0758	7.3504	10.5116
5	10.3240	15.3985	21.3573	9.2984	10.3871	15.5072

Figure 5.5 shows a comparison of the point to point RMS error values of both the above approaches as the number of facial landmark points used in the model decreases. Fast-SIC AAM fitting with 10 iterations was used in that. Figure 5.6 shows the same comparison with Fast-SIC AAM fitting with 5 iterations. The focus of those two comparisons is on determining the best combination of the number of facial landmark points and the number of iterations used for each of 360 those two approaches. Finally Figure 10 shows the comparison between Fast-SIC AAM fitting with and without using optical flow estimations for initialization, which were tested using 5 and 10 iterations each.

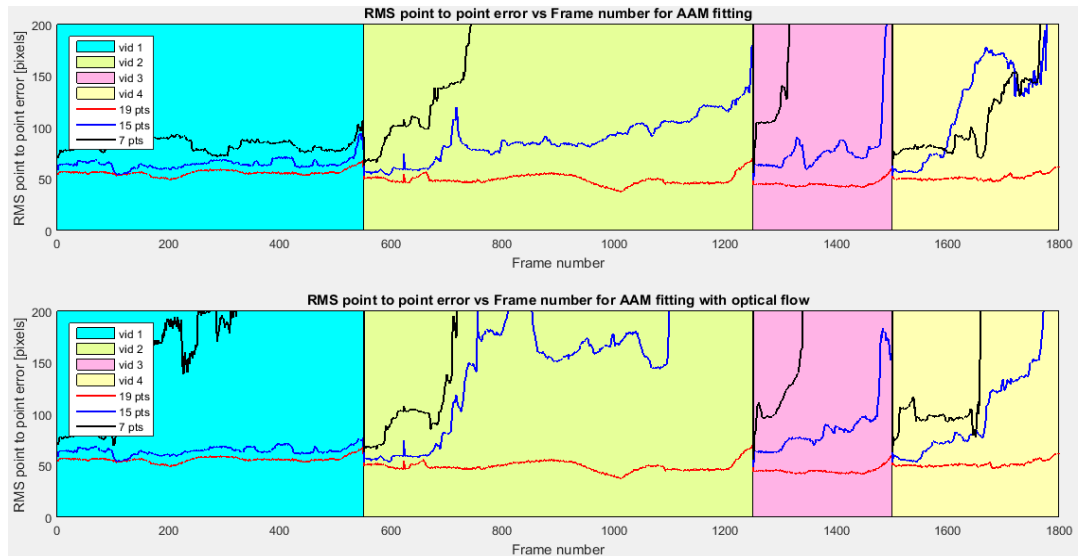


Figure 5.5. Comparison between point to point RMS errors for AAM fitting using 10 iterations and point to point RMS errors for AAM fitting with optical flow initialization using 10 iterations.

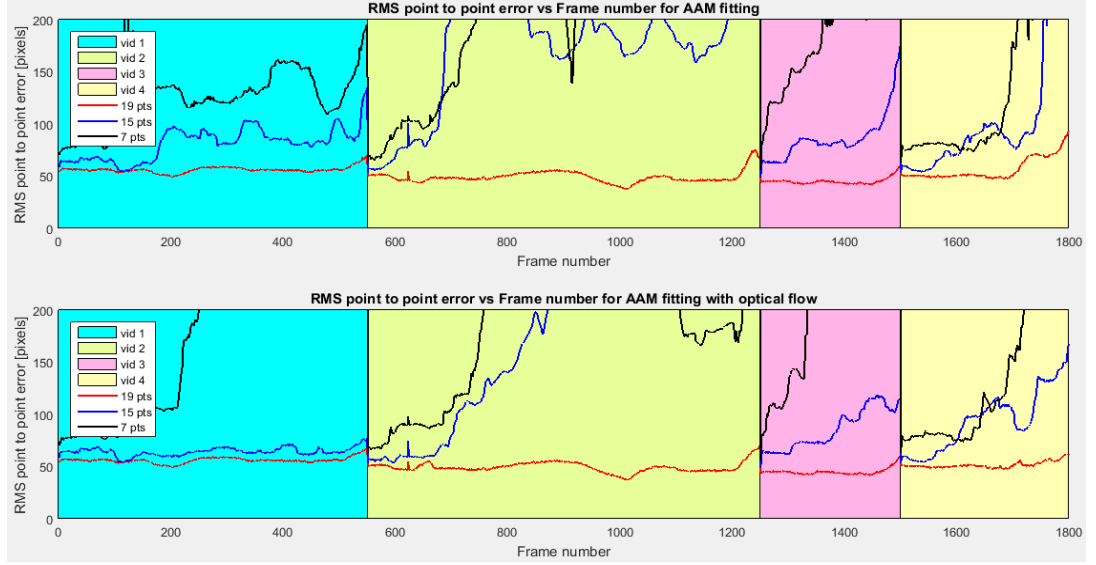


Figure 5.6. Comparison between point to point RMS errors for AAM fitting using 5 iterations and point to point RMS errors for AAM fitting with optical flow initialization using 5 iterations.

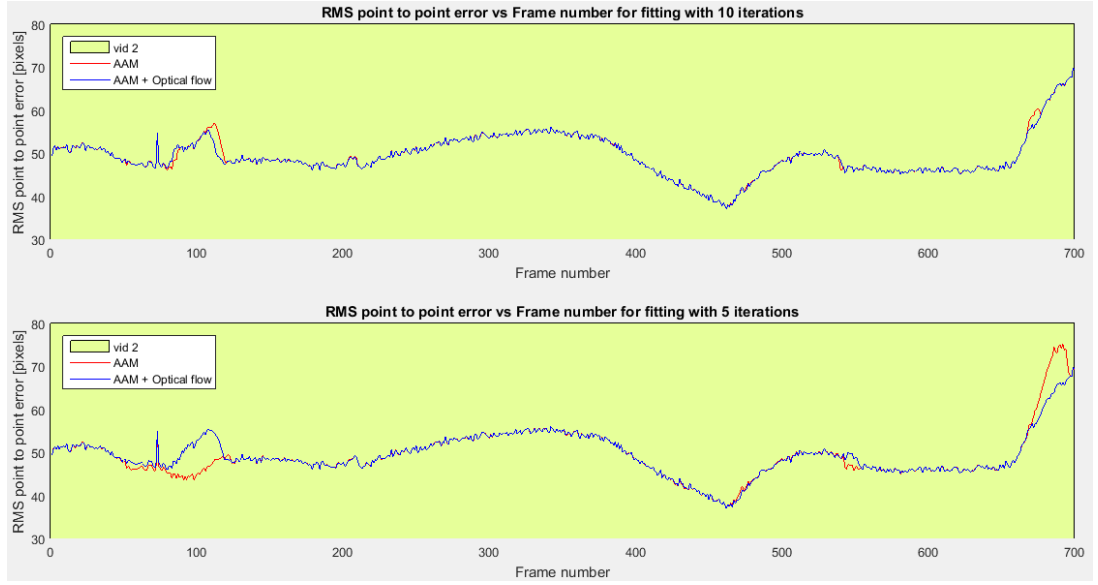


Figure 5.7. Comparison between RMS point to point errors for AAM fitting and AAM fitting with optical flow initialization, using 10 and 5 iterations each for video no. 2.

5.4.3 CLM algorithm

MATLAB implementation of the CLM algorithm given in [33] was used in the evaluation. In the original implementation, it has utilized 68 facial landmark points and is trained on the FG-NET talking face video image dataset [80]. In their work,

optimum feature point positions are found by adopting a linear SVM as used in [81] and optimization techniques as used in [82]. We further modified it by training the model on the LFPW dataset [78] using only 19 points in order to reduce the computation time. Furthermore, to improve the tracking speed, we combined the CLM fitting algorithm with the KLT tracking algorithm. In this approach, first the face was detected using the Viola-Jones algorithm and the facial model was initialized to fit inside the detected bounding box of the face. Then in the next frame, the CLM algorithm was used to correctly fit the initialized model onto the face. These correctly fitted facial landmark points by the CLM algorithm were fed into the KLT point tracker to be tracked in the subsequent frames. And whenever the tracking of a point was lost, the above process was repeated, reinitializing the model. Table 5.5 shows the FPS values obtained by evaluating the above approaches using a model of 19 facial landmark points and CLM fitting with 5 iterations in a 16×16 search area. Figure 5.8 shows the comparison of the point to point RMS error values obtained using the above two approaches.

Table 5.5: FPS for CLM algorithm and CLM Initialization + KLT Tracking with 5 iterations and 16×16 search area

Approach	FPS
CLM fitting	1.3986
CLM Initialization + KLT Tracking	40.8286

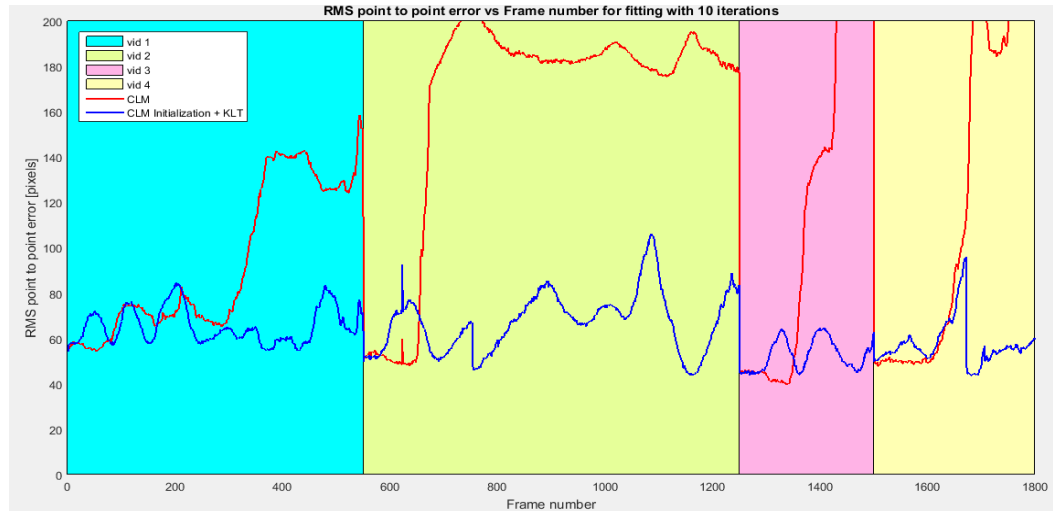


Figure 5.8: Comparison of point to point RMS error values of CLM based face tracking with 5 iterations in a 16×16 search area vs. KLT tracking with CLM initialization.

5.5 Discussion

5.5.1 Face detection algorithms

From the results given in Table 5.2, it is quite evident that the Viola-Jones face detection algorithm surpasses the other two face detection approaches both in terms of accuracy and speed with a detection rate of 98% and with an average running time of 1.1115 seconds. With respect to average time taken to process a single image, the Viola-Jones algorithm is around 4 times faster than the Neural Network based approach and around 6 times faster than the SVM based approach. It also exceeds the detection rates of the other two algorithms by more than 30% and has the least number of false positives and false negatives. Viola-Jones algorithm scoring a F_1 value close to 1 ensures that it gives the 395 optimum results when compared with that of other two algorithms. The neural network based approach can be considered as the second best approach since it has a better detection rate of 64% and an average running time of 4.2912 seconds than the SVM based approach. But still it performs poorer than the SVM approach in terms of false positive and false negative results.

5.5.2 Face tracking algorithms

5.5.2.1. KLT algorithm

The red coloured line graph which corresponds to the facial feature tracking approach in Figure 5.2 reaches negative values between frame numbers 1600 and 1800. This is because that particular approach sometimes loses track of at least one of the four facial features used. This happens often when the person in the video blinks and the pupil of the eye gets covered by the eyelid. The second approach which uses SIFT features to track the face 21 was followed to address this issue. As it can be observed in Figure 5.2, the SIFT feature tracking approach with KLT shows higher RMS error values than the facial feature tracking approach, 410 indicating a loss in accuracy. For each video, it shows a trend of increasing RMS error values as the video progresses. This could be a consequence of the accumulation of errors in the final inferred positions of the facial points in each frame occurring due to subtle errors of tracking in some of the SIFT feature points. Such errors happen when the

feature points taken as SIFT features are not good enough points with strong texture variations for the tracker to accurately track and hence large movements in the video make such points to go astray. Considering the FPS values of both the approaches, the first approach of tracking only the facial features of tracking only the facial features gives satisfactory results when compared to the second approach that uses SIFT features for the tracking.

5.5.2.2. AAM based face tracking

According to the FPS values obtained by AAM fitting alone and AAM fitting with optical flow initialization as shown in Table 5.4, it can be observed that the FPS value increases significantly with the reduction of the number of points used in building up the model. This is because as the number of points decreases, the area of the global appearance model reduces in size, thereby reducing the number of pixel comparisons that need to be done between the model and the image. But according to Figure 5.5 and Figure 5.6, as the number of points are reduced, face tracking loses early in the video frames in both the approaches. By reducing the number of iterations that is used to fit the model onto the faces in the video frames, the FPS value can be further increased. But with the reduction of the number of iterations, once again the tracking gets lost much earlier. According to Table 5.4, AAM fitting using optical flow initialization gives lower speeds than AAM fitting alone as the former requires additional processing. But according to Figure 5.7, it can be observed that in some instances AAM fitting with optical flow initialization performs better in terms of accuracy than the normal AAM fitting approach. Those are the instances where fast pose variations of the face can be observed. The reason behind this is that optical flow technique can estimate point positions up to large displacements with reasonable accuracy and running the AAM fitting afterwards gives more accurate positions of those estimated points near their neighborhood. AAM fitting alone is not capable of getting those first rough estimates of the positions of the points with large displacements. With respect to both the FPS and the RMS error values, the optimum performance can be observed with the AAM having 19 points and 5 fitting iterations. But its FPS value being 10.3240 makes it difficult to be used in an application with real-time requirements. Therefore further optimizations would be required to

improve the tracking speed. Another important observation was that in the first video which was taken in an outdoor environment with natural light, the model fitting was quite stable and resulted in low point to point RMS error values. The reason behind this can be the favourable illumination conditions in the outdoor environment.

5.5.2.3. CLM based face tracking

According to the FPS values given in Table 5.5, the CLM fitting algorithm when used alone for face tracking, performs significantly slow giving an FPS value closer to 1. But when this approach is combined with the KLT algorithm, an FPS value closer to 40 can be observed. According to Figure 5.8, in terms RMS error values, KLT feature tracking with CLM initialization performs better than the CLM taken alone.

5.6 Selection of the optimum approaches

According to the evaluation results discussed, the Viola-Jones algorithm performs better than the other two face detection algorithms in terms of accuracy as well as speed. Out of the three face tracking approaches evaluated, the optimum results were given by the facial feature tracking approach with the KLT algorithm, giving an average FPS value of 24.3 and point to point RMS error values within reasonable limits. Hence those two algorithms were used in developing the proposed application considering its requirements.

6. IMPLEMENTATION

6.1 Tools and libraries used

All implementation of the application was done using the Unity Game Engine [83]. Unity 3D is one of the world's most famous game development platforms that provide a multitude of different tools to build both 2D and 3D gaming, AR or VR applications with ease. It abstracts away a lot of low level implementation details like platform specific optimizations and deployments and lets the developers better focus on application logic without having to worry about platform specific considerations. Unity also has a very active community and a dedicated asset store that proves to be very useful in getting development issues solved quickly and easily.

To avoid reinventing the wheel, some computer vision libraries were used to facilitate the various image processing and computer vision requirements of the application. Namely, the libraries Accord.NET, EmguCV and OpenCV were tried out to choose the best candidate for the application development.

Accord.NET [84] is a machine learning framework written completely in C#. It has audio and image processing libraries built into it. By including pre-built dlls of Accord.NET inside Unity's assets folder, the functions provided by Accord.NET can be accessed from Unity scripts. Using Accord we implemented a Viola-Jones face detector that runs on the PC platform. In addition, the pair of eyes, nose and the mouth were detected by loading HaarCascades for eye pair, left and right eyes, nose and the mouth from OpenCV-compatible XML files provided at <http://alereimondo.no-ip.org/OpenCV/34>. The output obtained is shown in Figure 6.1.

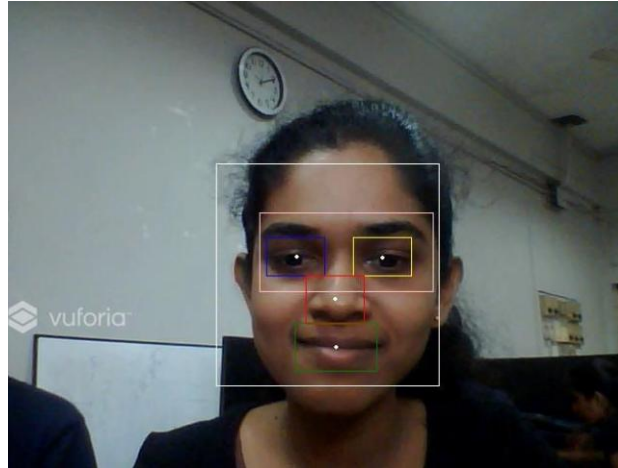


Figure 6.1. Face detection and facial feature detection outputs from Accord.NET framework integrated with Unity.

Imaging functions in Accord.NET have dependencies on the C# System.Drawing library. The issue of trying to get the face detector to run on Android was that some of the functions in System.Drawing library, which are being used by Accord did not have support on Android. Due to this lack of support, this solution had to be abandoned.

OpenCV is a famous open source computer vision library available to be used in several languages. Since development in Unity is done using C#, the freely available OpenCVSharp [85] library which is a .NET wrapper for the OpenCV library was used. The library supported all the functionality we required on the PC but failed to perform certain required operations when built and run on a mobile device. The reason behind was that mobile devices could not run the OpenCV native scripts built for PC.

Then we tried Emgu CV [86] which is also a cross platform .Net wrapper to the OpenCV image processing library. This allows OpenCV functions to be called from .NET. It is written entirely in C#. The benefit with EmguCV was that it could be compiled in Mono and therefore it was able to run on any platform that Mono supports, including iOS, Android, Windows Phone, Mac OS X and Linux. We implemented face detection with EmguCV and the results can be seen in Figure 6.2.

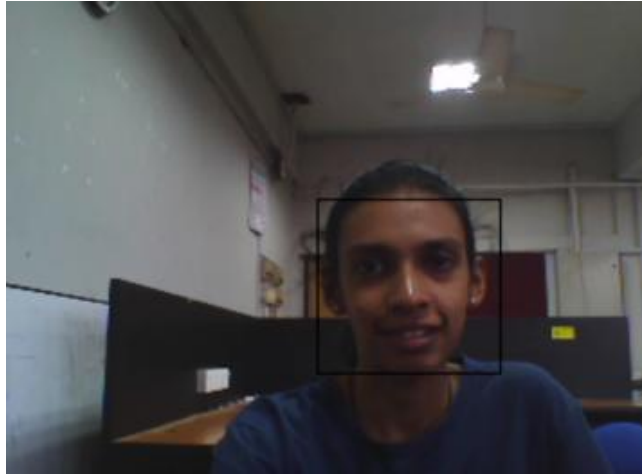


Figure 6.2. Face detection outputs from EmguCV

After researching for the better option between OpenCV and Emgu CV, we decided to proceed with OpenCV. This is because we required ultra fast and support with complex algorithms that we used, which can be readily provided by the OpenCV image processing library. On the other hand, OpenCV was cheaper than Emgucv.

Not being able to proceed with the free C# version of OpenCV, we purchased the proprietary OpenCV plugin built for Unity, OpenCVForUnity [87], which could run on any platform supported by Unity. This is the library that was finally used in the development of the final product.

6.2 Face Detection

The first step of the application process is to identify whether a human face is present in an image frame. If any human faces are not found not further processing is done.

For this purpose, the cascade detector provided in OpenCV was used along with a pre trained cascade classifier trained to identify frontal human faces. This cascade detector is built based on the famous Viola Jones face detection algorithm [2]. The input image is swept through using a window of a specified size to check for any possible human faces. If found a bounding rectangle of that face is returned.

The OpenCV API allows specifying certain parameters in running this detector and to suit the requirements of the application a window of size 90×90 pixels was used and scale factor of the image of 2 was specified.

6.3 Facial Feature Detection and Tracking

Once a face is detected, the next step is to extract the feature points on the face. Those detected feature points are then used to track the face and its pose variations.

The initial attempt was to use cascade detectors trained to detect specific parts of the face namely right eye, left eye, nose and then use those feature points to track the face using the KLT point tracker. This approach could give satisfactory results in general but the tracking of the two eye points lost its accuracy significantly when the person blinked eyes. The reason behind was that KLT point tracker makes use of the textural variations in the images and tracking the black pupil of the eye was not a problem, but as the pupil cannot be seen when the person blinks the tracked texture variation suddenly vanishes causing the tracking to lose accuracy. Accordingly it became evident that this approach is not robust enough to meet the requirements of our application.

The next attempt was to extract some good features to track from the facial region, ideally some sharp corner points, and use them to track the face. From the geometrical movement of those detected points, the positions of the four facial features two eyes, nose tip and center of mouth were estimated. This was achieved by computing the geometric transform between the matching corner point pairs in two consecutive image frames. This approach could solve the problem with the person blinking the eyes but showed a trend of reduced accuracy in the position estimates of the nose tip. Accordingly it was decided to use the normal feature tracking with the nose tip point and use the position estimates from geometrical movement of the corner points for the points of two eyes and center of mouth.

For extracting the good features to track (corner points) on face, the Shi-Tomasi feature extractor was used [88].

In determining the transformation matrix between two sets of matching points, two points from one set are chosen at random along with their corresponding points in the other set. Then those points are normalized so that their center of mass is at (0,0) and the mean distance from the center is $\sqrt{2}$. The normalized points are next used in

generating a matrix which is then subjected to single value decomposition. One of the resultant three matrices are used in the final computation of the transform matrix. This obtained matrix is used again to evaluate how close the transformed points using that matrix are to the actual points and if the results are not good enough the same process is followed a certain number of times until satisfactory results are achieved. This algorithm is based on [89] and [90].

Finally to improve the initial position extraction of the four facial features, a Constrained local model was used as described in section 6.3.1.

6.3.1. Facial feature extraction using Constrained Local Model

A joint model of a shape model and a patch model was developed to find landmarks of the main four facial feature points (left eye, right eye, tip of the nose and mouth) and some additional feature points (corner point of the left eye, corner point of the right eye and bottom most point of the mouth). The implementation of this procedure can be summarized as below.

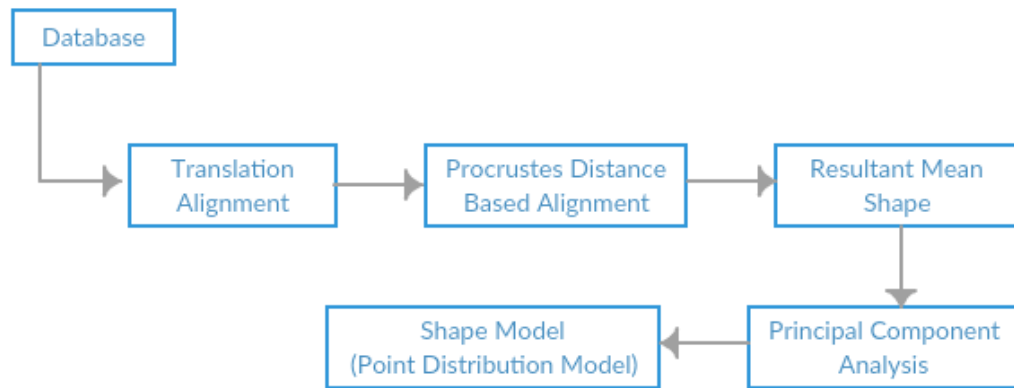


Figure 6.3. Process of facial feature extraction

This approach required a database of faces to set the landmark positions along with the faces. MUCT face database [91] that consists of 3755 faces with 76 manual landmarks was selected as the database of interest. Once data were prepared, shapes made of landmarks were aligned using Procrustes Distance based Alignment. This refers to the process of removing global rigid motion from a collection of points.

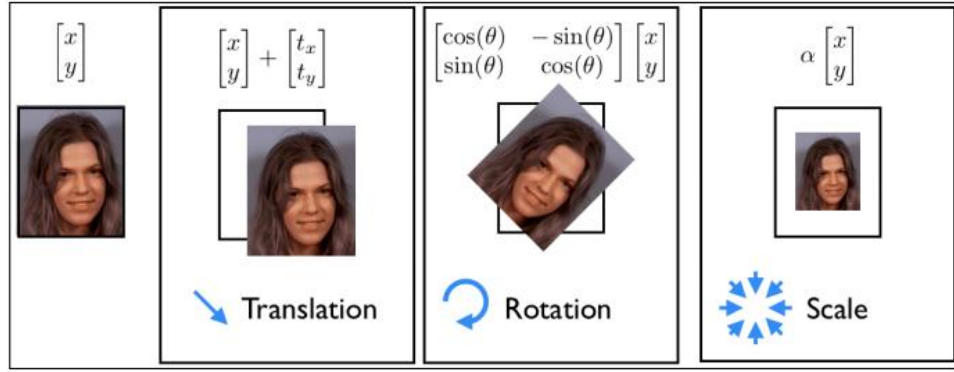


Figure 6.4. Procrustes distance based alignment [92]

After aligning the shapes, standard Principal Component Analysis (PCA) was computed to find main shape distortions. Following diagram shows an example of PCA mean at the center and main rigid basis deformations of it.

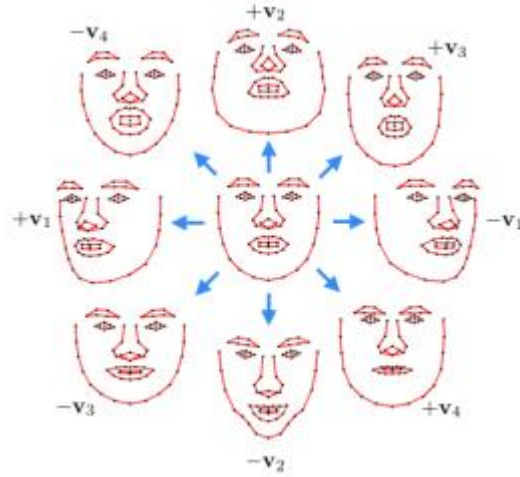


Figure 6.5. First four modes of variation [92]

The shape model analytical expression for shape 'X' can be expressed in a linear function as below.

Here 'X' refers to the mean shape which is represented by a $2n \times 1$ vector.

$$X = X' + Pb \rightarrow (6.1)$$

$$X = (x_1, \dots, x_n, y_1, \dots, y_n) \rightarrow (6.2)$$

'P' is a matrix constructed using the few first eigenvectors obtained from the Principal Component Analysis on shapes. 'b' is the corresponding mode of variation.

By varying values of b , usually between $\pm 3\sigma$, different face deformations can be modeled. This formulation is the shape model which is also known as the Point Distribution Model (PDM). That represents the mean geometry of a shape and some statistical modes of geometric variation inferred from the training set of shapes. Each mode changes the shape by moving landmarks along straight lines through mean positions. Thus, new shapes could be created by modifying mean shape with weighted sums of modes.

OpenCV implements a class for computing PCA, however, it requires the number of preserved subspace dimensions to be prespecified. As this is often difficult to determine a priori, we used a common heuristic to choose it based on the fraction of the total amount of variation it accounts for.

Then we built the facial feature extractor using a representation of a linear image patch. This representation gave reasonable estimates of facial feature locations for use in the KLT face tracking algorithm while enabling an extremely rapid evaluation that makes real time face tracking possible. Due to their representation as an image patch, the facial feature detectors were referred to as patch models. This image patch, when cross-correlated with an image region containing the facial feature, yields a strong response at the feature location and weak responses everywhere else. A visualization of this setup is shown in the following figure for the left outer eye corner.

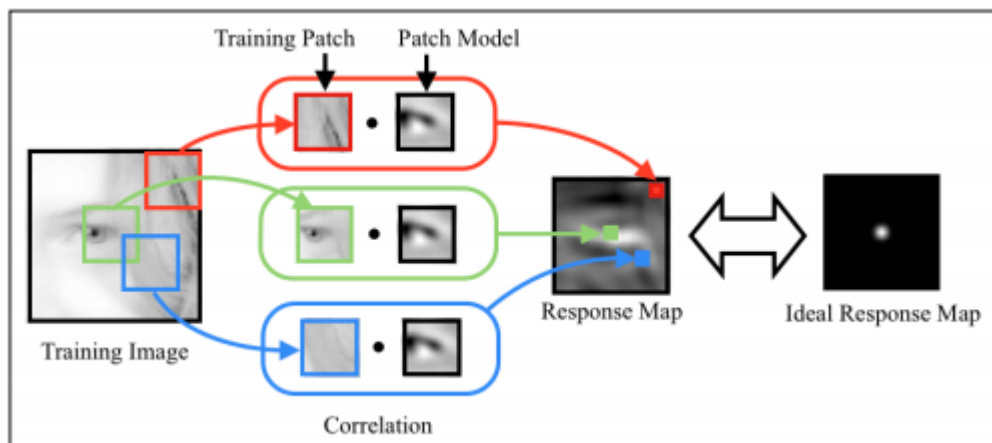


Figure 6.6: A linear image patch for left outer eye corner [92]

In this way, the landmarks of the shape model were extracted out and the main feature points were sent as inputs for the tracking process.

6.4 Head Pose Estimation

As suggested by A.H. Gee and R. Cipolla [18], head pose can be estimated geometrically by finding the facial normal of a given facial image considering the geometric distribution of the facial features. A modified version of their method is used in the application.

The roll angle of the head is computed by taking into account the angle between the line connecting the pair of eyes and the horizontal axis. The computation of the pitch angle relies on the assumption that in an average human being, the ratio between the lengths from the eye level to the base of the nose (l_f) and to the center of the mouth (l_m) stays roughly the same. This is indicated in Figure 6.7. In [18], the authors have proposed this ratio to be 0.6. And when the subject changes the pitch angle of the head, this ratio varies between 0 and 1. Hence the pitch angle can be calculated proportionately to the change in this ratio. As this ratio can vary from person to person, the exact ratio is calculated at the instance the face is detected, assuming that the person is looking straight at the camera.

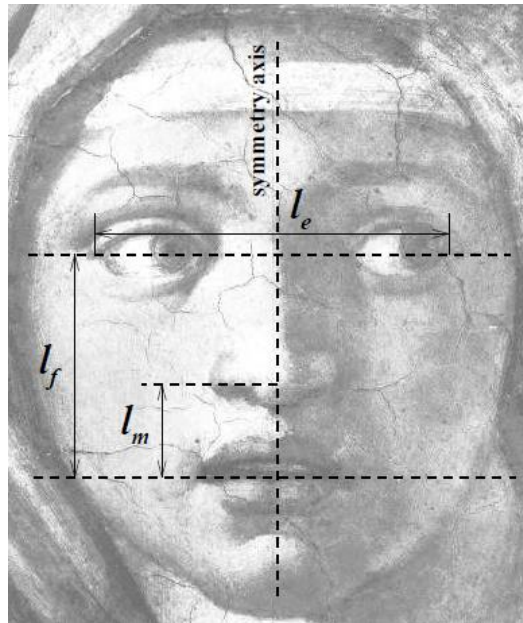


Figure 6.7: The facial model [18]

Similarly the yaw angle can be calculated proportionately to the change in the ratio between the length of the perpendicular drawn from the nose tip to the facial symmetrical axis and the length from the mid point between the eyes to the center of the mouth.

6.5 Optimizations

6.5.1 Histogram Equalization

Initially, without a proper way of handling low light conditions and irregularly illuminated faces, the application often could not detect faces and its features even in normally lit indoor environments. The first attempt at addressing this issue was to use the classical histogram equalization technique and do all the further processing of the image on the histogram equalized image. This could not show significant improvements since the classical histogram equalization follows a global approach and does not suite in instances where there is a foreground and a background with different levels of illumination. In a general usage scenario of our application, the face of the user which is close to the camera would be more lit than the background which is further away from the camera. Classical histogram equalization applied on such a scenario would improve the contrast of the background but cause the foreground to lose a majority of its features.



Figure 6.8: Global Histogram Equalization vs Contrast Limited Adaptive Histogram Equalization [93].

The Adaptive Histogram equalization technique addresses this issue by dividing the image into smaller regions of fixed size and performing the histogram equalization

locally to those regions. The downside of this approach is that any existing noise in the image also getting amplified. The Contrast Limited Adaptive Histogram Equalization (CLAHE) imposes a limit for the contrast so that it will not exceed a certain threshold and solves that issue.

So finally, it was decided to apply CLAHE to each input image frame before proceeding with further calculations. This significantly improved the sensitivity of the application and both face detection and feature tracking with decent accuracy became possible in even dimly lit environments.

6.5.2 Divergence Check

At each frame, a method is implemented to check whether the locations of the four facial feature points (the pair of eyes, tip of the nose and the center of the mouth) estimated move out of a defined area bounding the face. A face bounding rectangle is obtained at the first instance when a face is detected by the Viola-Jones face detector. Then the corner points of this rectangle is transformed at each frame using the transformation matrix calculated by taking into account the movement of salient feature points of the detected face in the video stream. This gives corner points of a convex polygon which bounds the actual face region. Then the following procedure is implemented to check whether a given facial feature point lies inside this convex polygon or not.

First vectors are drawn from the feature point to each of the vertices of the bounding polygon. Then the angle between consecutive vectors is calculated. The addition of the angles is equal to 360° if the point lies inside the polygon, otherwise the addition is equal to 0° . These two situations are depicted in Figure 6.9 and Figure 6.10.

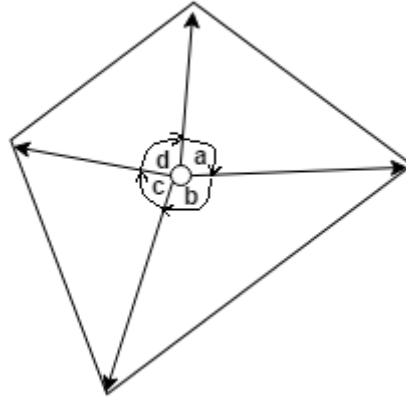


Figure 6.9: The point is inside the polygon

When the point is inside the polygon, $a + b + c + d = 360^{\circ}$.

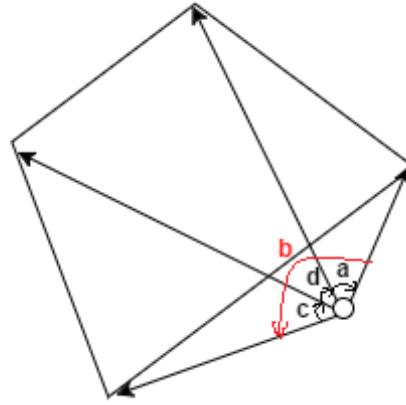


Figure 6.10: The point is outside the polygon

When the point is outside the polygon, $a + b + c + d \neq 360^{\circ}$.

In case if a particular feature point going beyond the bounding polygon which leads to a divergence of the feature points, the entire algorithm is reinitialized preventing the virtual object getting rendered in the incorrect location, scale and orientation.

6.6 Graphical user interface of the mobile application

The graphical user interface of the main mobile application was designed using Unity. Unity provides WebCamTexture class that starts the camera to get the live video feed. The set of 3D models of spectacles are displayed on the main interface of the application. User can select a preferred model and try it in real time.

The available 3D models of the accessories are stored in the 'Resources' folder of the project. That contains the 3D models as well as the images of the spectacles. These images can come in different file formats like png, jpeg and jpg. They are used to display a 2D view of the spectacle to the user through the main user interface. The 3D models, which can be developed using different 3D model building software like 3DSMax can come in different file types like fbx, obj, 3ds etc. Each model comprises 'Materials' and 'info' file as shown in Figure 6.11. Materials are useful to quickly change the appearance of the objects to look much more realistic. Info file is a JSON file which contains information about the 3D model, the distance between the pair of glasses in pixels, it's original rotation, the position it should be originally placed when dynamically adding them onto the scene.

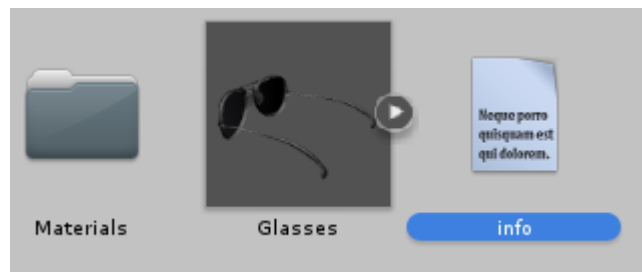


Figure 6.11: Assets of the 3D model

Each model is displayed to the user as a button with its corresponding image. Once the user selects a model by clicking the button, a 3D object of that model is dynamically added as a child object under a dummy parent object which is originally pivoted at position (0,0,0) and rendered to the canvas. First of all, Unity's Resources.Load function is used to load the asset stored at path in the Resources folder. Then the relevant parameters of the model are read from the info.json file and the positioning of the model is adjusted accordingly. Finally, the model is successfully added to the parent game object.

Addition of the 3D models for an empty parent object was developed as a workaround for the pivot position of the downloaded .fbx or .obj models being inconsistent. The model's pivot position inside the parent object is set to a little below the bridge of the glasses and included in the info.json file. This helps in correctly placing the glasses at mid of the line connecting the two eyes.

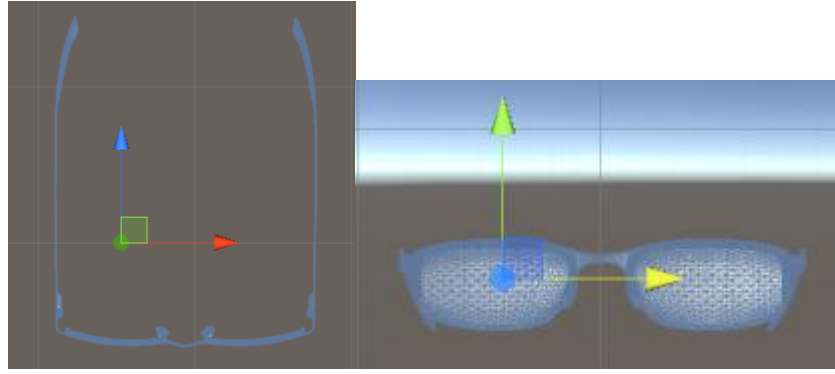


Figure 6.12: Pivot position of an imported 3D model

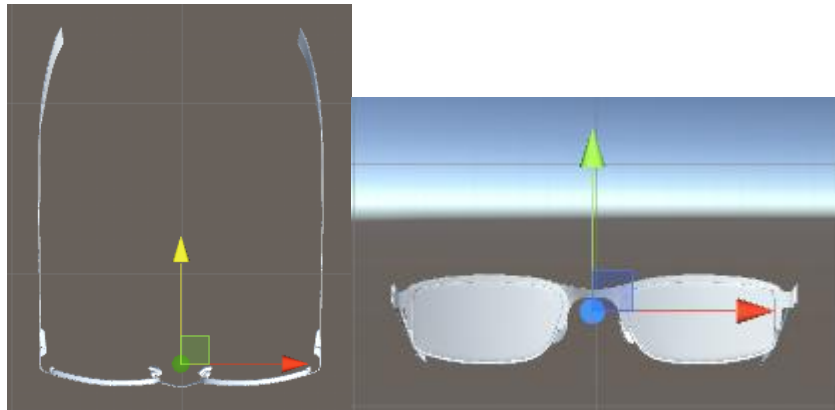


Figure 6.13: Correct pivot position after adding the 3D model to dummy parent object

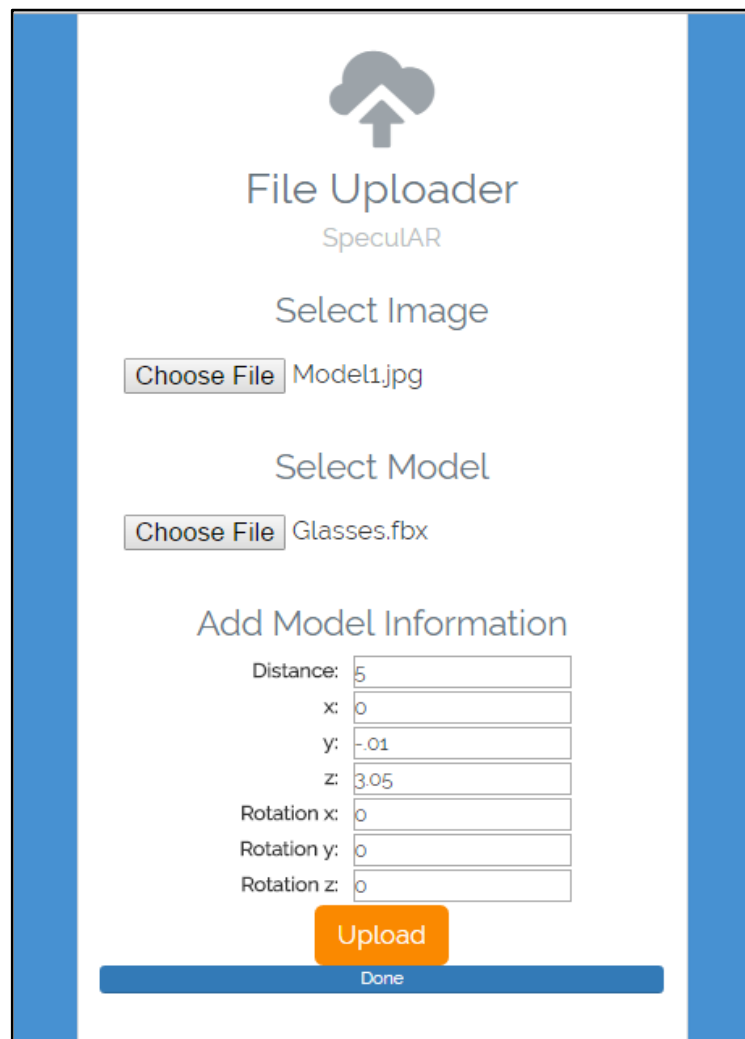
Unity's OnGUI function is used for handling the GUI events. A scroll view is added to the GUI in case the screen width is not sufficient to add buttons for all available 3D models of accessories. Unity's GUI functions are used to generate the buttons. A custom UnityGUI ScrollView was developed to facilitate dragging the list itself up and down with the finger. OnGUI code draws the scrolling list but no longer responds to clicks. The code for detecting and responding to touches is added to FixedUpdate method.

Another button control called 'Re-init' is also included in the main interface to restart the whole process of face detection, tracking and pose estimation.

6.7 Connection to the server and back end

The application that runs on the mobile phone has to be connected to a business client's server from where the newest models of products can be downloaded and the application has to be updated. A file uploader that uses AJAX was built to upload

the 3D models of eye wear to the backend server which maintains a repository of 3D models. It consist of a simple HTML + JavaScript front end to allow the user to select the 3D model and an image that is used for display purposes. Also the details such as the distance between the two glasses, coordinates required to adjust pivot position and the original rotation of the model are requested from the user to create the info.json file which is used when retrieving initial details of the 3D model when rendering it in the application. The file upload sent via the HTML form are handled using the NodeJS backend.



The image shows a web-based user interface for uploading 3D models. It has a clean, modern design with a white background and blue accents. The interface is divided into three main sections: 'Select Image', 'Select Model', and 'Add Model Information'. Each section has a 'Choose File' button and a text input field. The 'Add Model Information' section contains several input fields for numerical values. At the bottom, there is an orange 'Upload' button and a blue 'Done' button.

File Uploader
SpecuAR

Select Image

Choose File Model1.jpg

Select Model

Choose File Glasses.fbx

Add Model Information

Distance: 5

x: 0

y: -0.1

z: 3.05

Rotation x: 0

Rotation y: 0

Rotation z: 0

Upload

Done

Figure 6.14. User interface for uploading 3D Models to the server

As future work, the connection to the server from the Unity application has to be implemented. Then the user will be able to get the latest models of eyewear dynamically from the server.

7. EVALUATION AND RESULTS

The accuracy of the estimated locations of the facial features and the head pose were evaluated against ground truth data obtained from the GI4E head pose database [94] that consists of videos obtained with a standard webcam for testing head tracking and pose estimation. The GI4E database consists of 120 videos corresponding to 10 different subjects (6 males and 4 females) and 12 videos each. Every set of 12 videos is composed of 6 guided-movement sequences and 6 free-movement sequences. Data containing in the GI4E head pose database was found to be better suited for testing our application than the other available test suites (e.g. UUI head pose database [95], PRIMA head pose image database [96]) as it contains videos not just still images. The subjects in those videos were also captured with close proximity to the camera which is the ideal distance that the users of the developed application would use it. Figure 7.1 shows some sample video frames with significant variation in roll, yaw and pitch angles taken from the GI4E database.



Figure 7.1: Sample video frames taken from the database.

The point to point root-mean-square (RMS) error was taken as a measure to determine the accuracy of the estimated locations of the four facial features, the pair of eyes, nose tip and the mouth. Figure 7.2 shows the point to point RMS errors obtained for five different videos of subject 1 in the GI4E database. The five videos were chosen to represent translation, roll, pitch, yaw and the scaling of the subject's head as indicated by different background colors in the graphs.



Figure 7.2: Point to point RMS errors between the ground truth and the estimated facial feature points

The actual and the estimated values of the head pose in terms of roll, yaw and pitch were also plotted as denoted in Figure 7.3 to visualize the deviation of the estimated roll, yaw and pitch angles from the ground truth angles in the GI4E database.

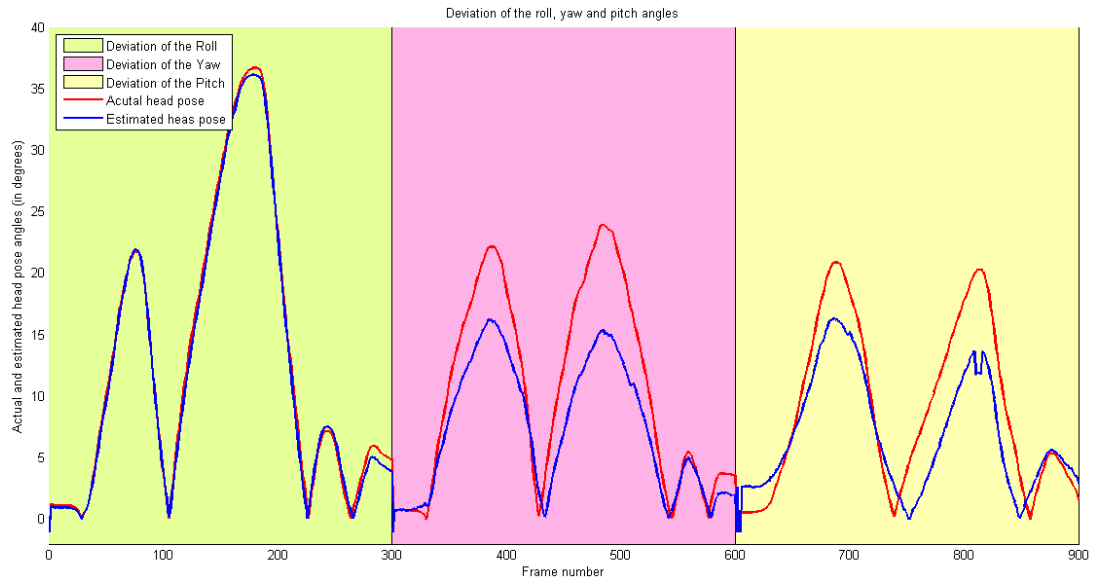


Figure 7.3: Deviation of the roll, yaw and pitch angles

According to the results obtained, the point to point RMS error between the actual and the estimated facial feature points fluctuates between 2 and 8 pixel lengths which indicates that the facial feature point estimations are quite accurate.

When considering the head pose, there exists only minor deviations between the actual and the estimated values of the roll and the yaw angles. But quite significant deviation can be noted in terms of the pitch angle. The deviation between the actual and the estimated values of yaw become larger as the yaw angle takes higher values. These can be due to the fact that the tracking accuracy of the tip of the nose drops as the yaw and pitch angles increase significantly. In such cases, the estimated nose tip usually appears a bit closer to the center of the face away from the actual nose tip, giving a value a bit less than the actual yaw.

8. DISCUSSION AND CONCLUSION

In this report we discussed about a mobile augmented reality application which uses face detection, face tracking and head pose estimation techniques to estimate the pose of a virtual object to be realistically placed on a human face in real time. This application would enhance the online shopping experience of customers as they can virtually try out models of eyewear before actually purchasing them. Augmented reality can be used in the ecommerce industry to enable users to virtually visualize themselves wearing models of fashionable facial accessories. The specifics of the application development, its architecture and implementation methodology were also discussed.

We also looked at several face detection and tracking algorithms on whether they can be utilized in achieving the real-time and accuracy requirements of the proposed application. The Viola-Jones algorithm, a neural network based algorithm and a SVM based algorithm were considered under face detection. Under face tracking, feature based KLT tracking algorithm and the model based AAM and CLM algorithms were considered. We also presented the results of a comprehensive evaluation that was carried out to select the optimum algorithms out of them to be used in the proposed application. The results of the evaluation can be of great use for the developers of similar AR applications that involves face detection and tracking. We also discussed how human head pose can be derived by geometric means once the position estimates of facial features in each video frame are obtained through a face tracking algorithm.

As future work, to further enhance the online shopping experience of customers, an online ordering option can be integrated to the proposed application making it easier for the customers to buy the items that they try out via the same application. The application can also be used in optical showrooms enabling the customers to try on virtual models of eyewear. This can prevent the physical damages that can happen to eyewear when customers physically handle and try them on.

REFERENCES

- [1] LiveRoom. Retrieved May 24, 2016, from <https://www.liveroom.xyz>
- [2] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-511). IEEE.
- [3] Freund, Y., & Schapire, R. E. (1995, March). A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23-37). Springer Berlin Heidelberg.
- [4] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1), 71-86.
- [5] Hannuksela, J. (2003). Facial feature based head tracking and pose estimation. *Department of Electrical and Information Engineering, University of Oulu, Finland*.
- [6] Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1), 23-38.
- [7] Osuna, E., Freund, R., & Girosit, F. (1997, June). Training support vector machines: an application to face detection. In *Computer vision and pattern recognition, 1997. Proceedings., 1997 IEEE computer society conference on* (pp. 130-136). IEEE.
- [8] Kawaguchi, T., & Rizon, M. (2003). Iris detection using intensity and edge information. *Pattern Recognition*, 36(2), 549-562.
- [9] Soriano, M., Martinkauppi, B., Huovinen, S., & Laaksonen, M. (2003). Adaptive skin color modeling using the skin locus for selecting training pixels. *Pattern Recognition*, 36(3), 681-690.
- [10] Hsu, R. L., Abdel-Mottaleb, M., & Jain, A. K. (2001). Face detection in color

images. In *Image Processing, 2001. Proceedings. 2001 International Conference on* (Vol. 1, pp. 1046-1049). IEEE.

[11] Hjelmås, E., & Low, B. K. (2001). Face detection: A survey. *Computer vision and image understanding*, 83(3), 236-274.

[12] Marr, D., & Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences*, 207(1167), 187-217.

[13] Yow, K. C., & Cipolla, R. (1997). Feature-based human face detection. *Image and vision computing*, 15(9), 713-735.

[14] Sobottka, K., & Pitas, I. (1998). A novel method for automatic face segmentation, facial feature extraction and tracking. *Signal processing: Image communication*, 12(3), 263-281.

[15] Sirohey, S. A., & Rosenfeld, A. (2001). Eye detection in a face image using linear and nonlinear filters. *Pattern recognition*, 34(7), 1367-1391.

[16] Brunelli, R., & Poggio, T. (1993). Face recognition: Features versus templates. *IEEE transactions on pattern analysis and machine intelligence*, 15(10), 1042-1052.

[17] Chow, G., & Li, X. (1993). Towards a system for automatic facial feature detection. *Pattern Recognition*, 26(12), 1739-1755.

[18] Gee, A., & Cipolla, R. (1994). Determining the gaze of faces in images. *Image and Vision Computing*, 12(10), 639-647.

[19] Wagener, D. W., & Herbst, B. (2001). Face Tracking: An Implementation of the Kanade-Lucas-Tomasi Tracking Algorithm. *PRASA, South Africa*.

[20] Cootes, T. F., Taylor, C. J., Cooper, D. H., & Graham, J. (1995). Active shape models-their training and application. *Computer vision and image understanding*, 61(1), 38-59.

[21] Milborrow, S., & Nicolls, F. (2008, October). Locating facial features with an

extended active shape model. In *European conference on computer vision*(pp. 504-513). Springer Berlin Heidelberg.

[22] Jesorsky, O., Kirchberg, K. J., & Frischholz, R. W. (2001, June). Robust face detection using the hausdorff distance. In *International Conference on Audio-and Video-Based Biometric Person Authentication* (pp. 90-95). Springer Berlin Heidelberg.

[23] Cootes, T. F., Edwards, G. J., & Taylor, C. J. (2001). Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6), 681-685.

[24] Tzimiropoulos, G., Alabort-i-Medina, J., Zafeiriou, S., & Pantic, M. (2012, November). Generic active appearance models revisited. In *Asian Conference on Computer Vision* (pp. 650-663). Springer Berlin Heidelberg.

[25] Gross, R., Matthews, I., & Baker, S. (2005). Generic vs. person specific active appearance models. *Image and Vision Computing*, 23(12), 1080-1093.

[26] Baker, S., & Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3), 221-255.

[27] Cootes, T. F., Edwards, G. J., & Taylor, C. J. (1999, September). Comparing Active Shape Models with Active Appearance Models. In *BMVC* (Vol. 99, No. 1, pp. 173-182).

[28] Liu, X., Wheeler, F. W., & Tu, P. H. (2007, September). Improved Face Model Fitting on Video Sequences. In *BMVC* (pp. 1-10).

[29] Cristinacce, D., & Cootes, T. F. (2006, September). Feature Detection and Tracking with Constrained Local Models. In *BMVC* (Vol. 1, No. 2, p. 3).

[30] Saragih, J. M., Lucey, S., & Cohn, J. F. (2011). Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision*, 91(2), 200-215.

[31] Asthana, A., Zafeiriou, S., Cheng, S., & Pantic, M. (2013). Robust

discriminative response map fitting with constrained local models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3444-3451).

[32] Unzueta, L., Pimenta, W., Goenetxea, J., Santos, L. P., & Dornaika, F. (2014). Efficient generic face model fitting to images and videos. *Image and Vision Computing*, 32(5), 321-334.

[33] Constrained local model (CLM) implementation - Xiaoguang Yan's Webpage. Retrieved July 29, 2016, from <https://sites.google.com/site/xgyanhome/home/projects/clm-implementation>

[34] Decarlo, D., & Metaxas, D. (2000). Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2), 99-127.

[35] Xiao, J., Moriyama, T., Kanade, T., & Cohn, J. F. (2003). Robust full- motion recovery of head by dynamic templates and re- registration techniques. *International Journal of Imaging Systems and Technology*, 13(1), 85-94.

[36] Dementhon, D. F., & Davis, L. S. (1995). Model-based object pose in 25 lines of code. *International journal of computer vision*, 15(1-2), 123-141.

[37] Gui, Z., & Zhang, C. (2006, November). 3D head pose estimation using non-rigid structure-from-motion and point correspondence. In *TENCON 2006-2006 IEEE Region 10 Conference* (pp. 1-3). IEEE.

[38] Tomasi, C., & Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2), 137-154.

[39] Cootes, T. F., Wheeler, G. V., Walker, K. N., & Taylor, C. J. (2002). View-based active appearance models. *Image and vision computing*, 20(9), 657-664.

[40] Aghajanian, J., & Prince, S. (2009, September). Face Pose Estimation in Uncontrolled Environments. In *BMVC* (Vol. 1, No. 2, p. 3).

[41] Jiménez, P., Nuevo, J., Bergasa, L. M., & Sotelo, M. A. (2009). Face tracking

and pose estimation with automatic three-dimensional model construction. *IET Computer Vision*, 3(2), 93-102.

[42] Ng, J., & Gong, S. (1999). Multi-view face detection and pose estimation using a composite support vector machine across the view sphere. In *In Proc. IEEE International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*.

[43] Sherrah, J., Gong, S., & Ong, E. J. (1999, September). Understanding Pose Discrimination in Similarity Space. In *BMVC* (pp. 1-10).

[44] Sherrah, J., Gong, S., & Ong, E. J. (2001). Face distributions in similarity space under varying head pose. *Image and Vision Computing*, 19(12), 807-819.

[45] Huang, J., Shao, X., & Wechsler, H. (1998, August). Face pose discrimination using support vector machines (SVM). In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on* (Vol. 1, pp. 154-156). IEEE.

[46] Li, Y., Gong, S., & Liddell, H. (2000). Support vector regression and classification based multi-view face detection and recognition. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on* (pp. 300-305). IEEE.

[47] Murphy-Chutorian, E., Doshi, A., & Trivedi, M. M. (2007, September). Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *2007 IEEE Intelligent Transportation Systems Conference* (pp. 709-714). IEEE.

[48] Ma, Y., Konishi, Y., Kinoshita, K., Lao, S., & Kawade, M. (2006, August). Sparse bayesian regression for head pose estimation. In *18th International Conference on Pattern Recognition (ICPR'06)* (Vol. 3, pp. 507-510). IEEE.

[49] Moon, H., & Miller, M. L. (2004, October). Estimating facial pose from a sparse representation [face recognition applications]. In *Image Processing, 2004. ICIP'04. 2004 International Conference on* (Vol. 1, pp. 75-78). IEEE.

- [50] Zhao, L., Pingali, G., & Carlbom, I. (2002). Real-time head orientation estimation using neural networks. In *Image Processing. 2002. Proceedings. 2002 International Conference on* (Vol. 1, pp. I-297). IEEE.
- [51] Rae, R., & Ritter, H. J. (1998). Recognition of human head orientation based on artificial neural networks. *IEEE Transactions on neural networks*, 9(2), 257-265.
- [52] Krüger, V., & Sommer, G. (2002). Gabor wavelet networks for efficient head pose estimation. *Image and vision computing*, 20(9), 665-672.
- [53] Osadchy, M., Le Cun, Y., & Miller, M. L. (2006). Synergistic face detection and pose estimation with energy-based models. In *Toward Category-Level Object Recognition* (pp. 196-206). Springer Berlin Heidelberg.
- [54] Wu, J., & Trivedi, M. M. (2008). A two-stage head pose estimation framework and evaluation. *Pattern Recognition*, 41(3), 1138-1158.
- [55] Hu, Y., Chen, L., Zhou, Y., & Zhang, H. (2004, May). Estimating face pose by facial asymmetry and geometry. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on* (pp. 651-656). IEEE.
- [56] Wu, J., Pedersen, J. M., Putthividhya, D., Norgaard, D., & Trivedi, M. M. (2004, August). A two-level pose estimation framework using majority voting of gabor wavelets and bunch graph analysis. In *Proc. Pointing 2004 Workshop: Visual Observation of Deictic Gestures* (pp. 4-12).
- [57] Zhu, Y., & Fujimura, K. (2004, June). Head pose estimation for driver monitoring. In *Intelligent Vehicles Symposium, 2004 IEEE* (pp. 501-506). IEEE.
- [58] Morency, L. P., Sundberg, P., & Darrell, T. (2003, October). Pose estimation using 3d view-based eigenspaces. In *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on* (pp. 45-52). IEEE.
- [59] Morency, L. P., Rahimi, A., & Darrell, T. (2003, June). Adaptive view-based appearance models. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (Vol. 1, pp. I-803). IEEE.

- [60] Mobile vision | Google developers. Retrieved July 29, 2016, from <https://developers.google.com/vision>
- [61] Face detection concepts overview. (2016). Retrieved July 29, 2016, from <https://developers.google.com/vision/face-detection-concepts>
- [62] Get started with the mobile vision API. (2016). Retrieved July 29, 2016, from <https://developers.google.com/vision/getting-started>
- [63] Visage Technologies. (2016). Face tracking and analysis. Retrieved July 29, 2016, from <https://visagetechnologies.com>
- [64] Visage Technologies. (2016). FaceTrack. Retrieved July 29, 2016, from <https://visagetechnologies.com/products-and-services/visagesdk/facetrack>
- [65] Visage Technologies. (2016). FaceTrack. Retrieved July 29, 2016, from <https://visagetechnologies.com/products-and-services/visagesdk/facetrack/#outputs>
- [66] Visage Technologies. (2016). HeadTrack. Retrieved July 29, 2016, from <https://visagetechnologies.com/products-and-services/visagesdk/headtrack>
- [67] Visage Technologies. (2016). FaceDetect. Retrieved July 27, 2016, from <https://visagetechnologies.com/products-and-services/visagesdk/facedetect>
- [68] Visage Technologies. (2016). FaceAnalysis. Retrieved July 29, 2016, from <https://visagetechnologies.com/products-and-services/visagesdk/faceanalysis>
- [69] OpenCV. (2016). ABOUT. Retrieved July 29, 2016, from <http://opencv.org/about.html>
- [70] OpenCV. (2016). PLATFORMS. Retrieved July 29, 2016, from <http://opencv.org/platforms.html>
- [71] MSQRD. (2015). Masquerade. Retrieved July 29, 2016, from <http://msqrd.me>
- [72] Acep TryLive. (2015). Solutions overview. Retrieved July 29, 2016, from <http://www.trylive.com/solutions>

- [73] Stage3D (2016). . In Wikipedia. Retrieved from <https://en.wikipedia.org/wiki/Stage3D>
- [74] Snapchat. Retrieved July 29, 2016, from <https://www.snapchat.com>
- [75] VoiceTube. Printing mode how Snapchat's filters work. Retrieved July 29, 2016, from <https://www.voicetube.com/videos/print/40224>
- [76] Vedaldi, A., & Fulkerson, B. (2011, December 24). VLFeat - home. Retrieved December 2, 2016, from <http://robots.princeton.edu/pvt/SiftFu/SiftFu/SIFTransac/vlfeat/doc/index.html>
- [77] Tzimiropoulos, G., & Pantic, M. (2013). Optimization problems for fast aam fitting in-the-wild. In Proceedings of the IEEE international conference on computer vision (pp. 593-600).
- [78] Belhumeur, P. N., Jacobs, D. W., Kriegman, D. J., & Kumar, N. (2013). Localizing parts of faces using a consensus of exemplars. IEEE transactions on pattern analysis and machine intelligence, 35(12), 2930-2940.
- [79] Test videos - Google drive. (2016). Retrieved December 10, 2016, from https://drive.google.com/drive/folders/0B8Taw5_UyILGSmRjX2VkVHNxNWM?usp=sharing
- [80] Talking Face Video. (2017). Www-prima.inrialpes.fr. Retrieved 28 March 2017, from http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html
- [81] Wang, Y., Lucey, S., & Cohn, J. F. (2008, June). Enforcing convexity for improved alignment with constrained local models. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on (pp. 1-8). IEEE.
- [82] Cristinacce, D., & Cootes, T. (2008). Automatic feature localisation with constrained local models. Pattern Recognition, 41(10), 3054-3067.
- [83] Unity - game engine. (2016). Retrieved October 2, 2010, from Unity Technologies, <https://unity3d.com>

- [84] Accord.NET machine learning framework. (2008). Retrieved October 2, 2016, from <http://accord-framework.net>
- [85] Shimat. (2016). Shimat/opencvsharp: .NET framework wrapper for OpenCV. Retrieved December 3, 2016, from <https://github.com/shimat/opencvsharp>
- [86] Emgu CV: OpenCV in .NET (C#, VB, C++ and more). (2017). Emgu.com. Retrieved 27 March 2017, from http://www.emgu.com/wiki/index.php/Main_Page
- [87] OpenCV for Unity. (2016). Retrieved December 10, 2016, from Enox Software, <https://enoxsoftware.com/opencvforunity>
- [88] Shi, J. & Tomasi, C. (1994, June). Good features to track. In Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on (pp. 593-600). IEEE.
- [89] Hartley, R., & Zisserman, A. (2003). Multiple view geometry in computer vision. Cambridge university press.
- [90] Torr, P. H., & Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding, 78(1), 138-156.
- [91] The MUCT Face Database. (2017). Milbo.org. Retrieved 27 March 2017, from <http://www.milbo.org/muct>
- [92] Emami, S., Levgen, K., Mahmood, N., Saragih, J., Shilkrot, R., Lelis Baggio, D., & Milan Escrivá, D. Mastering OpenCV with Practical Computer Vision Projects (1st ed.). Birmingham, UK: PACKT.
- [93] OpenCV: Histograms - 2: Histogram equalization. Retrieved January 20, 2017, from http://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html
- [94] Ariz, M., Bengoechea, J. J., Villanueva, A., & Cabeza, R. (2016). A novel 2D/3D database with automatic face annotation for head tracking and pose estimation. Computer Vision and Image Understanding, 148, 201-210.

[95] uulm image and video datasets. Retrieved January 10, 2017, from Institut für Neuroinformatik, <http://www.uni-ulm.de/in/neuroinformatik/mitarbeiter/g-layher/uulm-image-and-video-datasets/>

[96] Gourier, N., Hall, D., & Crowley, J. L. (2004, August). Estimating face orientation from robust detection of salient facial structures. In FG Net Workshop on Visual Observation of Deictic Gestures (Vol. 6).

APPENDIX - A: SUMMARY OF FACE DETECTION TECHNIQUES

Approaches	Techniques	Experimental results
Feature based	Viola Jones object detection framework	Viola Jones object detection framework has shown a detection rate of 95%. It requires less computational power and is significantly faster than other similar face detection algorithms. The performance of face detection using Eigenfaces degrades with changes in light, changes in size of the head and orientation. Skin based face detection requires less computational power, but it lacks robustness.
	Face detection using Eigenfaces	
	Skin based face detection	
Training based	Neural Networks	Training based approaches are not as fast as feature based approaches and they require more computational power. Neural network methods have shown a detection rate between 77.9% and 90.3% in a set of 130 test images, with an acceptable number of false detections. SVM has shown a detection rate of 97.1% and 74.2% for two different test sets, one comprising of 313 high-quality images with one face per image and the other comprising of 23 images of mixed quality with a total of 155 faces.
	Support Vector Machines	
Facial feature extraction	Low level methods	
	Color based	Color based technique has shown around 80% average accuracy with high computational time. Projections have shown a detection rate varying between 74% and 97% for different facial features and requires relatively low computational power. Gabor filters have shown a detection rate of between 80% and 95% but still requires a relatively high computational power.
	Edge detection based	
	Projections	
	Gabor filters	

	Template based methods	
	Correlation	These approaches have shown better accuracy than low level feature based methods. Still they consumed high computation time.
	Hough transform	

APPENDIX - B: SUMMARY OF FACE TRACKING TECHNIQUES

Approaches	Techniques	Experimental results
Feature based	Kanade-Lucas-Tomasi algorithm	KLT algorithm has shown remarkably robust and accurate results with limited computational power.
Model based	Active Shape Models	Model based approaches have produced significantly accurate results in facial fitting. ASM, AAM and CLM have shown the ability to robustly track human faces. ASM approach has shown faster results than the AAM approach. Deformable 3D face models have shown significant performance under challenging illumination conditions and on devices with low hardware specifications. The 3D motion recovery with dynamic templates and re-registration techniques have shown 98-100% accuracy for blink recognition.
	Active Appearance Models	
	Constrained Local Models	
	Deformable 3D face models	
	Optical flow constraints on deformable models	
	Dynamic Templates and Re-Registration Techniques	

APPENDIX - C: SUMMARY OF HEAD POSE ESTIMATION TECHNIQUES

Approaches	Techniques	Experimental results
Model based	POSIT	Experimental results have shown that POSIT converges to accurate pose measurements in a few iterations with low computational cost. Non-rigid structure from motion and point correspondence technique has shown better results in yaw and pitch information. View based AAMs have yielded good estimates of the head pose in most of the cases. The combined approach of automatic 3D model construction using SMAT, RANSAC and POSIT have correctly estimated head rotations up to $\pm 45^\circ$ while serving in real time.
	Non-rigid Structure From Motion and Point Correspondence	
	View Based AAMs	
	Patch based pose estimation with continuous regression	
	Automatic 3D model construction using SMAT, RANSAC and POSIT	
Feature based	Geometric distribution	This technique has shown accurate results with a minimum amount of computation power.
Appearance based	Template matching	
	SVM	SVM technique is not computationally attractive for real time use. Head pose estimation from prototype images still requires further improvements to facilitate good performance in real time use.
	Head pose estimation from prototype images	
	Nonlinear regression methods	
	SVR	Experiments on SVR have shown a detection rate of above 95%, recognition accuracy of above 90% and average pose estimation error around 10° . This required low computational power and memory. SVR with Localized Gradient Histograms has exhibited a significant improvement in estimation accuracy. The sparse representation combined with SVR has also shown promising results in estimating the pose more quickly and accurately. Neural networks approach requires more computational power and it is quite slow. GWNs have shown real time
	SVR with Localized Gradient Histograms	
	SVR with sparse representations	
	Neural networks	
	Gabor Wavelet Networks	
	Convolutional networks	

		performance. The use of a convolutional network has made pose estimation fast and it has shown accuracy for variations in yaw up to $\pm 90^\circ$, rotation up to $\pm 45^\circ$, and pitch up to $\pm 60^\circ$.
	Manifold embedding methods	
	Subspace analysis with the topography method	Subspace analysis with the topography method has utilized a two stage algorithm to lower the computational cost and achieve higher performance gain.
Hybrid approaches		Hybrid approaches have shown promising results for estimating the pose robustly and accurately in real-time by combining the benefits of different kinds of techniques.