UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

AASIM KHURSHID

Adaptive Face Tracking Based on Online Learning

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Jacob Scharcanski

Porto Alegre
December 6, 2018

CIP — CATALOGING-IN-PUBLICATION

Khurshid, Aasim

Adaptive Face Tracking Based on Online Learning / Aasim Khurshid. – Porto Alegre: PPGC da UFRGS,

.

116 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS,

. Advisor: Jacob Scharcanski.

1. Visual object tracking. 2. Face tracking. 3. Facial feature tracking. 4. Tracking error predictor. 5. Online learning. 6. Incremental PCA/ICA. 7. Dictionary learning. I. Jacob Scharcanski, . II. Título.

Dedications

To my mother Najma Nazir and my father Muhammad Khurshid.

ACKNOWLEDGMENT

I am grateful to Allah Almighty for this life, opportunities and keeping me blessed throughout my life. I am also very appreciative of the prophet of Allah, Muhammad (P.B.U.H) for his guidance and teachings. I am indebted to my parents (Muhammad Khurshid and Najma Nazir), my brothers (Aamir Khurshid, Omair Khan, and Awais Khan), my sister (Annam Khan) and other family members for their constant support throughout my entire life. I am very grateful to my advisor Prof. Jacob Scharcanski for his guidance and help during my work. I would also like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for financial support and Instituto de Informática, UFRGS for accepting me as a graduate student. I am grateful to my friends Aamir Khan and Muhammad Farhan for their support from the early years of my graduation till now, and Marcia Lima Rodrigues for her help and support during my stay in Brazil. Importantly, I want to pay my gratitude to my colleagues from the Laboratory (Cesar Salgado, Eliezer Bernart, Eliezer Flores, Pablo Barcelos, Rafael Madeiros) and my friends in Brazil who made my stay very pleasant and appreciable. Finally, I am very much indebted to my thesis proposal committee members (Prof. Dr. Edison Pignaton De Freitas (UFRGS), Dr. Maciel Zortea (IBM Reseach, Brazil), Prof. Dr. John Soldera (IFFAR)) for their help in improving this work.

Aasim Khurshid

ABSTRACT

Object tracking can be used to localize objects in scenes, and also can be used for locating changes in the object's appearance or shape over time. Most of the available object tracking methods tend to perform satisfactorily in controlled environments but tend to fail when the objects appearance or shape changes, or even when the illumination changes (e.g., when tracking non-rigid objects such as a human face). Also, in many available tracking methods, the tracking error tends to increase indefinitely when the target is missed. Therefore, tracking the target objects in long (uninterrupted) video sequences tends to be quite challenging for these methods. This thesis proposes a face tracking algorithm that contains two operating modes. Both the operating modes are based on feature learning techniques that utilize the useful data accumulated during the face tracking and implements an incremental learning framework. To accumulate the training data, the quality of the test sample is checked before its utilization in the incremental and online training scheme. Also, a novel error prediction scheme is proposed that is capable of estimating the tracking error during the execution of the tracking algorithm. Furthermore, an improvement in the Constrained Local Model (CLM) is proposed that utilize the training data to assign weights to the landmarks based on their consistency. These weights are used in the CLM search process to improve CLM search optimization process. The experimental results show that the proposed tracking method (both variants) perform better than the comparative state of the art methods in terms of Root Mean Squared Error (RMSE) and Center Location Error (CLE). In order to prove the efficiency of the proposed techniques, an application in yawning detection is presented.

Keywords: Visual object tracking. Face tracking. Facial feature tracking. Tracking error predictor. Online learning. Incremental PCA/ICA. Dictionary learning.

Rastreamento de Faces Adaptativo Baseado em Aprendizagem On-line

RESUMO

Rastreamento de objetos pode ser usado para localizar objetos em cenas e pode ser também usado para localizar mudanças na aparência ou na forma dos objetos ao longo do tempo. A maioria dos métodos de rastreamento de objetos disponíveis tendem a apresentar um desempenho satisfatório em ambientes controlados, mas tendem a falhar quando a aparência ou a forma dos objetos muda (por exemplo, quando objetos não-rígidos são rastreados), ou mesmo quando ocorrem mudanças de iluminação. Além disso, em muitos métodos de rastreamento disponíveis, o erro de rastreamento tende a aumentar indefinidamente quando o alvo é perdido. Portanto, rastrear objetos em sequencias de vídeos longas (e ininterruptas) tende a ser bastante desafiador para os mesmos métodos. Essa tese propõe um algoritmo de rastreamento facial que contém dois modos de operação. Ambos os modos de operação são baseados em técnicas de aprendizado de feições que utilizam os dados úteis acumulados durante o rastreamento da face e implementam um framework de aprendizado incremental. Para acumular os dados de treinamento, a qualidade da amostra de teste é verificada antes de sua utilização no esquema de treinamento on-line incremental. Adicionalmente, um esquema inovador de predição de erro é proposto e é capaz de estimar o erro de rastreamento durante a execução do algoritmo de rastreamento. Além disso, uma melhora em Modelo Local Restrito (Constrained Local Model - CLM) é proposta e utiliza os dados de treinamento para designar pesos aos pontos de referência (landmarks) baseados em suas consistências. Esses pesos são usados no processo de busca do CLM a fim de melhorar o processo de otimização da busca do CLM. Os resultados experimentais mostram que o método proposto de rastreamento (ambas variações) tem uma melhor performance que os métodos comparativos do estado da arte em termos de Erro Quadrático Médio (Root Mean Squared Error - RMSE) e Erro de Localização de Centro (Center Localization Error - CLE). A fim de provar a eficiência das técnicas propostas, uma aplicação em detecção de bocejos é apresentada.

Palavras-chave: Rastreamento dos objetos, Rastreamento facial, Aprendizado Incremental, Aprendizado por Dicionário, Reconhecimento de Bocejos.

LIST OF ABBREVIATIONS AND ACRONYMS

PCA	Principal Component Analysis
ICA	Independent Component Analysis
SVD	Singular Value Decomposition
IK-SVD	Incremental K-SVD
SKL	Sequential Karhunen-Loeve
SVM	Support Vector Machines
CLM	Constrained Local Models
W-CLM	Weighted Constrained Local Models
BP	Basis Pursuit
MP	Matching Pursuit
OMP	Orthogonal Matching Pursuit
UITDL	Unsupervised Information- Theoretic Dictionary Learning
NMF	Non-Negative Matrix Factorization
MMDL-FT	Multi-Model Dictionary Learning for Face Tracking
MMDL-FTU	Multi-Model Dictionary Learning for Face Tracking with Dictionaries Update
ILFT	Incremental Learning for Face Tracking
AFRTM	Adaptive Face Tracker with Resyncing Mechanism using CLM
AFRTM-W	Adaptive Face Tracker with Resyncing Mechanism using W-CLM
MMI	Multi-variate Mutual Information
ILRVT	Incremental Learning for Robust Visual Tracking
MTCNN	Multi-task Cascaded Convolutional Networks
KCF	Kernelized Correlation Filters
SRDCF	Spatially Regularized Discriminative Correlation Filters
iCCR	Incremental Cascaded Continuous Regression

CLE	Center Location Error
RMSE	Root Mean Square Error
TPR	True Positive Rate
TNR	True Negative Rate
FPR	False Positive Rate
FNR	False Negative Rate
CDR	Correct Detection Rate
BMA	Block Matching Approach
SURF	Speeded Up Robust Features
CAMShift	Continuously Adaptive Mean Shift
MIL	Multiple Instance Learning
AMS	Appearance Model Selection
CONDENSATION	Conditional Density Propagation
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
IFFAR	Instituto Federal de Educação, Ciência e Tecnologia Farroupilha
IBM	International Business Machines
P.B.U.H	Peace Be Upon Him

CONTENTS

LIST OF FIGURES **11**
LIST OF TABLES **12**
1 INTRODUCTION **13**
1.1 Contributions **17**
1.2 Characteristic Evaluation of the Tracking Methods **17**
1.3 Thesis Organization **18**
2 LITERATURE REVIEW **19**
3 FUNDAMENTAL CONCEPTS ON FACE TRACKING **29**
3.1 Probability Distribution **29**
3.2 Background on Incremental Learning **29**
3.2.1 Principal Component Analysis 32
3.2.2 Incremental Update of the Eigenbasis 34
3.2.3 Incremental Update of the Mean 37
3.3 Background on Dictionary Learning for Sparse Representation **37**
3.3.1 Sparse Representation 39
3.3.2 **Basis Pursuit(BP)** 40
3.3.3 **Matching Pursuit (MP)** 40
3.3.4 Dictionary Learning 41
3.3.5 Incremental Dictionary Learning 42
3.3.6 **Estimating New Dictionary Atoms** 44
3.4 Classification Methods and the Support Vector Machine (SVM) Classifier **45**
3.5 Constrained Local Models (CLM) **46**
3.5.1 CLM Model Building 47
3.5.2 CLM Search 48
3.6 Terminology **49**
4 PROPOSED METHODOLOGY **52**
4.1 Motion Model and Sampling **53**
4.2 Proposed Methodology for Multi-Model Dictionary Learning for Face Tracking (MMDL-FT) and MMDL-FT with Update Test (MMDL-FTU) **58**
4.2.1 MMDL-FTU Operating Mode 58
4.2.2 Dictionary Learning 61
4.2.3 Incremental Dictionary Update 61
4.2.4 Multi-Model Dictionary Learning 63
4.2.5 Reconstruction Dictionary 63
4.2.6 Classification Dictionary 64
4.2.7 Appearance Model and Tracked Target Face Selection using MMDL 65
4.2.8 Facial Landmarks Localization 67
4.2.9 Pseudo Code 67
4.3 Proposed Methodology for an Adaptive Face Tracker with Resyncing Mechanism (AFTRM) and AFTRM Weighted (AFTRM-W) **69**
4.3.1 AFTRM-W Operating Mode 69
4.3.2 Incremental Update of Eigenbasis and Mean 72
4.3.3 Weighted Constrained Local Models (W-CLM) as Re-syncing Feature Detectors 74
4.3.4 CLM Weighted Search 75
4.3.5 Appearance Model and Tracked Target Face Selection 77
4.3.6 Tracking Error Predictor and Resyncing Mechanism 79

5 EXPERIMENTAL RESULTS 82
5.1 Experimental Settings 82
5.1.1 Quantitative Evaluation Strategy 84
5.1.2 Choice of Batch Size 85
5.2 Experimental Qualitative Results 86
5.2.1 Qualitative Evaluation of the MMDL-FT and MMDL-FTU Face Tracking Method 89
5.2.2 Qualitative Evaluation of the AFTRM and AFTRM-W Face Tracking Methods 94
5.3 Quantitative Evaluation and Discussion of the Proposed MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W Face Tracking Methods 99
5.4 Evaluation of the Proposed Face Tracking Method in Yawning Detection 103
6 CONCLUDING REMARKS 107
6.1 Publications 109
6.2 Submitted Article 109
REFERENCES 110

LIST OF FIGURES

Figure 3.1 Pdf of a Gaussian distribution with μ=0 and $\sigma = 1$. 30

Figure 4.1 Block diagram of the proposed face tracking method. 53
Figure 4.2 Motion model example ($p(\chi(t)|\chi(t-1))$) in image space; Affine parameter space, each point in affine parameter space is warped into a bounding box in the image space. 56
Figure 4.3 Motion model example ($p(\chi(t)|\chi(t-1))$); Tracked target face bounding box (in red color) and candidate target face bounding boxes (in green color). 57
Figure 4.4 Motion model example going from the previous frame to the current frame ($p(\chi(t)|\chi(t-1))$). 57
Figure 4.5 Tracked target face sample (red bounding box) and candidate target face samples (green bounding boxes). 58
Figure 4.6 Block diagram of the proposed MMDL based face tracking method. 62
Figure 4.7 Block diagram of the proposed AFTRM-W face tracking method 73

Figure 5.1 Some example frames from the dataset with drivers performing different actions. 83
Figure 5.2 Batch size effect on error (ε) and number of resyncs (r). 87
Figure 5.3 Batch size effect on error (ε) and number of resyncs (r) in multiple videos (normalized to [0,1]). 88
Figure 5.4 An example plot of cost function $c(r_\tau, \varepsilon_\tau)$ and batch size τ (τ = [1 16]). 88
Figure 5.5 Results of the proposed MMDL-FTU method for different face condition evaluated in the tests, red = tracked landmarks, yellow = ground-truth landmarks. 91
Figure 5.6 RMSE across frames of 68 landmarks of the complete tested videos. 92
Figure 5.7 Mean Error across frames of all landmarks of the complete tested videos. ..93
Figure 5.8 Example figure showing face bounding box, facial landmarks, mean face, tracked target face, reconstruction error, reconstructed face and most important eigenbases. 95
Figure 5.9 Results of the Incremental Learning for Face Tracking Algorithm without Resync (ILFT). 96
Figure 5.10 Results of the Incremental Learning for Face Tracking Algorithm without Resync (ILFT). 97
Figure 5.11 Plot of $\Delta(t)$ and $\varepsilon(t)$. 98
Figure 5.12 Plot of $\overline{\Delta}(t)$ and $\overline{\varepsilon}(t)$. 99
Figure 5.13 Plot of $\hat{\Delta}(t)$ and $\hat{\varepsilon}(t)$ 99
Figure 5.14 Results of the proposed Adaptive Face Tracker with Resyncing Mechanism (AFTRM-W). 100
Figure 5.15 Results of the proposed Adaptive Face Tracker with Resyncing Mechanism (AFTRM-W). 101
Figure 5.16 Illustration of a closed mouth (a,b,c,d,e) and yawning sequence (f,g,h,i,j). 104

LIST OF TABLES

Table 1.1 Important characteristics of the face tracking methods. 18

Table 2.1 Table of the evolution of the face tracking methods. 26

Table 3.1 Table of symbols 49

Table 5.1 Key characteristics of the selected videos from the dataset for detailed evaluation. 84
Table 5.2 Average RMSE ε_M and number of times resync is activated for different batch sizes τ. 86
Table 5.3 Average frames per second (fps) and number of times resync is activated for different batch sizes τ. 86
Table 5.4 Average RMSE comparison of MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W with comparative methods (the best results are in bold). 102
Table 5.5 Center Location Error (CLE) comparison of MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W with comparative methods (the best results are in bold). 103
Table 5.6 Yawning Detection Results (the best result are in bold). 106

1 INTRODUCTION

Object tracking essentially deals with locating, identifying, and determining the dynamics of the moving (possibly deformable) target(s). The target(s) could be a single object or parts of an object. In fact, object tracking may become quite challenging when there are changes in the appearance or shape of the target, when the scene illumination changes, temporary occlusions and/or tracking conditions are altered in time. Similarly, noise and different lighting conditions during the day may affect the local illumination in various ways (JUNG; SCHARCANSKI, 2004). Keeping these issues in focus, numerous algorithms have been proposed in the literature for object tracking in video sequences such as incremental learning for robust visual tracking (ROSS et al., 2008), Multiple Instance Learning (MIL) discriminative classifier based tracking (BABENKO; YANG; BELONGIE, 2009), Appearance and shape models for face detection and facial landmark tracking (COOTES et al., 2001; LUCEY et al., 2009; CRISTINACCE; COOTES, 2006), Continuously Adaptive Mean Shift (CAMShift) tracker (BRADSKI, 1998), and so on. However, most methods available in the literature tend to perform well over short time spans and under controlled conditions. Furthermore, in most of these methods, when the object tracking method misses the target, the tracking error tends to increase indefinitely. This work proposes to minimize this difficulty by using online learning scheme that utilizes the data received during tracking to update the appearance model of the object (i.e., face). The appearance model is updated after checking the quality of the tracked target object samples before utilizing this sample to update the appearance. Also, a resyncing scheme is introduced that corrects the tracking process once the tracking error is estimated to be high during runtime.

In this thesis, a face tracking method is proposed which contain two operating modes. Both the operating modes are based on the incremental learning approach and update the appearance model of the tracked target object using incremental Singular Value Decomposition (SVD) algorithm (ROSS et al., 2008). Generally, there are two main components in an object tracking algorithm, which are the motion model and the appearance model. The motion model is responsible for handling the motion parameters of the object and estimation of the candidate target object samples. Both operating modes of the proposed tracking scheme in this thesis use the same motion model. The state of the tracked target object is represented using the affine parameters τ, and its motion is modeled using a Gaussian distribution. On the other hand, the appearance model is utilized to estimate

the tracked target object among the candidate target object samples. The operating modes of the proposed face tracking algorithm in this thesis differ in the representation of their corresponding appearance model. The appearance model of the first operating mode is based on incremental learning of a multi-model K-SVD dictionary, that utilizes incremental SVD to update the dictionary atoms (see details in Section 4.2). This method performs well, however, in complex scenarios like a rapid movement of the object, it tends to fail. For this reason, the second variation of the method adds another component to make the tracking process robust with an additional cost of time. The second operating mode of the proposed tracking algorithm uses incremental SVD as a representation model instead of sparse representation using dictionaries, that increases the speed at the cost of tracking quality. Furthermore, in this tracking mode, a resyncing scheme is utilized to improve the tracking process when the proposed tracking predictor indicates high tracking error. A weighted Constrained Local Model (CLM) scheme is used as a resyncing scheme that improves the tracking performance (see details in Section 4.3). Both these operating modes are explained briefly in the following paragraphs, and the detailed explanation of each method is presented explicitly for better understanding in their corresponding Sections 4.2 and 4.3, respectively in chapter 4.

Recently, the linear decomposition of data using a few atoms of a learned dictionary, instead of using a pre-defined set of bases has produced interesting results in different areas of machine learning and image processing, including object recognition (FANELLO et al., 2013) and texture analysis (PEYRÉ, 2009). This suggests that the sparse learned model using dictionaries can be effective for object detection, classification, and tracking. In this work, the dictionaries are explored in the context of face tracking. A new approach called Multi-Model Dictionary Learning for face tracking (MMDL-FT) is proposed that build two dictionaries - a classification dictionary and a reconstruction dictionary - in parallel. Both the dictionaries are based on the k-SVD dictionary learning technique and are combined into a single multi-model dictionary. The proposed MMDL-FT scheme can reconstruct the face, in addition to discriminating the face from the background. The proposed method learns the face appearance using dictionary atoms constructed from patches, that are taken from positive and negative samples of the training data which are collected during the tracking process.

Moreover, a smart approach is proposed to update the dictionaries incrementally and efficiently, making feasible the application of the proposed method to realistic tracking scenarios. The dictionaries are updated during tracking so that it is capable of adapting

to the changes in the tracked target face along time. Furthermore, the proposed MMDL-FT method collects training samples to update the two dictionaries during face tracking using the proposed scheme (see details in Section 4.2.4). The quality of the samples in terms of reconstruction error is assessed before utilizing them to update the dictionaries, which is an aspect that other methods that implement incremental learning seem to lack (ROSS et al., 2008). Both the dictionaries are initialized using the SVD, which is more efficient than initializing the process by combining random training samples as proposed elsewhere (ELAD; AHARON, 2006). As both dictionaries are learned incrementally, the number of atoms can increase until a limit is reached. Additionally, the weights of the atoms are updated in an adaptive manner.

Similarly, online learning has been proposed as a relevant resource to improve visual tracking such as online random forests (SANTNER et al., 2010), incremental learning using Principal Component Analysis (PCA) (ROSS et al., 2008), Karhunen-Loeve (KL) transform as appearance model with dynamic update of image database (LEVEY; LINDENBAUM, 2000) and Appearance Model Selection (AMS) (YUAN et al., 2013). These visual tracking methods work in real-time, but usually tend to miss the target in complex scenarios during special conditions such as when the head pose changes during the face, cluttered background and/or object occlusions (COOTES et al., 2001). On the other hand, shape and appearance models such as Active Appearance Models (AAM) (COOTES et al., 2001), Active Shape Models (ASM) (COOTES; TAYLOR et al., 2004; BEHAINE; SCHARCANSKI, 2012) and Constrained Local Models (CLM) (LUCEY et al., 2009; CRISTINACCE; COOTES, 2006; WANG; LUCEY; COHN, 2008; SARAGIH; LUCEY; COHN, 2009) can capture robust features even in cluttered or fast changing scenarios, and these robust features can improve the tracking process if adequate training data is provided. These methods often are based on local shape matching and require to minimize the difference between a tracked target object and the learned target appearance (i.e., to maximize the match). Unfortunately, most shape and appearance model based methods are not easily applicable to real time tracking due to their complexity. Nevertheless, combining online learning with shape and appearance models can increase the online learning efficiency (e.g., by using appearance models to correct the tracking process and reduce tracking failures). The second operating mode proposed in this work combines an online learning scheme with a shape and appearance model (i.e., CLM) that tends to improve tracking robustness.

The second operating mode of the proposed approach presented in this work im-

proves on a well-known object tracking method based on the incremental PCA (ROSS et al., 2008). The proposed scheme learns from the data generated during the object (i.e., face) tracking and corrects tracking mistakes with a resyncing mechanism. A dynamic tracking error predictor is proposed, which estimates the tracking correctness of the tracked target face during tracking, and checks if the target has been missed. The tracking error predictor adapts itself along time and tends to be consistent in long video sequences. If the estimated tracking error is increasing, the tracking process is corrected by a resyncing mechanism based on CLM. Moreover, an improvement in CLM called Weighted CLM (W-CLM) is proposed that utilize the training data to assign weights based on the consistency of each landmark (feature point) to be utilized in the CLM search process (see details in Section 4.3.3). To each landmark, a weight is assigned according to its consistency based on Multi-variate Mutual Information (MMI) (CRUYS, 2011). In this approach, the highest weight is assigned to the most consistent feature point and so on, and this weight is utilized in the CLM search process named weighted-CLM search (see details in Section 4.3.3). One of the possible applications of the proposed tracking method is face tracking where CLM/W-CLM can be used to relocate the facial landmarks when there is a tracking failure, and the target face has been missed. The proposed technique called Adaptive Object Tracker with Resyncing Mechanism (AFTRM) optimizes the CLM search process without using the weights. Whereas, also the proposed Adaptive Object Tracker with Resyncing Mechanism Weighted (AFTRM-W) applies a weight (calculated during the training phase) to every landmark in the search process.

In order to evaluate the proposed object tracking scheme in real world applications, we investigate its application in face tracking. Face tracking based on facial features is relevant for a number of applications, such as yawning detection, expression analysis, human computer interfaces, and face recognition (OMIDYEGANEH et al., 2016; VATER; IVANCEVIC; LEóN, 2017; SOLDERA et al., 2017; SOLDERA; DODSON; SCHARCANSKI, 2017). Furthermore, image-based measurements can provide cost effective solutions for fatigue and vigilance systems if the detected facial features are accurate (SHIRMOHAMMADI; FERRERO, 2014). In our experiments, the proposed method is evaluated in a face tracking application - yawning detection in the context of a driving scenario.

1.1 Contributions

The main contributions of the proposed method include:

- Multi-Model Dictionary Learning (MMDL), which combines two dictionaries (a classification and a reconstruction dictionary) and MMDL is utilized for face tracking which tends to improve tracking robustness.
- An error prediction scheme to evaluate the correctness of the tracking process during face tracking.
- Utilization of a resyncing mechanism based on the Constrained Local Models (CLM).
- An improvement in the classical CLM approach, namely Weighted CLM (W-CLM).
- An improvement in an application of the facial analysis (i.e., yawning detection).

1.2 Characteristic Evaluation of the Tracking Methods

Some important characteristics that can be used to evaluate the quality of the face tracking methods are listed and explained briefly as follows:

- **Training phase:** The tracking method requires training data to train the model beforehand in a training phase, and this model is used to track the target during tracking.
- **Real-time:** The real-time tracking method tracks the target at the rate at which the video is recorded by the camera. Real-time video processing means that the method is capable to process thirty (30) frames per second. However, in tracking algorithms, the object has smooth transition in appearance or shape changes. Therefore, it is possible to process sparse frames, i.e., skipping some frames every second. In this document, we consider any tracking method that tracks more than six (6) frames per second to be real-time because sparse video frames can be used in tracking to extract meaningful information. In the proposed method, we skip five (5) frames to extract the information to achieve real-time processing of the videos in real-scenarios. This kind of processing is called near real-time processing.
- **Error predictor:** If the tracker provides an in built capability to track the correctness of the tracking process.

- **Resyncing:** If the tracker uses some resyncing mechanism in case it fails to track the target face at times. The resyncing is not necessarily related to the tracking method. It is also possible to utilize a resyncing mechanism after every specific number of frames.

- **Robust:** If the tracker is robust enough to not lose track of the face and keeps tracking the face infinitely and under different tracking conditions.

- **Online:** If the tracker learns the appearance model incrementally and in an online way so that it utilizes the useful data available during tracking to update the appearance model. This makes the tracker capable of adapting to the recent changes in the tracked target face and the tracking conditions along time.

Table 1.1 provides the characteristics of the proposed face tracking method in comparison to the state of the art face tracking methods.

Table 1.1: Important characteristics of the face tracking methods.

Method	Training	Real-Time	Error Predictor	Resyncing	Robust	Online
(ROSS et al., 2008)	No	yes	No	No	No	Yes
(ZHENG; STURGESS; TORR, 2013)	Yes	Yes	No	No	Yes	No
(TERISSI; GÓMEZ, 2007)	No	Yes	No	No	No	Yes
(SÁNCHEZ-LOZANO et al., 2016)	Yes	No	No	No	Yes	Yes
MMDL-FT (Proposed)	No	Yes	No	No	Yes	Yes
MMDL-FTU (Proposed)	No	Yes	Yes	No	Yes	Yes
AFTRM (Proposed)	Yes	No	Yes	Yes	Yes	Yes
AFTRM-W (Proposed)	Yes	No	Yes	Yes	Yes	Yes

1.3 Thesis Organization

The remaining of this thesis is organized as follows: Chapter 2 reviews the relevant work in the face tracking and motivates the proposition of our approach. The fundamental concepts utilized in the proposed face tracking method are detailed in Chapter 3, the proposed face tracking scheme with its variants is detailed in Chapter 4, followed by the experimental results that are demonstrated and discussed in Chapter 5. Finally, Chapter 6 gives conclusions and future prospects of this work.

2 LITERATURE REVIEW

Numerous algorithms have been proposed in the literature for visual object tracking which are capable of identifying and locating the target object(s) in consecutive frames of a video sequence (YILMAZ; JAVED; SHAH, 2006; SANTNER et al., 2010; COLLINS; LIU; LEORDEANU, 2005; TERISSI; GÓMEZ, 2007). The work on visual tracking started with the use of different features, transformations and also explored many color spaces for robust tracking (CHRYSOS et al., 2014). For instance, Mean-Shift approach (CHENG, 1995) is a non-parametric approach that climbs the gradient of a probability distribution to find the nearest peak. In this method, the probability distribution is static, and it only updates the mean when there is a significant change in color, shape or size of the object. Whereas, in a continuous video sequence, the distribution derived from the object can change and move from one frame to the other. For this reason, the static mean-shift algorithm is not suitable for tracking applications. Bradski et al. (BRADSKI, 1998) proposed a variation to Mean-Shift called Continuously adaptive mean shift (CAMSHIFT). The CAMSHIFT use histogram methods to produce adaptive probability distribution for object tracking in video sequences.

Other popular techniques are based on linear and non-linear filtering including Kalman filters and particle filters. Conditional density propagation for visual tracking (CONDENSATION) algorithm is among the techniques that use Kalman filter (BLAKE; ISARD, 1997). The CONDENSATION algorithm uses factored sampling that generates samples randomly based on the previous observations instead of a Gaussian distribution as performed in Kalman filtering. The output of the sampling process is a weighted sample set of fixed size in each iteration. These samples are used as candidate samples which may contain the tracked target object. The tracked target object is selected among these candidate samples based on an appearance model (e.g., Principal Component Analysis). Also, 3D models have been explored for robust tracking, such as Malciu et al. (MALCIU; PRÊTEUX, 2000) which uses a generic 3D model to track the target object in a video sequence. They proposed to locate the tracked target object by matching 2D image features taken throughout the video sequence and the 3D object features of a generic head model. The 3D model features correspond to the geometry of the model, while the 2D image features correspond to the optical flow in the head region in the consecutive video frames and the texture features of the head in the current frame. Furthermore, in order to increase the frame rate in handling the rigid head motion and face deformation, their

method utilizes a non-linear optical flow based on interpolation.

Similarly, Terissi et al. (TERISSI; GÓMEZ, 2007) proposed an algorithm for facial motion tracking of an avatar based on a video sequence of a real person. This method predicts the location of the landmarks on the human face in two steps; the first step is based on the image segmentation and pattern comparison techniques that predict the position of each landmark independently by a matching process taking into account its texture information in the previous frame. The second step is based on Independent Component Analysis (ICA) technique that predicts the final location of the landmark in the current frame. Furthermore, in this method, Independent Components (ICs) are used to learn different facial expressions extracted from training videos. However, this algorithm tends to fail in the tracking of the facial features as the number of frames increases, which authors of the paper attribute to the propagation of tracking error. One of the main reasons for the failure of such methods is that they employ fixed appearance models of the target object. Such models are trained with the data available before the tracking begins, which limits the possible appearance changes that the system can handle.

Some researchers modeled the human face as a texture mapped cylinder and treated the object tracking as an image registration problem (CASCIA; SCLAROFF, 1999; CASCIA; SCLAROFF; ATHITSOS, 2000). The registration problems usually lead to higher error in cases of variations in illumination conditions and object motion. To handle this problem, the method proposed in (CASCIA; SCLAROFF, 1999) models the registration error using a linear combination of texture mapping templates and orthogonal illumination templates. An improvement made in the previous work from the authors that uses the same template model, however, to achieve stable on-line tracking a weighted least-squares minimization of the registration error is used (CASCIA; SCLAROFF; ATHITSOS, 2000).

Tracking rigid objects using 3D model is usually classified as general model-free tracking because it does not take into account local changes happening in the structure or shape of the object. On the other hand, non-rigid object tracking has fascinated more researchers due to its importance in certain areas like facial analysis, gaming applications, etc. Face analysis is arguably one of the most extensively studied field in image processing and computer vision due to its applications in security, expression analysis, fatigue detection, gesture detection and other automated applications (LANITIS; TAYLOR; COOTES, 1995; CHIANG et al., 2003; WANG; LUCEY; COHN, 2008; SARAGIH; LUCEY; COHN, 2009; CHRYSOS et al., 2014; SÁNCHEZ-LOZANO et al., 2016; SOLDERA et al., 2017; VATER; IVANCEVIC; LEóN, 2017). However, all these applications

require the correct detection of facial features which include mouth, eyes, lips.

Online learning has emerged as a relevant resource for visual tracking. Online learning approaches utilize useful data that becomes available during the tracking process to accommodate possible changes in the appearance and/or shape of the target during the tracking process (ROSS et al., 2008; LEVEY; LINDENBAUM, 2000; BABENKO; YANG; BELONGIE, 2009). A method that uses a variation of CONDENSATION algorithm to model the distribution called incremental learning for robust visual tracking (ILRVT) (ROSS et al., 2008) has been successful for rigid object tracking. ILRVT learns the eigenbases in an online fashion and does not need any training as compared to other eigentrackers such as Eigentracking: Robust matching and tracking of articulated objects using a view-based representation (BLACK; JEPSON, 1998), which requires an offline training with the data available before the tracking process has started. The data representation model update is based on PCA and uses a forgetting factor to ensure that less modeling effort is spent on fitting older observation data. The method shows effectiveness in indoor and outdoor environments with sufficient adaptability to changes in pose, scale, and illumination. However, once the tracked target object is missed, the error keeps on increasing, and the tracking fails indefinitely.

Babenko et al. (BABENKO; YANG; BELONGIE, 2009) trained a Multiple Instance Learning (MIL) discriminative classifier, which bootstraps itself by using the current tracker state to extract positive and negative examples. However, slight inaccuracies may arise and will be followed by a tracking target drift. Zheng et al. (ZHENG; STURGESS; TORR, 2013) recently proposed a learning scheme that combines the appearance of facial landmarks using binary features and logical operations to achieve real-time tracking. The methods described above work well for limited periods of time, but are prone to introduce slight inaccuracies which may lead to drift with losing tracking after a while. To tackle this problem, tracking error predictor could be helpful if it can automatically detect the inaccuracies in the tracking process. Hence, a necessary action could be taken, e.g., resync the tracker with a comprehensive feature detector to improve the performance, which could make tracking more robust and stable.

Non-rigid object tracking is important in many areas like hand tracking, face tracking, animal tracking etc, with varied applications ranging from game design to facial analysis applications and surveillance (BLACK; JEPSON, 1998; COOTES et al., 2001; CHRYSOS et al., 2014). The non-rigid object which attracted overwhelming attention happened to be face, although, the interpretation of human face is quite challenging due

to its vast variability (LANITIS; TAYLOR; COOTES, 1995). Non-rigid face tracking is usually treated as tracking a certain number of landmarks on the face that define the shape and appearance of the tracked target face at any particular moment of time t. Some prominent works on facial landmark tracking include (LANITIS; TAYLOR; COOTES, 1995; BLACK; JEPSON, 1998; ESSA; PENTLAND, 1994; ESSA et al., 1996; ESSA; PENTLAND, 1997; CORTES; VAPNIK, 1995; COOTES et al., 2001; COOTES; TAYLOR et al., 2004; BEHAINE; SCHARCANSKI, 2012; CRISTINACCE; COOTES, 2006; CHRYSOS et al., 2014; SÁNCHEZ-LOZANO et al., 2016). Some works combined the physical models with geometric models to describe the facial structure and expressions (LANITIS; TAYLOR; COOTES, 1995; BLACK; JEPSON, 1998; ESSA; PENTLAND, 1994; ESSA et al., 1996; ESSA; PENTLAND, 1997). The work of Essa et al. (ESSA; PENTLAND, 1994), (ESSA et al., 1996) used a geometric and a physical model of skin and muscle. Both of these models are driven by the optical flow to describe different facial expressions. In this method, the visual observation of the facial expression analysis system is modeled using optical flow, which is complemented by a physical model that models the muscle motion. Soon after, an improvement of the algorithm followed which models facial motion using the Kalman filter based algorithm for the motion estimation along with the physical model to measure different facial expressions (ESSA; PENTLAND, 1997). Koelstra et al. (KOELSTRA; PANTIC; PATRAS, 2010) uses an affine registration to remove the translation and rotation affects and then utilize Motion History Images (MHIs) method (RUECKERT et al., 1999), to extract dense facial motion in consecutive frames in a video sequence. After this, a quad-tree decomposition is performed to select the regions related to each Action Unit (AU). Next step uses the Gentleboost and Hidden Markov Model (HMM) classification methods to classify the video sequence into segments containing different facial expressions. A recent tracking algorithm called Face Flow (SNAPE et al., 2015) presents an optical flow based approach that formulates dense motion in facial shape as an energy minimization problem. The major contribution of the paper includes a model-based formulation that adopts a low-rank optimization, which makes it efficient to calculate the motion.

In order to achieve high tracking accuracy, some methods chose to combine multiple techniques which could also be used independently to track objects (RANGANATHA; GOWRAMMA, 2017; SALHI et al., 2017). Ranganatha et al. combine Continuously Adaptive Mean Shift (CAMShift) and Kalman filter to track a human face in video sequences (RANGANATHA; GOWRAMMA, 2017). This method utilizes a cascaded face

detection scheme to detect a face as an initialization, followed by key feature detection using Speeded Up Robust Features (SURF). These features are then used to localize the target face by estimating the center location and bounding box containing the target face using CAMshift and Kalman filter respectively. Salhi et al. combines multiple approaches to solve a face tracking problem which include Block Matching Approach (BMA), Meanshift, Camshift, and Kalman filter (SALHI et al., 2017). This method uses BMA to detect the face in the beginning and then applies Meanshift, Camshift and Kalman filter together to achieve high confidence score for tracking the target face. The block matching algorithm used in this method is based on color histograms, which makes it difficult to apply in the applications prone to lighting condition changes along time.

In recent years, deep learning and neural networks have also been used in face detection and tracking applications (GIRSHICK et al., 2014; REN et al., 2017; CHOI; KIM, 2018; KWON et al., 2017). Girshick et al. utilize the region proposal and convolution neural network (CNN) to design the R-CNN for object detection (GIRSHICK et al., 2014). It detects the objects in a scene and generates a mask for each instance as well. However, it carries out the deep learning feature leaning for each region, which is a slow and time-consuming process. Ren et al. uses a lightweight CNN and combines the network with Kalman filter for face tracking. However, in occlusion scenarios, the target face is detected by using only the Kalman filter, which may result in tracking failure when the target face is occluded (REN et al., 2017). The method proposed by Kwon et al. incorporates Multi-task Cascaded Convolutional Neural Networks (MTCNN) and Kernelized Correlation Filters (KCF) to track human faces at real-time speed (KWON et al., 2017). This method utilizes color histograms and geometric features in order to handle occlusions and provides a robust scheme for face tracking. Recently, Choi et al. proposed a Deep Manifold Embedding Active Shape Model (DME-ASM) applied to Face Tracking (CHOI; KIM, 2018). This method uses a manifold learning technique with ASM for handling different poses, and a CNN is trained for categorizing the pose range. This method shows good results on the database which is also built by the authors on different pose scenarios. Despite the accuracy of the Neural Network methods, these methods have a high dependency on hardware and dataset, which limits their scope in small organizations and in applications where the resources such as data and computational capacity are low.

The problem of sparse facial feature tracking has attracted a lot of attention in the recent past (COOTES; TAYLOR et al., 2004; COOTES et al., 2001; LUCEY et al.,

2009; CRISTINACCE; COOTES, 2006; BEHAINE; SCHARCANSKI, 2012; WANG; LUCEY; COHN, 2008; SARAGIH; LUCEY; COHN, 2009; CHRYSOS et al., 2014; SÁNCHEZ-LOZANO et al., 2016). Currently, the deformable face tracking has converged to tracking a set of facial landmarks, however, most of these methods require an initialization (CHRYSOS et al., 2014). The most popular techniques in facial landmark detection involve face detection followed by a landmark localization technique. A variation called model free tracking initializes a bounding box around the center of the face, followed by the landmark localization within the tracked bounding box. Furthermore, hybrid methods have also been investigated that place bounding box for the landmark localization to improve the robustness.

Appearance and shape models (COOTES; TAYLOR et al., 2004; COOTES et al., 2001; LUCEY et al., 2009; CRISTINACCE; COOTES, 2006; WANG; LUCEY; COHN, 2008; SARAGIH; LUCEY; COHN, 2009) are popular feature detector methods, that tend to provide accurate results if sufficient training data is available. Shape models like Active Shape Models (ASM) use texture models in small regions around the landmarks to minimize the distance between the corresponding landmarks and use constraints on global shape deformation. The key difference of ASM from the previous model based approaches is that ASM only allows the deformation in the model that is consistent with the training set, to fit the test data (COOTES et al., 1995). The allowable deformation is called Allowable Shape Domain (i.e., any allowed shape) and any point in this domain can be achieved by using a mean shape and a set of eigenvectors with its corresponding eigenvalues (weight of the variance) which represent the variation in shapes of the object.

The initial experiments of ASM were performed on transistor shapes and hand shapes, however, later it showed good performance on other non-rigid objects as well, such as faces (BEHAINE; SCHARCANSKI, 2012). The work by Behaine et al. (BEHAINE; SCHARCANSKI, 2012) proposed an improvement in the ASM landmark selection scheme in the context of face recognition. The method selects robust landmarks and assigns higher weights to the most relevant facial landmarks, which tends to enhance the performance of the face classification in an ASM based approach. On the other hand, Active Appearance Model (AAM) combines a shape model with a statistical model of appearance (COOTES et al., 2001). AAM uses an appearance model of the whole target region as a reference and seeks to minimize the difference between the shape of this global appearance model and the target shape. To build a statistical appearance model, each training sample is warped in such a way that its landmark points match the mean shape. Next,

gray level information is sampled from the normalized image over the region covered by the mean shape. This way shape and appearance of any sample can be summarized by the shape and appearance model, which are represented in a high-dimensional space (i.e., PCA). However, in strong head movements, the applicability of AAM may suffer due to its high sensitivity to initialization and results in low robustness (VATER; IVANCEVIC; LEóN, 2017). Vater et al. proposed to automate the initialization process by embedding an iris localization scheme (VATER; IVANCEVIC; LEóN, 2017).

Constraint Local Models (CLM) (CRISTINACCE; COOTES, 2006; LUCEY et al., 2009) capitalize advantages of both the AAM and the ASM approaches and combines the shape and texture information while maintaining local constraints, so that the target object can be located more efficiently. CLM uses PCA to build the shape model, and each shape may be represented using eigenvectors and mean shape. Although the appearance is modeled using a patch model, that is posed as a classification problem. Patch model can be considered as a patch expert trained to discriminate between positive and negative patch samples. Both of these models are combined into one model while searching for the object in the image using an optimization technique. Given a test image, the CLM search process tries to find a shape that is similar to the mean shape, while the patch model response is as high as possible by using the optimization process.

The aforementioned landmark localization methods tend to be accurate to estimate the location of facial features but usually are not easily applicable in practice because the optimization process is costly. However, the accuracy of these methods could be advantageous if they are utilized less frequently in a resyncing mechanism. These landmark localization methods can be used with online learning techniques to re-sync in case of failure of these methods during the tracking process. This hybrid of the two different classes of methods could provide better results in terms of execution time and accuracy in comparison to their individual use. This motivated one of the operating modes of the proposed tracking scheme in this work (details on this in Section 4.3). The second operating mode of the proposed tracking scheme is also based on the incremental learning procedure, although it utilizes multi-model dictionary learning as a sole appearance model (details on this in Section 4.2).

Table 2.1 provides an overview of some important face tracking methods proposed in the literature and the proposed method. The table contains a short overview of the method mentioned in the first column, its advantages and disadvantages.

Table 2.1: Table of the evolution of the face tracking methods.

Method	Overview	Advantages	Disadvantages
(COOTES et al., 1995)	Active Shape Models (ASM) use texture models in small regions around the facial landmark.	Use constraint on global shape, makes it more effective.	Slow optimization and dependency on training set.
(BLAKE; ISARD, 1997)	Conditional density propagation for visual tracking models motion using Kalman filtering and appearance using PCA.	Introduced use of Kalman filtering in tracking.	Offline training, No appearance model update.
(MALCIU; PRÊTEUX, 2000)	Matches 2D image features taken from video sequence and trained 3D model to locate the tracked target.	Use of geometric features in 3D model.	Difficult to analyze 3D features from 2D images.
(COOTES et al., 2001)	Active Appearance Models (AAM) combines shape model with statistical model of appearance.	Statistically analyze the training data, which makes it less prone to tracking errors. Also, locate important facial features.	Slow optimization process and dependency on training set.
(ROSS et al., 2008)	Models motion using Gaussian distribution and estimates tracked object using appearance model based on PCA.	Incremental learning usage, works well for rigid objects.	Error prone in non-rigid objects, fast movement of object.
(LUCEY et al., 2009)	Constraint Local Models (CLM) combines shape model and patch model to locate facial landmarks.	Comprehensive patch model with local constraints makes it an accurate facial landmark detector.	Slow optimization process makes its use in real applications challenging.

Continued on next page

Table 2.1 – *Continued from previous page*

Method	Overview	Advantages	Disadvantages
(ZHENG; STURGESS; TORR, 2013)	An approximation of CLM called Approximate Structured Output Learning for CLM to achieve high speed.	Real time tracking of facial features capable of handling appearance change.	Fails in large face appearance change and face movements.
(GIRSHICK et al., 2014)	Trained a region based CNN.	Works well for different illumination conditions with pose invariance.	Feature learning for each region for CNN is slow. Large amount of training data required and dependency on hardware for training.
(SÁNCHEZ-LOZANO et al., 2016)	Incremental Cascaded Continuous Regression (iCCR) method to track face and facial landmarks. Uses full covariance matrix capturing real statistics of how faces vary between consecutive frames rather than on the shape model eigenvalues.	Updates the shape model with the new tracked shape of the target face.	Training data need to account for large intra-class variation for linear regresser.
MMDL-FTU Operating mode	Incremental multi-model dictionary learning for face tracking.	Multi-model dictionary learning tends to be robust. Adapts to the face appearance change and background change along time. Update dictionaries only using samples with small error.	Error prediction is done using the reconstruction error.

Continued on next page

Table 2.1 – *Continued from previous page*

Method	Overview	Advantages	Disadvantages
AFTRM Operating mode	Incremental learning of appearance with resyncing mechanism in case of tracking error.	Incremental appearance model update, Independent error prediction mechanism. Error prediction and resyncing scheme makes it accurate.	Slow if frequent resyncing of facial features is required.

3 FUNDAMENTAL CONCEPTS ON FACE TRACKING

This chapter explains the fundamental concepts which are used in the proposed face tracking approach.

3.1 Probability Distribution

A probability distribution function describes the probability of occurrence of possible values of a random variable. Many distribution functions are available in the literature, which can be utilized in the generation of random numbers following a specific distribution, which includes Gaussian distribution, Rayleigh distribution, and Rice distribution. In the proposed face tracking method, Gaussian functions is utilized to draw multiple affine parameters to generate multiple affine parameters. These affine parameters are used to warp the candidate target face samples, which may contain the tracked target face in the current frame.

Gaussian functions can be used to model many mathematical processes, which makes them very useful in the fields of signal and image processing (GUO, 2011). Gaussian functions are usually used to represent a probability density function (pdf) of a random variable x that is distributed normally with expected value μ and standard deviation σ as follows:

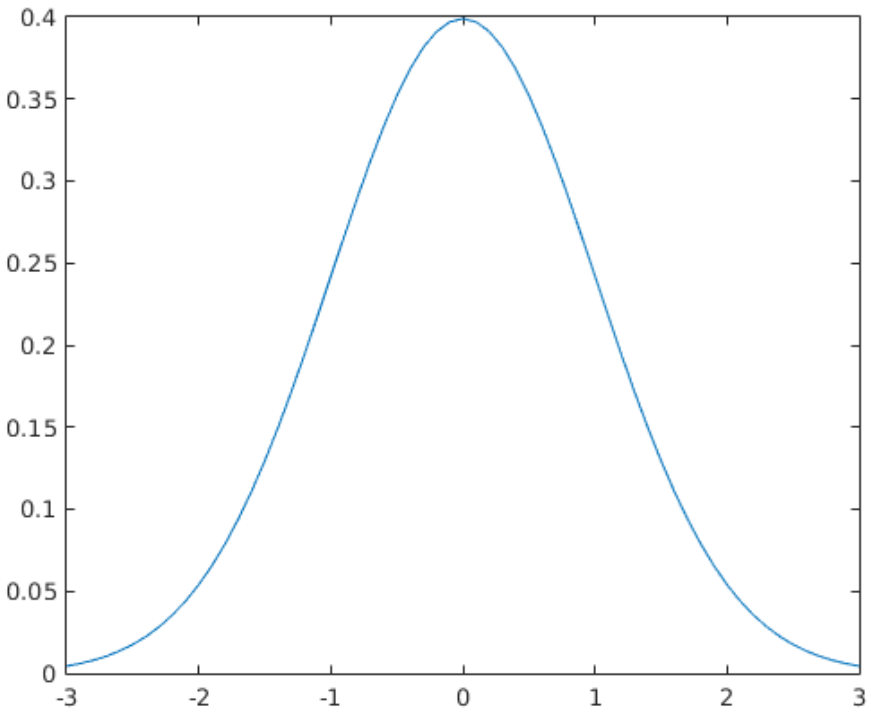
$$\mathcal{N}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{\frac{1}{2}(\frac{x-\mu}{\sigma})^2} . \tag{3.1}$$

The model presented in Eq. 3.1 can be used to obtain values of a random variable. Figure 3.1 shows an example of the pdf of a Gaussian distribution with parameters, μ=0 and $\sigma = 1$. In the Gaussian distribution, about 68% values of the random variable drawn lies within the one standard deviation σ away from the mean μ; 95% within two standard deviations (i.e., 2σ); and 99.7% within three standard deviations (i.e., 3σ).

3.2 Background on Incremental Learning

In recent years, the information flow has increased to enormous levels owing the generation of textual, audio, and video data possessing important insights. The big tech companies such as Google and Facebook own data centers capable of storing 10-15 exabytes (1 exabyte=1 million terabytes) of data (LOSING; HAMMER; WERSING, 2018).

Figure 3.1: Pdf of a Gaussian distribution with μ=0 and $\sigma = 1$.



To mine the collected data for the relevant information and/or predict the future developments, machine learning methods are employed. However, classic machine learning methods which need all the data to be accessed simultaneously are not capable of handling a large amount of data in an environment changing scenario such as video processing. Generally, these methods can not update the model during processing without the need to build the model from scratch. This result in wastage of resources such as time, memory and processing power.

To overcome these issues of batch processed machine learning techniques, the machine learning paradigm was shifted to the online and incremental learning approaches. Incremental learning can be defined as a learning scheme that generates a sequence of models based on the streams of data. This allows the utilization of information as soon as it is available and keeps the model up to date. This scheme also allows better storage management as compared to the batch learning scheme. The main challenges of the incremental learning are the following:

- The model should be updated gradually in such a way that it utilizes the previously computed model effectively and efficiently.
- The previously acquired data should be preserved, with some forgetting factor, to

keep the model up-to-date.

- A limited training data can be maintained at a time so that that memory can be utilized efficiently.

The incremental learning algorithm may be employed in an off-line or an online setting. In an off-line setting, the labeled training data is available beforehand, and the model is trained incrementally by providing this data in small chunks. Therefore, less processing power and memory are required to update the model. It is worth mentioning that, the model h_i is solely dependent on the model h_{i-1} that is constructed previously and a limited number of new labeled data samples. In online settings, the data that is utilized to update the model is not stored (SAFFARI et al., 2009).

Furthermore, online incremental learning algorithms utilize the test data (which is labeled by using the previous model h_{i-1}) to update the model. Each intermediate model is considered for performance evaluation, and each of the models only predicts a limited number of the following test examples. To update the model to h_i, the previous model h_{i-1} and the new examples that are evaluated for performance are used to update the model. This setting of incremental learning can be used for a potentially infinite amount of data or where the training data is not available in advance. Furthermore, it can also be useful in the continuously changing environment such as visual object tracking.

Online and incremental learning approaches apply data in streams for processing and update models to incorporate the data that is available during program execution (LOSING; HAMMER; WERSING, 2018). Numerous techniques have been developed to implement the incremental settings for Big Data. However, most of these methods adapt batch processing techniques to incremental settings (CAUWENBERGHS; POGGIO, 2001). Additionally, the incremental techniques have been developed for machine learning and are applied to different learning tasks such as visual tracking (ROSS et al., 2008), object matching and detection (TOIVANEN; LAMPINEN, 2011), face recognition (WENG; EVANS; HWANG, 2000), discriminative models for classification (SOH; DEMIRIS, 2014), incremental object matching and detection with Bayesian methods and particle filters (TOIVANEN; LAMPINEN, 2011) and so on. Although there are various incremental learning approaches available which can update the model in changing environment during program execution, the selection of a specific incremental approach is problem specific (CHEN; TSAI, 1998).

Incremental and online learning has attracted much attention recently in the context of visual object tracking such as incremental learning using Principal Component

Analysis (PCA), which updates the appearance during tracking (ROSS et al., 2008), Karhunen-Loeve (KL) transform as appearance model with dynamic update of image database (LEVEY; LINDENBAUM, 2000) and so on. Online learning approaches suffer from drifting in case the appearance of the tracked object is changed during tracking (SANTNER et al., 2010). To avoid this problem, Santner et al. (SANTNER et al., 2010) proposed to use the whole spectrum of adaptability to represent the appearance of the tracked target, i.e., a template model (Non-adaptive), online random forest (semi-adaptive), and an optical flow based mean-shift tracker (highly adaptive). Similarly, Appearance Model Selection (AMS) that uses multiple duplicate models to avoid occlusion and in case of large changes in appearance, it stops updating one of them (YUAN et al., 2013).

In this work, the incremental learning is utilized for updating the PCA, which is used as the representation model of the target object in one of the proposed object tracking techniques. Moreover, this update of the eigenbasis is utilized in the incremental update of the reconstruction and the classification dictionaries in the second proposed method. The PCA and its incremental learning procedure are explained next.

3.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique that maximizes the variance of data in the projected space, which makes it suitable for high dimensional data analysis. PCA uses an orthogonal transformation to convert data of a possibly correlated set of variables to a set of uncorrelated variables called principal components. For a set of d-dimensional data vectors X_n, where $n \in \{1, 2, ...N\}$, the q principal axes U_j, $j \in \{1, 2, ...q\}$, are those orthogonal vectors for which the retained variance is maximal.

- The first step is to compute the mean of the data matrix :

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{n} x^i. \tag{3.2}$$

- After this, the mean of the data is subtracted from every data sample to make the data zero mean (centered around zero), which is denoted by $X = \{x^1, x^2, ..., x^n\}$

and each element is computed as shown below:

$$x^i = x^i - \overline{x},\ i = \{1, 2, .., n\}. \tag{3.3}$$

- Next, the eigenvalues and eigenvectors of the matrix $A = X^T X$ are computed, which is essentially a covariance matrix as X is zero centered. As A is a symmetric matrix, it can be diagonalized:

$$A = ULU^T, \tag{3.4}$$

where U represents the eigenvectors matrix and L is a diagonal matrix representing the eigenvalues.

It is worth mentioning that the higher eigenvalue means the eigenvector corresponding to this eigenvalue has higher variation and vice versa. Each column of U is an eigenvector, also called principal axis. The first principal axis is the combination of X variables that have the highest variance, and the second principal axis has the second highest variance, and so on. In other words, the principal axes account for the data variation. Moreover, the projections XU of the data on the principal axes are called the principal components, also known as scores. The j^{th} column of XU represents the j^{th} principal component, whereas the i^{th} data point is given by the i^{th} row of XU in the new PCA space. As the most common objective of the PCA is dimensionality reduction, therefore, the eigenvectors that correspond to higher eigenvalues are usually kept for further processing, and those that correspond to lower eigenvalues are discarded.

The eigenvectors and eigenvalues can also be computed using the Singular Value Decomposition (SVD) directly from the centered matrix X.

$$X = UCV^T, \tag{3.5}$$

where C represents the singular values which are the square roots of the eigenvalues in the covariance matrix, whereas U and V are the left and right orthonormal eigenvectors.

An important property of PCA is that, of all orthogonal projections, the first principal component projection minimizes the reconstruction error. This property can be utilized in finding that how well a new data sample fits into the model defined by a given training set. However, the PCA model suffers from a problem in that it does not define a proper probability model in the space of input data. Suppose we perform a PCA on a cer-

tain data to evaluate how well the new data fit in the model, the only criterion used is the squared distance of the new data from their projections into the principal subspace. It is possible that the data sample is very different from the training data but near the principal subspace. For this reason, a distance measure that defines a proper covariance structure in the data space is used in our method (ROWEIS, 1998). The likelihood of a sample that is different from the training dataset will be much lower even if this data sample is near the principal subspace.

3.2.2 Incremental Update of the Eigenbasis

The shape and appearance of a non-rigid target object may change substantially along time. For this reason, a tracking method should update the appearance model of the object to accommodate local and global changes occurring over time, making the tracking process more robust. Furthermore, in video sequences, the appearance of the target is required to be updated incrementally due to the unavailability of all the training data at the start of the tracking process. This incremental learning of the model is useful in the adaptation of the model to the recent changes in the appearance and shape of the target, and also the changes in the environment (e.g., illumination changes, background changes, etc.).

In the proposed method, the estimation of the tracked target is based on the incremental PCA. To update the PCA incrementally, Levey et al. (LEVEY; LINDENBAUM, 2000) proposed an algorithm, called Sequential Karhunen–Loeve algorithm (SKL) to update the eigenbases such that the speed and memory requirements are improved. This allows the computation of the eigenbases incrementally, in case of a large amount of data and also at times when all the training data is not available in advance. The SKL algorithm provides an online learning scheme that embeds the data sequentially when it is available, instead of waiting to receive the whole data to compute the eigenbases. Furthermore, storage of old data is not required to update the eigenbases, since the method uses the new set of recently received images and the old eigenbases that were built from the previously available data. These features allow SKL application in large datasets feasible.

Ross et al. (ROSS et al., 2008) proposed an improvement in the Sequential Karhunen-Loeve (SKL) algorithm (LEVEY; LINDENBAUM, 2000), and applied this modified scheme to update the eigenbases used in PCA. In this thesis, the incremental update in the eigenbases is inspired by the incremental update of the PCA (ROSS et al., 2008) and is ex-

plained as follow. In (ROSS et al., 2008), each sample is represented by a column vector in order to build and update the PCA, that describes global texture information. However, in order to accommodate local changes, the proposed scheme constructs and updates the eigenbases using the local patches of fixed size ($v \times v$) obtained from target object samples of fixed size ($u \times u$ and $u > v$). As the mean $\mu(t)$ at time t of the features play an important role in PCA, an efficient scheme obtains a weighted update of the mean $\mu(t)$ to accommodate the new data recently acquired. Also, it may be noted that the mean $\mu(t)$ plays a key role in the selection of the tracked target object among the candidate target object samples in the proposed tracking scheme.

Suppose a data matrix $A = \{\mathbf{I}(1), ..., \mathbf{I}(n)\}$ which is $d \times n$ dimensional, where each column $\mathbf{I}(t)$ represents an observation composed of patches of the tracked target object samples represented as a column vector of d dimensions and n is the number of samples in A. The singular value decomposition (SVD) of A can be obtained as follows :

$$A = UCV^T, \tag{3.6}$$

where C is a diagonal matrix with the square roots of non-zero eigenvalues and U and V are the set of left and right orthonormal eigenvectors. Each eigenvalue represents the weight of its corresponding eigenvector. The eigenvector corresponding to the highest eigenvalue represents the direction of the maximum variation in the data, the eigenvector corresponding to the second highest eigenvalue represents the second maximum variation direction, and so on. Suppose a new data $B = \{\mathbf{I}(1), ..., \mathbf{I}(m)\}$ of dimensions $d \times m$ is received over time, where m is the number of new samples received. The SVD of the combination of $[A\ B]$ is represented by:

$$\begin{bmatrix} A & B \end{bmatrix} = U^{'} C^{'} V^{'\,\mathrm{T}}. \tag{3.7}$$

The computation of the SVD from scratch every-time new data is received is time consuming and impractical for applications like object tracking. This motivated the incremental updates of the eigenbases. The concatenation of A and B can be expressed in a partitioned form that utilize the already computed SVD of A as below (ROSS et al., 2008):

$$\begin{bmatrix} A & B \end{bmatrix} = \begin{bmatrix} U & \widetilde{B} \end{bmatrix} \begin{bmatrix} C & U^T B \\ 0 & \widetilde{B}^T B \end{bmatrix} \begin{pmatrix} V^T & 0 \\ 0 & 1 \end{pmatrix} \tag{3.8}$$

where $R = \begin{bmatrix} C & U^T B \\ 0 & \widetilde{B}^T B \end{bmatrix}$ is a square matrix of size $q + m$, q is the number of singular values in C and $\widetilde{\mathrm{B}}$ is a component of B orthogonal to U. To compute $\widetilde{B}$ and R, the QR decomposition method is used that decomposes matrix into an orthogonal matrix and an upper triangular matrix. Therefore, $\widetilde{B}$ and R can be obtained by QR decomposition of $[UC\ B]$ as:

$$\begin{bmatrix} UC & B \end{bmatrix} \stackrel{QR}{=} \begin{bmatrix} U & \widetilde{B} \end{bmatrix} R, \tag{3.9}$$

The SVD of R can be computed in constant time regardless of the initial data size. The SVD of R is then calculated as follow:

$$R = \widetilde{U}\widetilde{C}\widetilde{V}^T, \tag{3.10}$$

where $\widetilde{C}$ is a diagonal matrix containing singular values and $\widetilde{U}$ and $\widetilde{V}^T$ represents left and right eigenbases of R. Now, the SVD of $[A\ B]$ can be expressed as:

$$\begin{bmatrix} A & B \end{bmatrix} = \begin{bmatrix} U & \widetilde{B} \end{bmatrix} \begin{bmatrix} \widetilde{U}\widetilde{C}\widetilde{V}^T \end{bmatrix} \begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix}, \tag{3.11}$$

that can be organized as:

$$\begin{bmatrix} A & B \end{bmatrix} = \left(\begin{bmatrix} U & \widetilde{B} \end{bmatrix} \widetilde{U} \right) \widetilde{C} \left(\widetilde{V}^T \begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix} \right), \tag{3.12}$$

where I is an identity matrix, U'= $[U\ \widetilde{B}]\ \widetilde{U}$ and C' = $\widetilde{C}$. Since in the current tracking method and the incremental PCA, only U' and C' will be utilized, thus V' is not needed. The desired number of eigenbases associated with non-zero singular values can be selected for further processing, while other eigenvectors and singular values that are in excess can be discarded.

In applications based on a visual representation of objects, it is desirable to focus primarily on recent images due to various factors such as memory usage, processing cost, visual appearance or shape changes of an object. For instance, a human face may change its appearance due to occlusion, expression changes, pose changes, illumination changes, etc. So it can be implied that the recent observations are more indicative of the face appearance than the old ones. For this reason, assigning higher weights to the recent observations is desirable in an online learning scheme. Furthermore, all the observations

cannot be retained since the data increases with time, and it can become so large that relative contributions of recent blocks can be overwhelmed. This would make the learner blind to recent changes to the point that adaptivity is affected severely. However, to moderate the balance between the old and the new observations, a forgetting factor can be incorporated when computing the eigenbases. One way to introduce a forgetting factor is to multiply the singular values by the forgetting factor f (ROSS et al., 2008). The forgetting factor ($f \in [0\ 1]$) down-weights the contribution of the old data (e.g., $f = 0.5$, only carry half of the old samples contribution and $f = 1$, keeps all the contribution from the old data samples while updating the eigenbases).

3.2.3 Incremental Update of the Mean

The mean $\mu(t)$ of the observations at time t is important in computation of the PCA as well as in the proposed technique to detect the tracked target object (i.e., face). As the contribution of the old data is decreased in the eigenbases, therefore, it is required to update the contribution of the old data when computing the mean $\mu(t)$ at time t. The mean $\mu(t)$ at time t is calculated as shown below:

$$\mu(t) = \frac{f \cdot n \cdot \mu_n + m \cdot \mu_m}{m + f \cdot n}, \tag{3.13}$$

where μ_n represents the mean of the data matrix A and is given by:

$$\mu(n) = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{3.14}$$

where, x_i represents an observation (e.g., a tracked target face), and μ_m is the mean of the newly added observations B, $t = m + n$, and f is the forgetting factor. A benefit of having a forgetting factor is that the mean $\mu(t)$ at time t can still change in response to new observation, even if the total number of observations are very large.

3.3 Background on Dictionary Learning for Sparse Representation

Dictionary learning has shown significant improvement in signal reconstruction, instead of using off-the-shelf bases (MICHAL; MICHAEL; ALFRED, 2005). A disadvantage of representing the real-valued signals using Fourier or wavelet bases is that they

do not provide specialized representation for the particular dataset. The linear decomposition of data using few atoms of a learned dictionary instead of pre-defined bases led the state of the art in different areas of machine learning and image processing, such as image reconstruction (ELAD; AHARON, 2006), image denoising (MICHAL; MICHAEL; ALFRED, 2005), object recognition (FANELLO et al., 2013), texture analysis (PEYRÉ, 2009), which indicates that the sparse learned model can be useful in representation of natural images. While learning a dictionary has shown significant improvement in the state of the art in signal and image processing, the corresponding optimization problem is a significant computational challenge as well as memory, especially for a large amount of image data.

Despite the apparent advantages over fixed bases, there are some important issues to address while representing the data using dictionaries:

- What dictionary should be used so that a cost-effective solution can be found for an application at hand.
- A theoretical problem, if there exists a unique sparse representation? Also, it is possible that there is no unique sparse representation for the data, so the closest approximation is achieved using different techniques.

There are two stages in a dictionary learning algorithm that are repeated to achieve a dictionary that is capable of representing the data sparsely. These stages are the following: the first one is the sparse coding stage, which is finding a solution to the following problem:

$$min_{\alpha}||X - D\alpha||_2^2 + \lambda||\alpha||_0, \tag{3.15}$$

where, α represents sparse code and D is a fixed dictionary, the sparse coding problem can be solved using Eq. 3.15 as an approximation to L_0 regularization problem as explained in Section 3.3.1. The second problem is the dictionary update in which atoms are updated by changing sequentially each column d_i in order to better represent the data items mapped to the dictionary columns. This can be solved using the equation below:

$$min_{D}||X - D\alpha||_2^2, \tag{3.16}$$

where the subscript 2 $||.||_2$ represents the default L2 (Euclidean) norm $||x||_2 = \sqrt{(x_1^2 + x_2^2 + ... + x_n^2)}$. Consider x = $\{x_1, x_2, \ldots, x_n\}$ is a vector with n elements, $||x||_2^2$ is this number power 2 (the number multilpied by itself). The Idea is that

the vector that minimizes the $L2$ norm (a number) is the same that minimizes this number power 2 (i.e., $||x||_2^2 = x_1^2 + x_2^2 + \ldots + x_n^2$), and the last one is easier to evaluate. The atom updating stage in Eq. 3.16 is the characteristic that distinguishes many different dictionary learning methods. The dictionary $D \in \mathbb{R}^{d \times n}$, with n being the number of atoms and d the length of each atom, is "undercomplete" if $n < d$, which means that there are less number of atoms than the size of each atom, the dictionary is considered to be "complete" if $n = d$, and "overcomplete" if the number of atoms are higher than the length of each atom, i.e., $n > d$. The overcomplete dictionaries are the most common dictionaries for a sparse dictionary learning problem, because either they start big or multiple dictionaries are merged to make dictionaries. The objective of the dictionaries is to achieve the sparsest solution for the signal, that allows reconstructing the signal with high resolution that is not possible with the traditional non-adaptive approaches in linear time.

3.3.1 Sparse Representation

With a known dictionary D , the sparse representation of a signal x can be found using:

$$\begin{aligned} \alpha = min||\alpha||_0^0 \ , \\ s.t. \ x = D\alpha. \end{aligned} \tag{3.17}$$

It is possible that α is not unique, so the goal is to select the sparse representation that minimizes the reconstruction error:

$$\begin{aligned} min||\alpha||_0^0 \ , \\ s.t. \ ||D\alpha - x||_2^2 \leq \epsilon^2. \end{aligned} \tag{3.18}$$

Solving the above equation is a NP-hard problem because of its time complexity. However, there are two most common approximations which are used. The first one, is to relax the condition $min||\alpha||_0^0$, called Basis Pursuit (CHEN; DONOHO; SAUNDERS, 2001; DONOHO; ELAD, 2003), and use continuous optimization techniques. Second approximation builds a solution one element at a time in a greedy approach called Matching Pursuit (MALLAT; ZHANG, 1993).

3.3.2 Basis Pursuit(BP)

Basis Pursuit (BP) decomposes a signal into an optimal decomposition of the dictionary elements, such that the achieved decomposition has the smallest $l1$ norm among all the decompositions. Instead of solving L_0 norm in Eq. 3.15, the BP method solves the L_1 norm. There are several efficient ways to solve a quadratic problem. The equation takes the form (DONOHO; ELAD, 2003):

$$\begin{aligned} min||\alpha||_1 \ , \\ s.t. \ ||D\alpha - x||_2^2 \leq \epsilon. \end{aligned} \tag{3.19}$$

Due to the recent advances in linear programming methods (such as associated with interior-point methods) and close formulation of BP to linear programming, it is possible to solve BP in nearly linear time with certain dictionaries (CHEN; DONOHO; SAUNDERS, 2001).

3.3.3 Matching Pursuit (MP)

The Matching Pursuit is based on a greedy approach that starts from the initial approximation and builds up a sequence of sparse approximations in steps by selecting one atom at a time (MALLAT; ZHANG, 1993). This is achieved by minimizing the reconstruction error while adding one atom at a time. The Matching Pursuit technique chooses one atom at each step which reduces the reconstruction error, and its addition may help achieve higher sparsity. The iterative procedure is explained as follows:

1. Find an atom that best matches the signal, i.e., the one with the smallest error and keep it.

2. Given previously found atom, find the next best fit.

3. Keep on finding the next atom until the error $||D\alpha - x||_2^2$ is less than a certain threshold or a maximum number of iterations is reached.

An enhancement in the matching pursuit method called Orthogonal Matching Pursuit (OMP) re-evaluates the coefficients of the selected atoms by using the least square error after each time a new atom is selected (RUBINSTEIN; ZIBULEVSKY; ELAD, 2008). The algorithm changes the values of the coefficients of the atoms to reduce the error and

finds the best values for the selected atoms. In the MMDL based face tracking operating mode of the proposed tracker, OMP is used for computing the sparse representation of the data over the specified dictionary.

3.3.4 Dictionary Learning

For dictionary learning, it is assumed that the good-behaved images have a sparse representation. One approach to choose D is from a known set of transforms such as frequency dictionaries are based on Fourier or cosine transforms. Also, time-scale dictionaries have been explored such as wavelets, curvelets, contourlets, etc. Time-Frequency dictionaries are also explored such as Gabor dictionary. To learn a dictionary, it is initialized usually with a subset of data or cosine transform computed over the data which is to be represented sparsely using a dictionary. To learn a dictionary, the Eq. 3.16 leads to the following optimization problem:

$$\begin{aligned} min_D \sum_{j=1}^{P} ||D\alpha_j - x_j||_2^2, \\ s.t.\ \forall_j ||\alpha||_0 \leq L, \end{aligned} \tag{3.20}$$

where P is the total number of images in the data for which sparse code α is computed with L number of non-zero elements in α and $||.||_0$ represents L_0 norm which counts the number of non-zero elements in a matrix.

The dictionary D can be learned online or offline. The offline dictionaries are learned from the training data that is already available before building a dictionary. These dictionaries can handle a limited amount of data or otherwise require infinite memory. To solve this problem, data is provided in batches to update the dictionary so that it can handle the large size of data. Dictionary can also be learned online from the data available along time. This way dictionary can handle a large amount of data and can adapt to the changes in the appearance and shape of the object. For example, in video processing, there is a lot of data that is received over time. Discarding this data without using is analogous to a waste of information. Also, the dictionaries trained offline are not adaptive to the changes in the environment. For these reasons, it is important to learn a dictionary online and use valuable information which is received during runtime in the form of data.

Olshausen et al. (OLSHAUSEN; FIELD, 1996) first proposed to utilize dictio-

nary training methods with the motivation to capitalize the statistical regularities in the natural images. The authors suggested that sparsity may provide efficient coding for a natural scene. This method is subsequently extended by Lewicki et al. (LEWICKI; SEJNOWSKI, 2000). Kreutz et al. (KREUTZ-DELGADO et al., 2003) proposed an important contribution to the training of sparse representation of dictionaries that focused on the relationship among the sparse coding dictionary design and the vector quantization problem. Also, a generalization of the well known K-Means algorithm was proposed. The K-SVD algorithm (MICHAL; MICHAEL; ALFRED, 2005) uses a different approach for the generalization of the K-means algorithm and gathers a lot of attention in the dictionary learning research. The K-SVD algorithm is a direct generalization of the K-means algorithm (GERSHO; GRAY, 2012) and it can be easily used with any pursuit algorithm. In this work, we propose an improvement in the K-SVD dictionary by introducing an incremental update procedure, so that it can be utilized in large datasets and in applications such as object tracking where the test data is useful to adapt to the changes in the appearance of the tracked target over time.

3.3.5 Incremental Dictionary Learning

The K-SVD has shown good performance on small datasets (MICHAL; MICHAEL; ALFRED, 2005) due to its good model with clear mathematics and physics meanings. However, it is impossible to employ this model for large datasets because of the memory limit to store all the data at the same time as well as to perform SVD computation of very large datasets. The incremental learning techniques can help in learning a dictionary for large datasets (WANG et al., 2014) by introducing new atoms in the dictionary in order to represent the new data available.

Considering a dataset $X_n = \{x_1, x_2, ..., x_n\}$, and the dictionary $D_n = \{d_1, d_2, ..., d_n\}$ and sparse code α_n, when a new set of data samples $X_m = \{x_{n+1}, x_{n+2}, ..., x_{n+m}\}$ are received, they need to be represented sparsely, so the new sparse code α_{m+n} needs to be computed. As D_n already represents X_n sparsely as α_n, it is interesting to test if D_n is able to represent the new data X_m. In this case, only the coefficients α_m need to be computed, which belong to the sparse representation of X_m. However, this is not usually the case; therefore the dictionary should also be updated. Therefore the new objective

function of the updated dictionary is given by:

$$min_{\alpha}||X_{n+m} - D_{n+m}\alpha_{n+m}||_2^2 + \lambda||\alpha_{n+m}||_0. \tag{3.21}$$

It should be noted that, for consistency every time the dictionary is updated, the number of new samples used to update the dictionary should be the same i.e., $X_m = \{x_{n+1}, x_{n+2}, ..., x_{n+m}\}$, where the number of new samples m is constant for each dictionary update. As there are m new samples, the coefficients α_m for X_m are $\{\alpha_{n+1}, \alpha_{n+2}, ..., \alpha_{n+m}\}$. Now, all the coefficients of X_{m+n} at time t are given by $\alpha_{n+m} = \{\alpha_1, \alpha_2, ..., \alpha_n, \alpha_{n+1}, \alpha_{n+2}, ..., \alpha_{n+m}\}$.

While solving the Eq. 3.21, the j^{th} column of α is denoted by α_T^j as in the classical K-SVD. For an arbitrary kth atom of the dictionary D_t at time t, the first term of the objective function in Eq. 3.15 takes the form (WANG et al., 2014):

$$||X_{n+m} - D_t\alpha_{n+m}||_2^2 = ||\alpha_{n+1} - \sum_{j=n+1}^{k-1} d_j\alpha_T^j - \sum_{j=k+1}^{n+m} d_j\alpha_T^j - d_k\alpha_T^k||, \tag{3.22}$$

where, $n+1 \leq k \leq n+m$, which means the atoms are added to only accommodate the new data/images. The atoms updated in the dictionary are $d_{n+1}, ..., d_t$, where $t \leq n+m$. In Eq. 3.22, the current samples X_m and the old atoms are linked into one objective function. The Eq. 3.22 can be re-written as below:

$$||X_{n+m} - D_{n+m}\alpha_{n+m}||_2^2 = ||E_m^k - d_k\alpha_T^k||, \tag{3.23}$$

where, $E_m^k = \alpha_{n+1} - \sum_{j=n+1}^{k-1} d_j\alpha_T^j - \sum_{j=k+1}^{n+m} d_j\alpha_T^j$ is an error matrix which shows how well the new dictionary without a specific atom d_k represents the newly added data while the information from the the old dictionary D_n is still associated.

The next issue is the initial value of the new atoms to be added. We propose to update possibly all the atoms of the dictionary by initializing the update using the incremental SVD (ROSS et al., 2008) unlike proposed in (WANG et al., 2014) that just updates the new atoms in the dictionary. Hence, our approach modifies the dictionary update in a way that it adapts to the current appearance of the tracked target object. Therefore, the old atoms of the dictionary are also changed, and a forgetting factor is also applied to reduce the influence of the old samples. This initialization and estimation of the new atoms values are explained next.

3.3.6 Estimating New Dictionary Atoms

It is possible to add new atoms to the current dictionary; however, it is an important task to set the initial values of the new atoms. It is also important to check if it is possible to represent the newly added data with the old dictionary D_n sparsely and efficiently. In such a case, there is no need to update the dictionary. However, this is not usually the case, as newly added images most of the time have slightly different information and require the dictionary to be updated in a way that these new images can also be represented efficiently. Selecting the initial value of the new atoms is very critical. In case of inappropriate values of new atoms, the training process will be slow and inefficient. The most common method is to use the set of images as an initial value, which is a slow and inefficient technique. Other methods such as (WANG et al., 2014) perform the sparse coding for X_m using the already existent dictionary D_n to represent the current samples and select the samples whose coefficients are not sparse. This is done by using the following equation:

$$min_{X_n}||X_m - D_n\alpha_n||_2^2 + \lambda||X_n||_0. \tag{3.24}$$

Now, among all the samples whose coefficients are not sparse enough, the samples with maximum disagreement between the different atoms are selected as the initial values of the new atoms. This is achieved using the entropy theory as below:

$$max_{\alpha_i \epsilon \alpha_m} H(\alpha_i), \tag{3.25}$$

where, the entropy $H(\alpha_i)$ gives the disagreement or lack of order as below:

$$H(\alpha_i) = \sum_{j=1}^{n} p(l = d_j|\alpha_i) log(p(l = d_j|\alpha_i)), \tag{3.26}$$

where,

$$p(l = d_j|\alpha_i) = \frac{\alpha_i(j)}{\sum_{b=1}^{n} \alpha_i(b)}. \tag{3.27}$$

With this initial value, the new atoms are found by repeating SVD k times so that the desirable sparsity is achieved using Eq. 3.15.

The initialization of new atoms is important to keep the dictionary up to date. However, only adding the new atoms can lead to an infinite dictionary, which is not suitable for applications that use small memory and processing power such as target object

tracking. For problems such as tracking, this dictionary learning scheme cannot be applied directly. Adding more atoms every time a new data is received can lead to an immensely large dictionary which may affect the adaptivity of the dictionary severely. This could be solved by limiting the maximum number of atoms in the dictionary at any point in time. It is worth mentioning that the tracked target objects (e.g., non-rigid objects) change the appearance and shape over time, so in order to track the target efficiently, the dictionary should be able to adapt to the recent changes in the target object. This would allow the dictionary to represent the target effectively. For this reason, the contribution of old data must be minimized.

As the recent observations are more indicative of the current shape of the object, more recent data should receive high weights. This can be done by assigning higher weights to the newly added samples while updating the dictionary or embedding a forgetting factor. In the proposed system, a forgetting factor $f \in [0, 1]$ is used in the update of the dictionary to down-weight the contribution of the old samples. As the dictionary is based on the SVD, the dictionary is updated using the incremental update in the SVD (ROSS et al., 2008). This will be the initial value of the dictionary atoms, and the dictionary is optimized until the desired sparsity is achieved.

3.4 Classification Methods and the Support Vector Machine (SVM) Classifier

In machine learning, a classification method assigns a class to a new observation among the set of predefined classes, based on the training set of observations. Several methods have been explored for classification applied to images, which are reviewed by Wang et al. in (WANG et al., 2017). The algorithm that categorizes the data into their corresponding classes is called a classifier. The classifiers can be categorized into supervised and non-supervised. Usually, the term non-supervised classification is referred to as clustering and group the observations by using some inherent similarity measure. Supervised classifiers utilize the training dataset with the sample labels to identify the class of the new observation.

Several classifiers have been proposed in the literature and are used to classify different observations, such as k-nearest neighbor (k-NN) applied to face classification from non-faces (JOSE; POORNIMA; KUMAR, 2012), Random Forest (SEE et al., 2017) also used for face classification, Decision Tree (DT) utilized for automatic diagnostic of

ophthalmic images that is used in the treatment of the eyes (TAHIR et al., 2006), Boosted Trees (BT) used in the detection of fetal anatomies from ultrasound images (CARNEIRO et al., 2008), and SVMs utilized for automatic diagnosis of ophthalmic images (WANG et al., 2017) methods. Among these classifiers, the SVM classifier has produced a desirable performance in image processing applications (YANG et al., 2016; WANG et al., 2017).

The SVM classifier is a supervised classification technique that utilizes the training data to build a linear model to categorize new data observations. SVMs can also perform non-linear classification by implicitly mapping their inputs into a high dimensional space. In the proposed tracking scheme, the SVM is utilized in the Constrained Local Model (CLM) to create a patch classifier which works as a patch expert and classifies the positive and negative samples, which is a binary classification problem. To build a binary classifier, a linear Support Vector Machine (SVM) is trained with positive and negative samples. The goal of the SVM is to design a hyperplane that classifies all the training vectors into two classes. The best choice is the hyperplane that leaves maximum margin between the two classes.

Suppose there are k training vectors $\{v^{(1)}, v^{(2)}, \ldots, v^{(k)}\}$, and each training sample $v^{(c)} = \{v_1^{(c)}, v_2^{(c)}, \ldots, v_l^{(c)}\}$ is a column vector of l dimensions. An input value for class attribution of $u^{(c)} = [-1, 1]$ must be assigned for each training sample $c = \{1, 2, ...k\}$. The SVM classifier output is written as a linear combination of input vector elements:

$$u^{(c)} = \Omega^T v^{(c)} + \Theta, \tag{3.28}$$

where $\Omega^T = [\Omega_1, \Omega_2, \ldots, \Omega_Z]$ represents the weight of each element of the input data, and Θ is a constant acting as a bias to prevent overfitting. The hyperplane is called SVM and is created in a way that each training output $u^{(c)}$ is set to 1 if the training observation is a positive sample, and -1 otherwise.

3.5 Constrained Local Models (CLM)

Constrained Local Models (CLM) tend to be the accurate facial feature detectors, on the other hand, they are known to have a slow convergence process, making their use in the face tracking quite challenging. Nevertheless, if CLM is used less often in comparison to other components of the tracking process, the tracking system could be viable for real-time operation. Since the proposed tracking method is applied to faces,

so the CLM is discussed here in the context of a face and facial landmark localization. In practice, CLM consists of two stages (modules), which are CLM model building and CLM search (LUCEY et al., 2009), as discussed next.

3.5.1 CLM Model Building

A CLM model is comprised of two models: (a) a shape model, that takes in shape information, and (b) a patch model that considers the local patch information. Both models combine to a comprehensive model to represent the target object (i.e., a face). Images of the cropped faces and a set of facial feature points are used as the training data to build the CLM face model.

Shape Model: In order to build the CLM shape model, all the shapes are aligned with the first (initial) shape of the training set using procrustes analysis (KENDALL, 1989), which removes the adverse effects of shape variations in terms of scale, translation and rotation, leaving only the intrinsic face shape variations. On these aligned faces, the PCA is performed to capture the face variations (eigenvectors) in the training data, and to obtain an indication of the total face variation by the eigenvalue of each eigenvector. Suppose a training shape matrix $M = [S_1, S_2, \ldots, S_N]$ of size $e \times N$, where each column is a face shape vector $S_r = [x_1, y_1, x_2, y_2, \ldots, x_Z, y_Z]^\top$ of size $e = 2Z$ (in current experiments, $Z = 68$) which contains the Z facial landmarks represented by their coordinates (x_i, y_i) and N is the total number of training face shape vectors. In order to compute the eigenvalues and eigenvectors, the mean $\overline{S}$ of these shape vectors is subtracted from each column S_r of the training shape matrix M to center them around zero, i.e., $\forall_{S \in M} \hat{S}_r = S_r - \overline{S}$. These zero mean shape vectors are combined to form a single matrix $\hat{M} = [\hat{S}_1, \hat{S}_2, \ldots, \hat{S_N}]$, and the covariance matrix $M_c = \hat{M}\hat{M}^T$ is utilized to compute the eigenvectors P and eigenvalues λ. Therefore, each shape can be written as a linear combination of the eigenvectors P and the mean shape ($\overline{S}$) as:

$$S = \overline{S} + P_s H, \tag{3.29}$$

where $H = P^T \hat{S}$ is a column vector containing the coefficients of the corresponding eigenvectors P for representing the shape in M.

Patch Model: In order to build a patch model for each feature point, a linear Support Vector Machine (SVM) (CORTES; VAPNIK, 1995) is trained with positive and

negative samples as a patch classifier. The negative examples are randomly selected from patches captured elsewhere in the training image (i.e., outside of the face). Positive examples are patches captured only from the face block around the particular landmark for which SVM is being trained. Afterward, the linear SVM is trained to discriminate between the positive and negative examples. This training process produces an equal number of linear support vectors that are used to tell whether a patch corresponds to a positive or a negative example class. It is worth mentioning that, for each feature point c, a linear SVM is trained with positive and negative examples. In other words, if there are a total of Z feature points, Z linear SVMs will be trained that can be considered as patch experts for that particular feature point.

Suppose there are k training vectors $v^{(1)}, v^{(2)}, \ldots, v^{(k)}$, and each training sample $v^{(c)} = (v_1^{(c)}, v_2^{(c)}, \ldots, v_l^{(c)})$ is a column vector of l dimensions. An input value for class attribution of $u^{(c)} = [-1, 1]$ must be assigned for each training sample $c = \{1, 2, \ldots, k\}$. SVM classifier output is written as a linear combination of input vector elements as:

$$u^{(c)} = \Omega^T v^{(c)} + \Theta, \tag{3.30}$$

where $\Omega^T = [\Omega_1, \Omega_2, \ldots, \Omega_Z]$ represents the weight of each element of the input data, and Θ is constant acting as a bias. In case of CLM, the set of training data is constituted by patches extracted from the training images, so the training output $u^{(c)}$ is set to 1 if the sample comes from positive feature points, and -1 otherwise.

3.5.2 CLM Search

The CLM search process combines the shape and patch models in such a way to detect feature point locations of a face shape with high SVM response. Given a set of initial feature points (landmarks), the CLM model is used to generate a set of gray value texture patches. The face shape templates are applied to the search image, and the response images are computed by the SVM based patch model, while at the same time maintaining the shape constraints. Both of these goals are combined using the following objective function:

$$f(S_t) = \sum_{i=1}^{Z} \imath_i(x_i, y_i) - \beta \sum_{j=1}^{o} \frac{-h_j^2}{\lambda_j}, \tag{3.31}$$

where, $\imath_i(x_i, y_i)$ is the patch of size $\sqrt{l}\times\sqrt{l}$ that is selected by the SVM based patch model in the $g \times g$ neighborhood of the current location of the landmark i ($g = 8$ in our experiments). The term $\sum_{j=1}^{o} \frac{-h_j^2}{\lambda_j}$ is the shape constraint, where $h_j \in H$ is the corresponding eigenvector coefficient in the eigenvectors representation H, λ_j is its eigenvalue and o is the number of eigenvectors, whereas the parameter $\beta \in [0\ 1]$ is a bias determining the compromise between shape fit and the SVM based patch model. The Eq. 3.31 is optimized using quadratic programming to obtain the optimum landmarks locations (i.e., shape) using an error measure, or until the maximum number of iterations is reached (LUCEY et al., 2009). In this work, the following error measure is used to describe the optimum landmark locations:

$$\epsilon_{rec} = S_r - PP^T S_r, \tag{3.32}$$

where S_r represents the current shape and P is the eigenvectors matrix from the CLM shape model.

3.6 Terminology

Table 3.1 provides the symbols used in this document with their meanings.

Table 3.1: Table of symbols

Symbol	Meaning	Symbol	Meaning
A	data matrix	**I**	Observation (target face)
d	size of observation (e.g. 32 x 32=1024)	n	number of observations in A
B	Newly received data	m	number of observations in B
$\sum$	eigenvalues of data matrix A	U and V	left and right orthonormal eigenvectors of A
$\sum'$	eigenvalues of data matrix $[A\ B]$	U' and V'	left and right orthonormal eigenvectors of $[A\ B]$
$\widetilde{\mathrm{B}}$	component of B orthogonal to U	U^T	Transpose of U
μ	mean	μ_n	mean of data A
μ_m	mean of data B	f	forgetting factor

Continued on next page

Table 3.1 – *Continued from previous page*

Symbol	Meaning	Symbol	Meaning
Z	number of feature points (Z=68)	S	shape
$\overline{S}$	mean shape	P_s	eigenvectors s= no of eigenvectors s= used
H	Coefficients corresponding to eigenvectors	k	no of training vectors for SVM
v	training sample for SVM	Z	total number of landmarks
u	label of training sample for SVM	Ω	weights of sample of patch model
Θ	bias for patch model		
$f(p)$	CLM search function	c	feature number
$\imath_c$	response of patch model	(x_c, y_c)	location of landmark
b_j	weight of eigen vector j	λ	eigenvalues
β	bias for choosing relative weight of patch and shape model	$\chi(t)$	affine parameters
$\mathcal{I}(t)$	set of images at time t	$p(\chi(t)\|\mathcal{I}(t))$	probability of affine parameters given data $\mathcal{I}$ at time t
(x_t, y_t)	translation parameters	s_t	scale
$\theta(t)$	rotation parameter	$\alpha(t)$	aspect ratio
$\phi(t)$	skew direction	t	time
$\psi(t)$	diagonal matrix with variances of affine parameters	δ	distance of sample from reference
δt	distance from sample to subspace	δw	within the subspace from projected sample to the subspace center
p	probability	ϵI	additive noise

Continued on next page

Table 3.1 – *Continued from previous page*

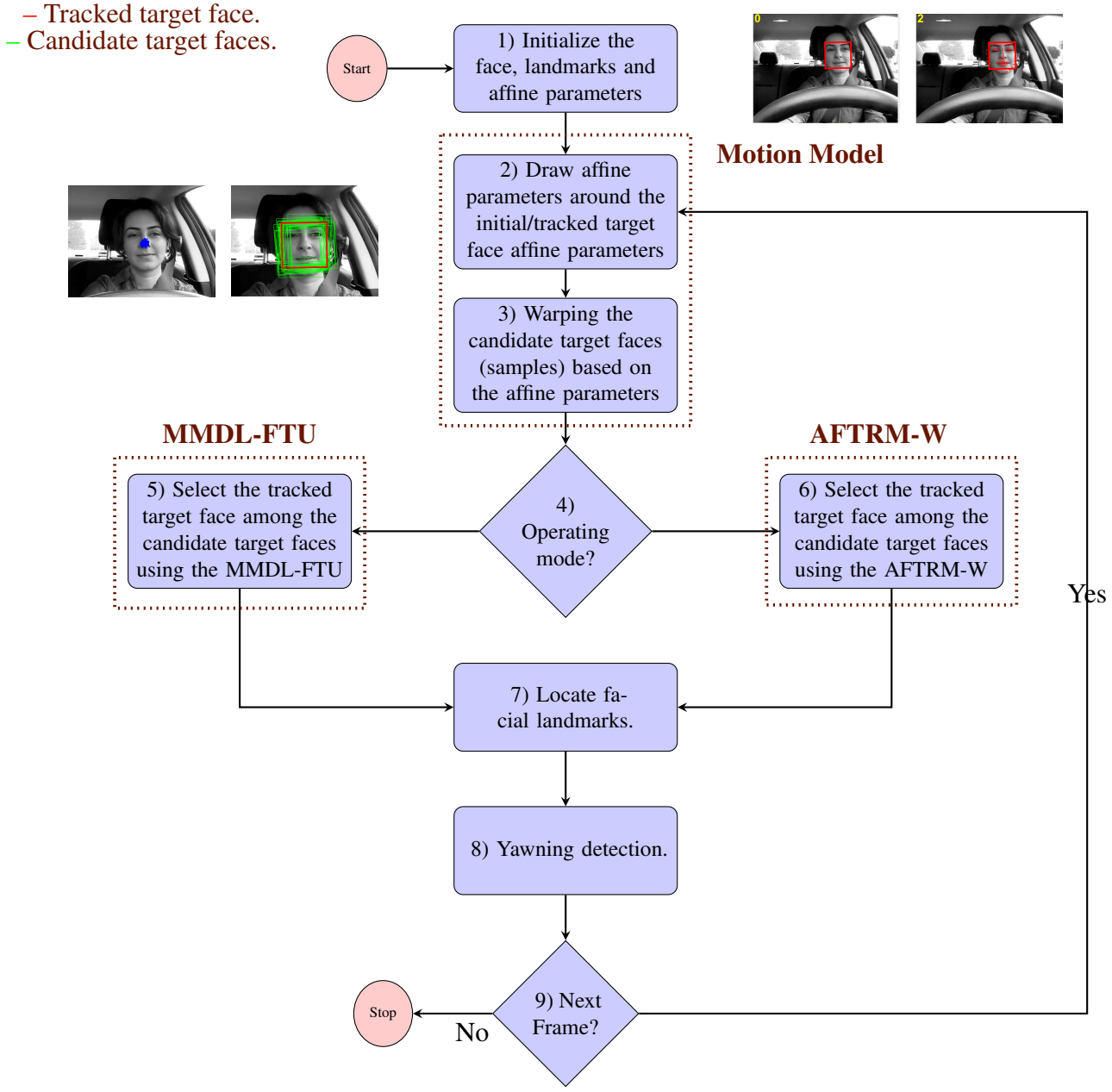
Symbol	**Meaning**	**Symbol**	**Meaning**
N	Gaussian distribution	$I(t)$	video frame at time t
Δ	Difference in the tracked landmarks	i	landmark number
TP	position of landmark in (x, y) estimated by proposed	Λ_G	groundtruth position of landmark in (x, y)
Γ_T	Median of Δ	Ψ	flag for Re-syncing
$\varepsilon(t)$	Track error at t	Z	total no of landmarks
NBC	no of black pixels in current frame	NWC	no of white pixels in the current frame
NBR	no of black pixels in reference frame	TP	true positive
FP	false positive	TN	true negative
FN	false negative	TPR	true positive rate
TNR	true negative rate	CDR	correct detection rate
r	no of resyncs required	κ	bias for tracking error

4 PROPOSED METHODOLOGY

This chapter provides the details of the proposed adaptive face tracking scheme and its operating modes. The face tracking scheme consists of two steps: the first step is modeling the motion and sampling the candidate face samples, and the second step is the selection of the tracked target face by using the observation model that models the appearance of the tracked target face. The appearance model of the proposed method is based on an incremental learning scheme. Furthermore, it estimates the tracking error during tracking, and a resyncing procedure is proposed to correct the tracking process if the proposed tracking predictor estimates high error. Figure 4.1 shows the block diagram of the proposed tracking method and is explained below:

- Block 1: In the first frame, the initial target face, the affine parameters ($\chi(t)$), while the facial landmarks are provided by a face landmark localization method (LUCEY et al., 2009);
- Block 2: In the subsequent frames, a finite number (η) of affine parameters are drawn around the affine parameters of the initial/tracked target face using a Gaussian distribution (see Eq. 4.2);
- Block 3: To locate the tracked target face in the frame at time t, the candidate target face samples (u $\times$ u) are warped according to the computed affine parameters to be compared with the tracked target face (see the example in Fig. 4.6 at the left of Block 3: in red, the tracked target face from the previous frame; in green, candidate target face samples). See details in Section 4.1;
- Block 4-6: Among the candidate target face samples, the tracked target face is selected using one of the operating mode, i.e., Multi-Model Dictionary Learning for Face Tracking with dictionaries Update (MMDL-FTU) or An Adaptive Face Tracking Scheme using Resyncing Mechanism with Weighted CLM search (AFTRM-W);
- Block 7: The facial landmarks of the tracked target face are localized;
- Block 8: Yawning is detected;
- Block 9: Finally, if there are more frames to process the affine parameters of the current tracked target face are used in the next frames, and the process re-starts from Block 2.

Figure 4.1: Block diagram of the proposed face tracking method.



Both the operating modes of the tracking scheme shares the same motion model which is explained in Section 4.1, and differs by how the representation of the tracked target face is modeled, which are explained in Sections 4.2 and 4.3.

4.1 Motion Model and Sampling

Visual tracking can be designed using a Markov model with hidden state variables (ROSS et al., 2008). In the current experiments, the tracker is applied to track the face in a video sequence. For tracking the face, state of the tracked target face is de-

scribed by a variable $\chi(t)$ that contain the affine parameters which describes the location of the object at time t. Furthermore, the affine parameters of the tracked target face are used to estimate the face landmarks, detect yawning and calculate the tracking error (see Eq. 4.19). For a set of tracked target face samples at time t, $\mathcal{I}(t)$=$\{\mathbf{I}(1),\mathbf{I}(2)...,\mathbf{I}(T)\}$, the face tracker estimates the hidden state variable $\chi(t)$ using:

$$p(\chi(t)|\mathcal{I}(t)) \propto p(\mathbf{I}(t)|\chi(t))\times \int p(\chi(t)|\chi(t-1))p(\chi(t-1)|\mathcal{I}(t-1))d\chi(t-1). \tag{4.1}$$

The candidate target face samples that may contain the tracked target face are sampled following the motion model between two states $p(\chi(t) \mid \chi(t-1))$, assuming a Gaussian distribution around the tracked target face location in the previous frame. At time t, the state of the target face in a video sequence is described by the affine parameters $\chi(t)$=($x(t)$, $y(t)$, $s(t)$, $\theta(t)$, $\beta(t)$, $\phi(t)$), where $x(t)$ and $y(t)$ represent the translation of the tracked target object (i.e., face) with respect to the origin of the image, $s(t) = M/u$ is the scale of the tracked target face w.r.t the size of the image $(M \times N)$ which contains the tracked target face ($u \times u$), whereas $\theta(t)$, $\alpha(t)$ and $\phi(t)$ are the rotation angle w.r.t the horizontal axis, the aspect ratio, and the skew direction, respectively. It is worth mentioning that, the aspect ratio $\alpha(t)$ is used along with the scale $s(t)$ to maintain the scale of the tracked target object in xy-coordinate system in the image space similar to the approach used by Ross et al (ROSS et al., 2008). The dynamics of each parameter in $\chi(t)$ is modeled independently by a Gaussian distribution centered at $\chi(t-1)$, and going from $\chi(t-1)$ to $\chi(t)$ is given by (for details on Gaussian distribution, see Section. 3.1):

$$p(\chi(t)|\chi(t-1)) = \mathcal{N}(\chi(t); \chi(t-1), \psi(t)), \tag{4.2}$$

where $\psi(t)$ is a diagonal matrix with each element representing the variance of its corresponding affine parameters element, and $\mathcal{N}$ represents a Gaussian distribution. These affine parameters are used to warp the candidate target face samples that may contain a face in the current frame. These candidate target face samples are tested for quality using the appearance model, and one of them is selected as the tracked target face in the current frame at time t using the techniques explained in Sections 4.2 and 4.3.

Figures 4.2, 4.3 and 4.4 show an example of the working of the motion model of the tracking process, i.e., how the affine parameters are distributed to generate the candidate target face samples in the current frame. The affine parameters $\chi(t)$ are represented

by a point in affine parameter space, as shown in Figures 4.3. The affine parameter space is a six dimensional space, and only three dimensions are shown in Figure 4.3. The red point in the Figure 4.3 represent the affine parameters of the tracked target face in the previous frame. Numerous affine parameters are computed using the Gaussian distribution centered around the affine parameters of the tracked target face in the previous frame using Eq. 4.2, and these affine parameters are shown as blue points in Figure 4.3. Furthermore, these affine parameters are used to warp the candidate target face samples which may contain the tracked target face in the current frame, which are shown in green color faces in Figure 4.3.

This phenomenon in the image space going from the previous frame to the current frame is shown in Figure 4.2, where red bounding box shows the tracked target face in the previous frame, and green bounding boxes show candidate target face samples in the current frame. Furthermore, the working of the motion model from the previous frame to the current frame is demonstrated in Figure 4.4. Figure 4.4 (a) shows the tracked target face in the previous frame and Figure 4.4 (b) shows the corresponding candidate target face samples using these affine parameters in the current frame. Figure 4.5 shows an image with the bounding box of the tracked target face from the previous frame (red color bounding box) and the candidate target face samples of the current frame (green color bounding boxes). Among these candidate target face samples, the tracked target face is computed using one of the operating modes which are MMDL-FT operating mode and AFTRM operating mode. The operating mode is the algorithm that selects the tracked target face among the candidate target face samples.

Figure 4.2: Motion model example ($p(\chi(t)|\chi(t-1))$) in image space; Affine parameter space, each point in affine parameter space is warped into a bounding box in the image

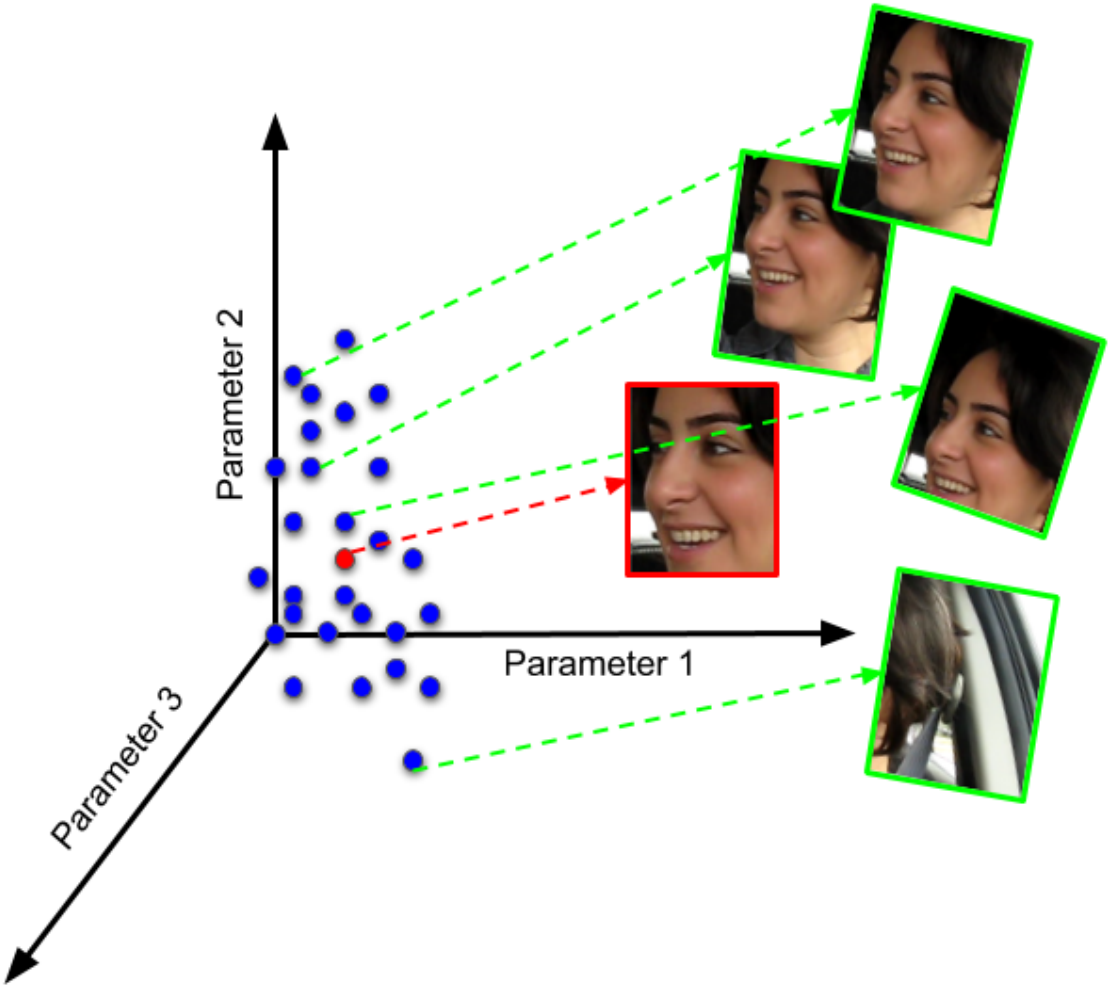


Figure 4.3: Motion model example ($p(\chi(t)|\chi(t-1))$); Tracked target face bounding box

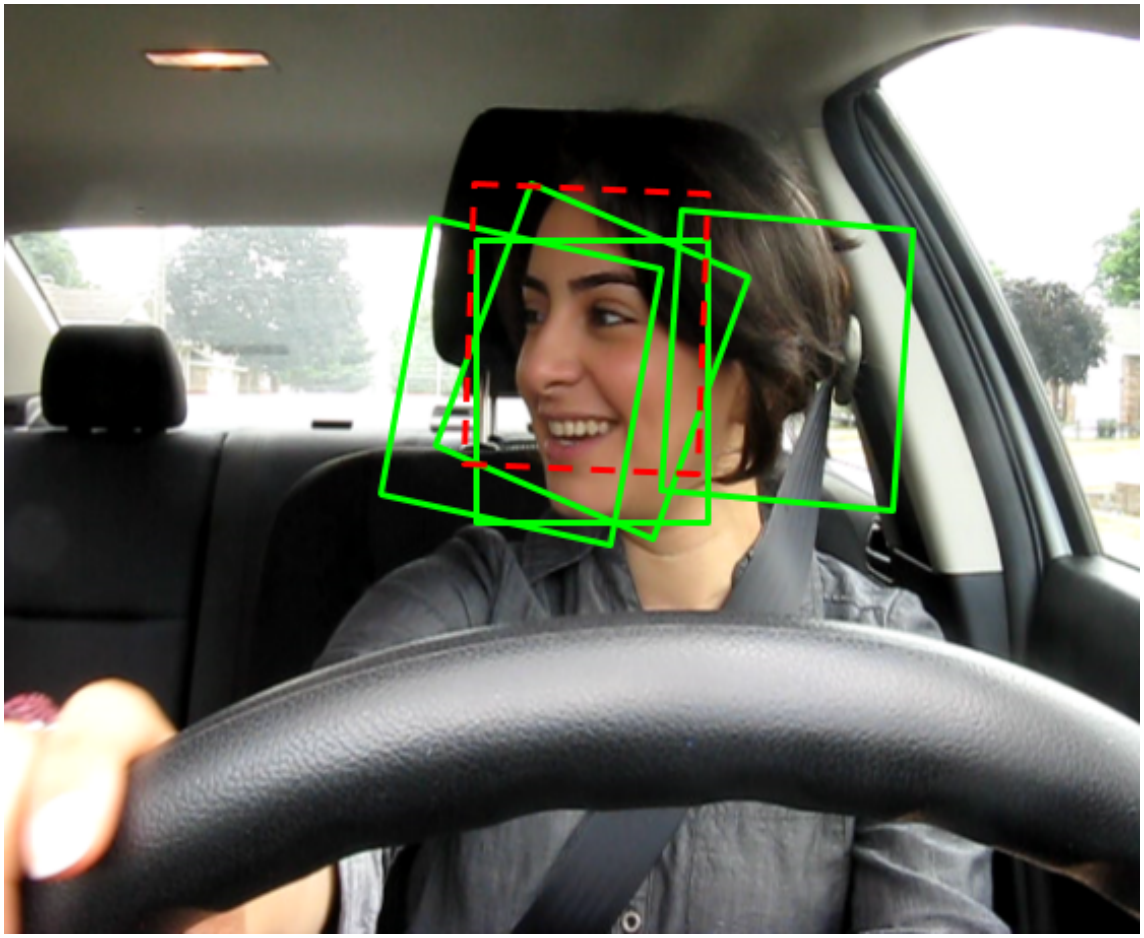
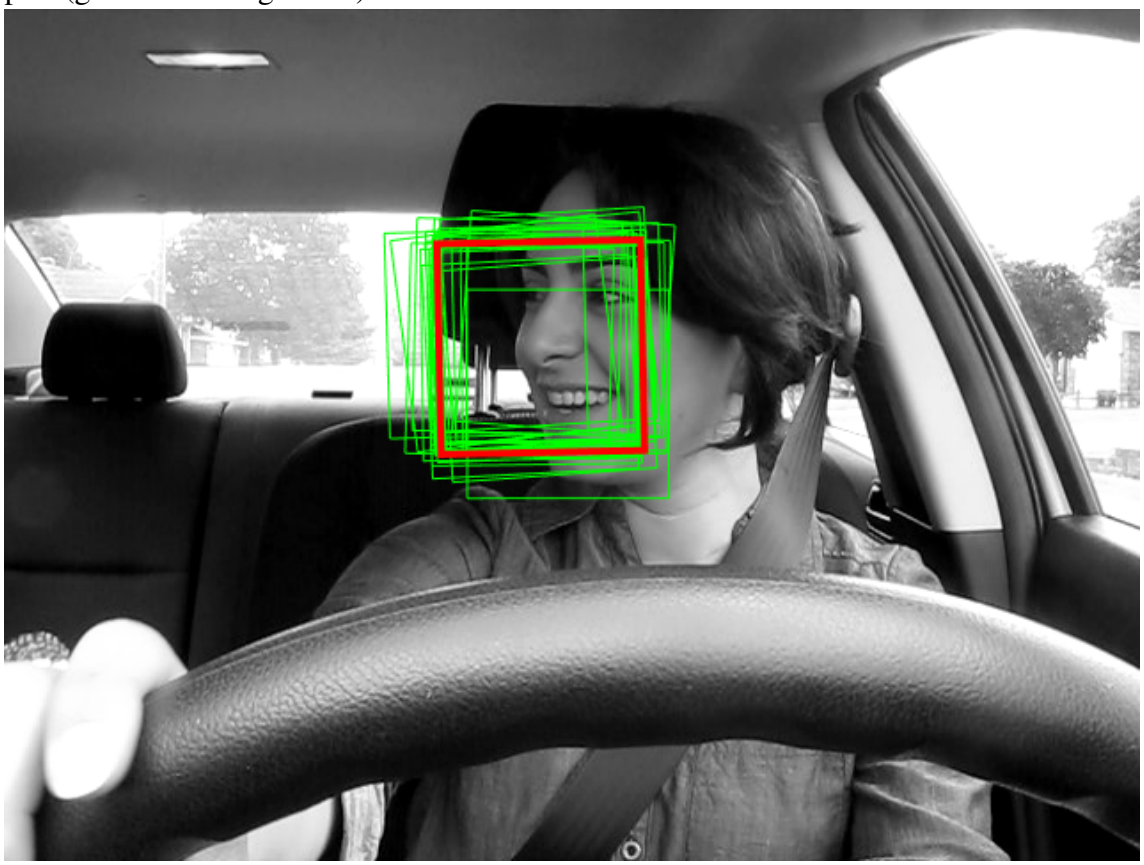


Figure 4.4: Motion model example going from the previous frame to the current frame ($p(\chi(t)|\chi(t-1))$).

(a) Tracked target face bounding box; (b) Candidate target face samples bounding boxes.

Figure 4.5: Tracked target face sample (red bounding box) and candidate target face samples (green bounding boxes).



4.2 Proposed Methodology for Multi-Model Dictionary Learning for Face Tracking (MMDL-FT) and MMDL-FT with Update Test (MMDL-FTU)

This section explains MMDL-FTU operating mode of the proposed face tracking method. There are two techniques used for the training data collection to build and update the dictionaries incrementally. In the MMDL-Face Tracker (MMDL-FT), the dictionaries are updated using the tracked target face samples collected without checking their quality, as proposed by Ross et al (ROSS et al., 2008). In the MMDL-Face Tracker with Update Test (MMDL-FTU), only those tracked target face samples are collected which has reconstruction error smaller than a specific threshold ε.

4.2.1 MMDL-FTU Operating Mode

Dictionary learning has been explored in object tracking, but the proposed dictionary learning methods usually are based on static dictionaries that are not updated during

object tracking (LIU; LI; FANG, 2015). Most of the methods that use dictionaries for object tracking are focused on the representation of the target (CHENG et al., 2014), or on the discrimination between the target and the background (XIE et al., 2014). In this work, a new approach called Multi-Model Dictionary Learning (MMDL) is proposed for face tracking that builds in parallel two dictionaries based on the k-SVD (i.e., a classification dictionary and a reconstruction dictionary), and combines them into a single multi-model dictionary.

The reconstruction dictionary (D_p) is used to estimate the appearance difference between the reconstructed sample and the candidate target face sample, whereas the classification dictionary (D_c) is utilized to discriminate the candidate target face from the background. These two dictionaries are combined into a single multi-model, which tends to improve the tracking robustness. Therefore, the proposed MMDL scheme can reconstruct the face in addition to discriminating the face from the background. The proposed method learns the face appearance using dictionary atoms constructed from patches, that are taken from positive and negative samples of the training data. Furthermore, a smart approach is proposed to update incrementally and efficiently the dictionaries, making the application of our method to realistic tracking scenarios feasible. Furthermore, the proposed method collects training samples to update the two dictionaries during face tracking using a proposed scheme (see details in Section 4.2.6). The quality of the samples (i.e., reconstruction error) is assessed before utilizing them to update the dictionaries, which is an aspect that other methods that implement incremental learning seem to miss (ROSS et al., 2008). Both the dictionaries are initialized using Singular Value Decomposition (SVD), which is more efficient than initializing the process by combining some random training samples as proposed elsewhere (ELAD; AHARON, 2006). As both dictionaries are learned incrementally, the number of atoms can increase until a limit is reached. Additionally, the weights of the atoms are updated adaptively.

The proposed approach is illustrated in the block diagram shown in Fig. 4.6, which is explained below:

- Block 1: Initializes the face tracking process for the first frame, the initial target face, the affine parameters ($\chi(t)$), while the face landmarks are provided by a face landmark localization method (LUCEY et al., 2009). The tracked target face is assumed to be contained in a window of fixed size (u $\times$ u). This window and its contents are warped according to the affine parameters to obtain the candidate target face samples. These samples are used in a template matching process to detect the

tracked target face in the frame at time $t + \delta t$. The initial target face serves as the mean face ($\mu(t)$) until two dictionaries are created which are reconstruction and classification dictionaries. The two dictionaries are build and updated after a number (τ) of new tracked target face samples have been gathered during face tracking;

- Block 2: In the subsequent frames, a finite number (η) of affine parameters are drawn around the affine parameters of the initial/tracked target face using a Gaussian distribution (see Eq. 4.2);

- Block 3: To locate the tracked target face in the frame at time t, the candidate target face samples (u $\times$ u) are warped according to the computed affine parameters to be compared with the tracked target face (see the example in Fig. 4.6 at the left of Block 3: in red, the tracked target face from the previous frame; in green, candidate target face samples). See details in Section 4.1;

- Block 4: Next, a test is performed to check if the dictionaries already exist, or if enough tracked target face samples (τ) have been collected to create the two dictionaries;

- Block 5: When τ tracked target face samples are available, the samples are decomposed into fixed size patches (v $\times$ v, and v $\leq$ u), and the dictionaries are created using these patches;

- Block 6: Afterwards, the probability of each candidate target face sample being the tracked target face is computed using the learned dictionaries (see Eq. 4.18);

- Block 7: On the contrary, if the condition in Block 4 is not satisfied, the probability of the candidate target faces being the tracked target face is calculated using the distance from the mean face $\mu(t)$ (see Eq. 4.14);

- Blocks 8-9: Next, the candidate target face with the highest probability is selected to be the current tracked target face, and is used as training data to update the two dictionaries depending on its quality (i.e., reconstruction error);

- Blocks 10-11: The two dictionaries are created, or updated depending on the number of tracked target face samples accumulated (see details in Section 4.2.5 and 4.2.6). The training samples are gathered during the face tracking process based on their reconstruction errors (see Section 4.2.4);

- Block 12: Finally, if there are more frames of the video to process, the affine parameters of the current tracked target face are used in the next frames, and the process re-starts from Block 2.

As this operating mode of the tracker is based on the dictionary learning, the dictionary learning is detailed in Sections 4.2.2 and 4.2.3. The face appearance is represented based on the multi-model dictionary learning as explained in Sections 4.2.4. Finally, Section 4.2.7 details how to locate the tracked target face in consecutive frames by using a comprehensive appearance model based on multi-model dictionary learning.

4.2.2 Dictionary Learning

Given a set of data items $X = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^{e \times n}$, and its sparse code α on a dictionary $D \in \mathbb{R}^{e \times k}$ with k atoms/columns, there are two stages in the dictionary learning algorithm. The sparse coding stage, that is given for a fixed dictionary D using L_0 regularization as follows:

$$\arg\min_{\alpha} ||X - D\alpha||_2^2 + \lambda||\alpha||_0, \tag{4.3}$$

where λ is an adjustment parameter controlling the sparsity. Afterwards, the dictionary and its atoms are updated to represent the data sparsely. The dictionary atoms are obtained as follows:

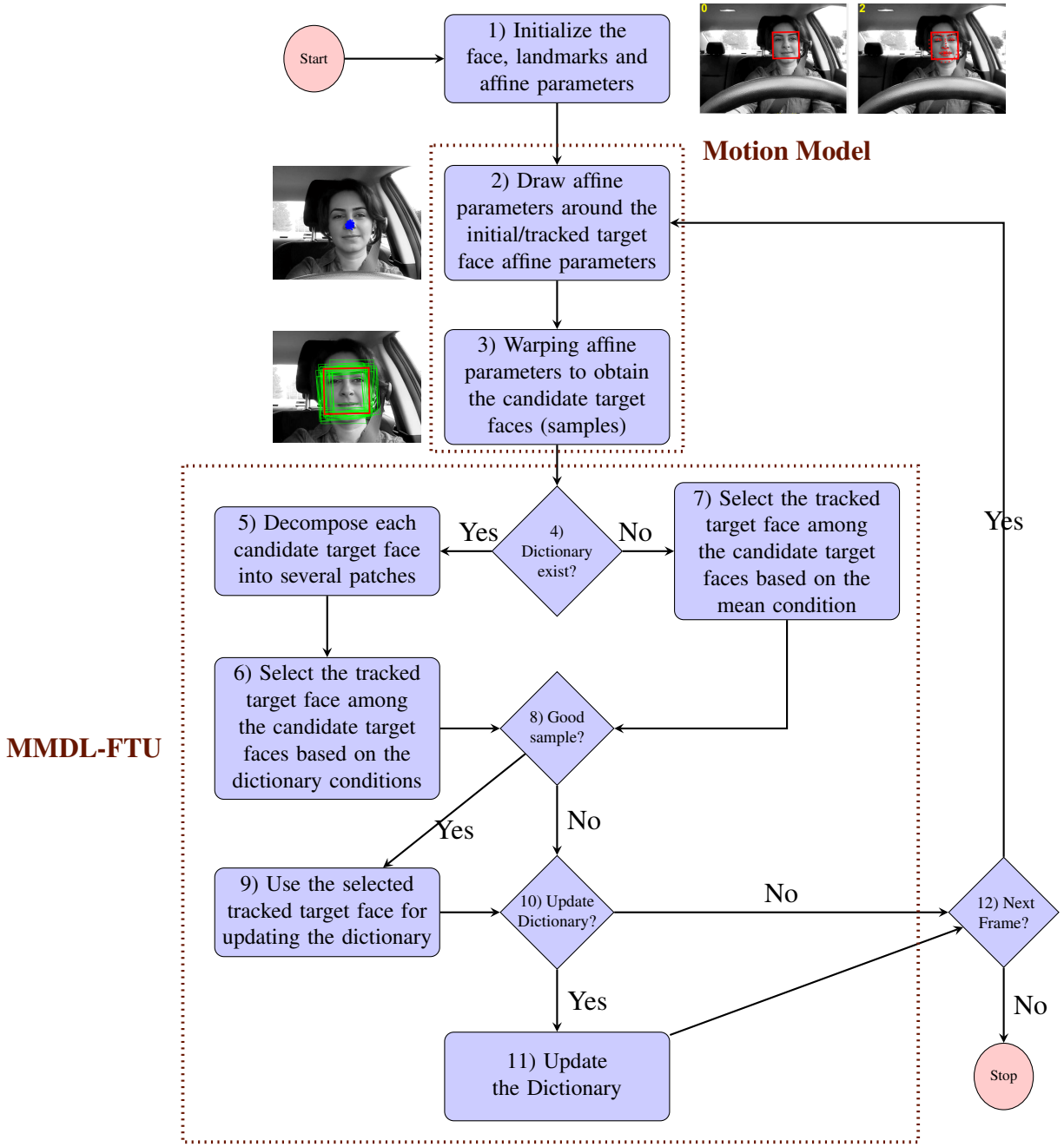
$$\arg\min_{D} ||X - D\alpha||_2^2. \tag{4.4}$$

The atoms updating stage in Eq. 4.4 is a characteristic that distinguishes different dictionary learning methods.

4.2.3 Incremental Dictionary Update

The discussion below applies to both the dictionaries. Consider a dataset $X_n = \{x_1, x_2, ..., x_n\}$ that can be represented using a dictionary D with a sparse code α_n. When new data $X_m = \{x_{n+1}, x_{n+2}, ..., x_{n+m}\}$ is added to X_n, the dictionary D is updated generating D'. As D already describes X_n sparsely in terms of α_n, it is useful to test if the dictionary D represents the new data sparsely as well, and in this case the dictionary update is not necessary. Otherwise, the dictionary D must be updated and the coefficients

Figure 4.6: Block diagram of the proposed MMDL based face tracking method.



α_{m+n} need to be calculated. Since the dictionary is applied in an object tracking application, it should be updated to adapt to the current appearance of the tracked target (e.g face). Furthermore, the dictionary is obtained using the K-SVD (MICHAL; MICHAEL; ALFRED, 2005), which tries to represent the data as sparsely as possible. For this reason in order to update the dictionary, the first step is to apply the incremental SVD (ROSS et al., 2008) to embed new data (for details on incremental update of SVD, see Section. 3.2.2). While updating the dictionary using the SVD, old atoms are down-weighted with a forgetting factor so the dictionary have an updated appearance of the tracked target.

To make the dictionary adaptive to new changes in the appearance of the tracked object (e.g. face), the objective function in Eq. 4.4 is modified as follows (WANG et al., 2014):

$$\arg\min_{\alpha} ||X_{n+m} - D^{'}\alpha_{n+m}||_2^2 + \lambda||\alpha_{n+m}||_0, \tag{4.5}$$

where the parameter λ is used for regularization. Eq. 4.5 is minimized until the maximum required sparsity is achieved.

4.2.4 Multi-Model Dictionary Learning

The proposed Multi-Model Dictionary Learning (MMDL) scheme, combines a reconstruction dictionary (D_p) that represent strictly the appearance of the target, and a classification dictionary (D_c) that discriminates the tracked target from the background. The dictionary D_c is primarily helpful in target tracking in changing background conditions. The dictionary update uses positive and negative samples obtained during target tracking to learn the dictionary incrementally, making it adaptable to contextual changes. The positive samples are the tracked target faces, and also scaled, rotated versions of these faces from previous frames. The negative samples correspond to the background and are taken from the previous frame with an overlap ratio ov (ov=0.05 in our experiments) with the tracked target face. Each positive/negative sample is decomposed into $v \times v$ sized patches, which are combined to make a patch matrix used to learn the dictionaries.

4.2.5 Reconstruction Dictionary

Firstly, the reconstruction dictionary (D_p) is learned using patches of only positive samples as described in Section 4.2.2. D_p estimates the appearance difference between the reconstructed sample $D_p\alpha_j$ and the candidate target face sample $\mathbf{I}_c$. Hence, the reconstruction error of the candidate target face $\mathbf{I}_c$ and the reconstructed sample $D_p\alpha_j$ using the dictionary D_p is given by:

$$\varepsilon_r = ||\mathbf{I}_c - D_p\alpha_j||_2^2, \tag{4.6}$$

where, $\mathbf{I}_c$ is the patch matrix of the candidate target face samples, and α_j are the D_p sparse coefficients.

4.2.6 Classification Dictionary

Secondly, the classification dictionary (D_c) is built by Unsupervised Information-Theoretic Dictionary Learning (UITDL) proposed by Flores et al (FLORES; SCHARCANSKI, 2016). However, the proposed MMDL method is based on K-SVD dictionary, while Non-Negative Matrix Factorization (NMF) is used in the case of UITDL. Given a dictionary $(D_c^{(0)})$ obtained by K-SVD (see Section 4.2.2), and the initial sparse representation α of the patch matrix Y (with positive and negative samples), and the maximum number of atoms to be selected, MMDL tries to learn a dictionary D_c by maximizing the following criteria:

$$f(.) = \sigma_1 MI(D_c; D_c^{(0)} - D_c) + \sigma_2 MI(Y; D_c), \tag{4.7}$$

where, the parameters $0 \leq \sigma_1 \leq 1$, $0 \leq \sigma_2 \leq 1$, and $\sigma_1 + \sigma_1 = 1$. These parameters σ_1 and σ_2 are used to balance the dictionary representation and compactness; $MI(A; B)$ represents the mutual information between two matrices A and B as below:

$$MI(A; B) = \sum_{A_{i,j}} \sum_{B_{i,j}} p(A_{i,j}, B_{i,j}) \times log \frac{p(A_{i,j}, B_{i,j})}{p(A_{i,j}), p(B_{i,j})}. \tag{4.8}$$

To find an atom d_i that maximizes Eq. 4.7 is equivalent to optimize (FLORES; SCHARCANSKI, 2016):

$$\underset{d_i \in D_c^{(0)} - D_c}{\arg\max} \{ \frac{[\sum]_{(i,i)} - \sigma_{D_i}^T \sum_D^{-1} \sigma_{D_i}}{[\sum_D]_{(i,i)} - \sigma_{\bar{D}_i}^T \sum_{\bar{D}}^{-1} \sigma_{\bar{D}_i}} \}, \tag{4.9}$$

where, $\bar{D}_c = D_c^0 - (D \cup d_i)$, $\sum_D$ is the covariance matrix of $D_c = [\sigma_{D_1 c}, \sigma_{D_2 c}, ..., \sigma_{D_n c}]$ and $\sum_{\bar{D}c}$ denotes the covariance matrix of $\bar{D}$. For classification, each candidate target face is represented in terms of the compact and representative dictionary D_c. The classification error is essentially a regression loss given by:

$$\varepsilon_c = ||H_i - W\alpha_i||^2, \tag{4.10}$$

where $H_i \in [0\ 1]$ is the label indicator, and $W \in R_{b \times k}$ is the linear classification parameters learned with a labeled dictionary computed using:

$$W = (\alpha \times \alpha^T)^{-1} \times \alpha \times H', \tag{4.11}$$

where H is the label vector of the training data represented in D_c by the sparse matrix α.

These two dictionaries D_p and D_c (reconstruction and classification) are combined in a single model, to create a multi-model dictionary which tend to improve the tracking robustness, as shown in Eq. 4.18. Both the dictionaries are updated separately using the technique mentioned in Section 4.2.3, after τ new tracked target face samples have been gathered. The collection of the samples is based on the proposed MMDL Face Tracker with Update test (MMDL-FTU), which collects only the samples with a reconstruction error smaller than ε (ε=0.05 in our experiments).

In face tracking, the mean face plays an important role, and must adapt to the current appearance of the tracked target face. Therefore, more weight is given to more recent observations, by employing a forgetting factor, and the mean face $\mu(t)$ at time t is updated as follows:

$$\mu(t) = \frac{f \cdot n \cdot \mu_n + m \cdot \mu_m}{m + f \cdot n}, \tag{4.12}$$

where μ is the updated mean, μ_n represents the mean of the older data (X_n), μ_m is the mean of the newly added observations (X_m) and $t = m + n$, whereas f is the forgetting factor. An important advantage of the forgetting factor is that the mean face can still change in response to the new observations, irrespective of the total number of observations.

4.2.7 Appearance Model and Tracked Target Face Selection using MMDL

The candidate target faces are sampled using the motion model explained in Section 4.1. The probability $p(\mathbf{I}(t) \mid \chi(t))$ of each warped candidate target face be the tracked target face is estimated using the probabilistic interpretation of MMDL (see Eq. 4.18). The probability of the candidate target face being the tracked target face is based on the joint probability of reconstruction and classification errors. For the reconstruction dictionary D_p, given a candidate target face $\mathbf{I}(t)$ predicted by $\chi(t)$, it is assumed that $\mathbf{I}(t)$ is represented in terms of the dictionary (D_p). The probability of a candidate target face being well represented by D_p is inversely proportional to the distances d_t and d_w of the candidate target face to the reference (mean face $\mu(t)$) represented by D_p. The term d_t is the distance of the candidate target face to D_p and d_w is the distance of the candidate target face to the reference face ($\mu(t)$) represented by D_p. The probability p_{dt} of the candidate

target face being the tracked target face based on the current dictionary D_p is:

$$p_d(\mathbf{I}(t)|\chi(t)) = \mathcal{N}(\mathbf{I}(t); \mu(t), D_p^{\dagger} + \epsilon I) = exp(-d_t), \tag{4.13}$$

which is the negative exponential value of d_t, where, $d_t = ||(\mathbf{I}(t) - \mu(t)) - D_p^{\dagger}(\mathbf{I}(t) - \mu(t))||^2$, $D_p^{\dagger} = (D_p^T D_p)^{-1} D_p^T$ is the pseudo-inverse matrix of D_p, and D_p is the reconstructive dictionary, which is constructed with positive samples only (FLORES; SCHARCANSKI, 2016). It is worth mentioning that when a dictionary is not available, $D_p^{\dagger}$ is set to 0 in Eq. 4.13, and only the distance from the mean face $\mu(t)$ is used to find the tracked target face as:

$$p_d(\mathbf{I}(t)|\chi(t)) = exp(-||(\mathbf{I}(t) - \mu(t))||^2). \tag{4.14}$$

Furthermore, the probability $p_{wt}(\mathbf{I}(t)|\chi(t))$ of the candidate target face being the tracked target face represented by the dictionary D_p can be modeled by the Mahalanobis distance from $\mu(t)$ represented by D_p:

$$\begin{aligned} p_w(\mathbf{I}(t)|\chi(t)) &= \mathcal{N}(\mathbf{I}(t); \mu(t), D_p \Sigma^{-2} D_p^T) \\ &= exp(-d_w), \end{aligned} \tag{4.15}$$

where, $d_w = -||(\mathbf{I}(t) - \mu(t))^T D_p \Sigma^{-2} D_p^T (\mathbf{I}(t) - \mu(t))||^2$, Σ is a diagonal matrix containing the coefficients of the dictionary D_p atoms. The probability of the candidate target face being the tracked target face is given by (similar to $\mu(t)$ represented by D_p):

$$p_r(\mathbf{I}(t)|\chi(t)) = p_d(\mathbf{I}(t)|\chi(t)) p_w(\mathbf{I}(t)|\chi(t)) \tag{4.16}$$

Furthermore, the likelihood of the candidate target face being well represented in terms of the classification dictionary D_c is given by the expected value of the negative classification error, in other words, the candidate target face with the small classification error receives higher value, and vice versa:

$$p_c(\mathbf{I}(t)|\chi(t)) = exp(-\varepsilon_c). \tag{4.17}$$

where, $\varepsilon_c = ||Y_i - W\alpha_i||^2$ is given by Eq. 4.10. To obtain the combined probability of the candidate target face to be the tracked target face, the reconstruction and classification

probabilities are combined as follows :

$$p(\mathbf{I}(t)|X(t)) = \Omega p_r(\mathbf{I}(t)|\chi(t)) + (1-\Omega)p_c(\mathbf{I}(t)|\chi(t). \tag{4.18}$$

where, Ω is a weight associated to the classification and reconstruction dictionaries, and indicates the trade-off between reconstruction and classification dictionaries probability of the candidate to be the tracked target face.

4.2.8 Facial Landmarks Localization

The candidate face sample that has higher combined probability is selected to be the tracked target face, and the associated affine parameters $\chi(t)$ are used to estimate landmarks on the tracked target face:

$$\Lambda_T(t) = \chi(t) \times [\Lambda(1); \vec{1}], \tag{4.19}$$

where, $\Lambda(1)$ are the landmark locations in the initial target face and $\vec{1}$ is an unitary vector of length Z (total number of landmarks). This tracked target face is used to update the two dictionaries depending on the reconstruction error obtained with Eq. 4.6.

4.2.9 Pseudo Code

The pseudo code of the proposed procedure is shown in Algorithm 1. The algorithm receives the current frame $I(t)$, the landmark locations $\Lambda_T(t-1)$ and the affine parameters $\chi(t-1)$ of the tracked target face in the previous frame, the current dictionaries D_p and D_n, the sparse code α_p and α_c, the mean μ_o, a $flag$ variable that counts number of face samples to update the dictionary, and a variable Υ that checks if there are more frames available to process. The algorithm returns the facial landmarks in the current frame $\Lambda_T(t)$, the affine parameters $\chi(t)$ in the current frame, the updated dictionary D, the updated mean $\mu(t)$, the tracked target frame in the current frame at time t and the sparse code α.

The face tracking process is initialized in the first frame with the initial target face, the affine parameters ($\chi(t)$), while the face landmarks are provided by a face landmark localization method (LUCEY et al., 2009). The tracked target face is assumed to

Algorithm 1 Dictionary Update and Face Tracking.

procedure DUT($I(t)$, $\Lambda_T(t-1)$, $\chi(t-1)$, D_p,D_c, α_p,α_c μ_o, $flag$, Υ)

▷ $I(t)$ is current frame, $\Lambda_T(t-1)$, $\chi(t-1)$ are landmarks and affine parameters of previous frame respectively, Υ is 1 if there is at least one more frame to process, otherwise Υ is 0, α_p is sparse code of data over D_p and α_c is the sparse code over D_c, μ_o is older mean.

while ($\Upsilon \equiv 1$) **do**
  Draw affine parameters against $\chi(t-1)$ using Eq.4.2.
  Warp candidate target face samples from $I(t)$ using these affine parameters.
  Find the probability of each candidate target face sample being the target face using Eq. 4.18.
  Select the tracked candidate target face ($\mathbf{I}(t)$) using Eq. 4.18.
  Estimate the landmarks of the tracked target face using Eq. 4.19.
  calculate reconstruction error using Eq. 4.6.
  if ($\varepsilon_r \leq 0.05$) **then**
    $flag \leftarrow flag + 1$. ▷ Use this tracked target face sample for training.
  if ($flag \geq \tau$) **then**
    $flag \leftarrow 0$.
    Update the dictionaries D_p and D_c using Eq. 4.5.
    Update the mean $\mu(t)$ using Eq. 4.12.
return $\Lambda_T(t)$, $\chi(t)$,D, $\mu(t)$, $\mathbf{I}(t)$, α.

be contained in a window of fixed size (u $\times$ u). If there are frames to process, the affine parameters are drawn around the affine parameters in the previous frame using a Gaussian distribution. These affine parameters computed previously are used to warp the candidate target face samples that may contain the tracked target face in the current frame. Next, the probabilities of these candidate target face samples to be the tracked target face are calculated. The candidate target face with the highest probability to be the tracked target face is selected. The affine parameters associated with this candidate target face are utilized to estimate the facial landmarks in the current frame.

In the next stage, the reconstruction error of the tracked target face is computed, and a quality test is performed before utilizing this target face to update the dictionaries. Furthermore, the dictionaries are updated after a specific number of target face samples have been accumulated. Also, the mean $\mu(t)$ is updated. Finally, if there are more frames to process, the affine parameters of the current tracked target face are used in the next frames.

4.3 Proposed Methodology for an Adaptive Face Tracker with Resyncing Mechanism (AFTRM) and AFTRM Weighted (AFTRM-W)

This section provide details of the AFTRM-W operating mode of the proposed face tracking method. This operating mode evaluates the quality of the face tracking process using an error prediction scheme and the tracking process is corrected using a resyncing scheme. This operating mode has two variations, which are based on the resyncing mechanism it uses. The AFTRM variant of this operating mode uses Constrained Local Models (CLM) as a resyncing mechanism, whereas, AFTRM-W utilizes Weighted CLM (W-CLM) to resync important features.

4.3.1 AFTRM-W Operating Mode

Visual tracking is relevant in various areas such as car tracking (LIU; LI; FANG, 2015), face detection (CHRYSOS et al., 2014), and drivers monitoring (OMIDYEGANEH et al., 2016). Object tracking may become quite challenging when there are changes in appearance or shape of the tracked target object, or when the tracking conditions change (e.g., scene illumination changes). Therefore, most methods available in the literature tend to perform well over short time spans and under controlled conditions. However, often when the object tracking method misses the tracked target object, the tracking error tends to increase which results in missing the tracked target indefinitely. Often, visual object tracking methods do not take into account visual appearance changes (TERISSI; GÓMEZ, 2007), and for this reason, they may interrupt tracking the target object after a period of time when the tracking conditions change (e.g., the scene illumination changes).

Some authors proposed approaches that use the data generated during the tracking process to accommodate possible appearance changes of the tracked target object, such as online learning (BABENKO; YANG; BELONGIE, 2009), incremental PCA (ROSS et al., 2008), and online feature learning techniques combined with dictionaries (LIU; LI; FANG, 2015). Often online visual tracking methods tend to miss the tracked target object in complex scenarios, such as when the head pose changes while tracking faces, or in cluttered backgrounds and/or in object occlusions (COOTES et al., 2001). The reasons for this behavior include the inability to access the tracking error and to update the object appearance at runtime. Danelljan et al (DANELLJAN et al., 2015) proposed Spatially Regularized Discriminative Correlation Filters (SRDCF) to track objects visually, im-

proving the Discriminative Correlation Filters (DCF). The SRDCF use negative samples in a way to not corrupt positive samples while improving the SRDCF tracking performance. The SRDCF method penalizes off center patches to allow filters to concentrate on the center of the training patches, this penalization and switching between spatial and Fourier domains makes SRDCF too complex for online tracking. More recently, Sanchez et al. (SÁNCHEZ-LOZANO et al., 2016) proposed an Incremental Cascaded Continuous Regression (iCCR) method to track faces. The iCCR method is a new formulation for Cascaded Continuous Regression (CCR), which updates the model at each frame and can be utilized in incremental learning.

On the other hand, geometric shape and appearance models such as Active Appearance Models (AAM) (COOTES et al., 2001), Active Shape Models (ASM) (COOTES et al., 1995) and Constrained Local Models (CLM) (LUCEY et al., 2009) can capture robust features even in cluttered or fast-changing scenarios, and improve the tracking process. These methods often are based on local shape matching, which requires optimization to minimize the difference between a tracked target object and the learned target model appearance (i.e., to maximize the match). Unfortunately, most shape and appearance model based methods are not easily applicable to real-time tracking due to their complexity. Nevertheless, combining online learning with shape and appearance models can increase the online learning efficiency. Particularly, if the shape and appearance model is utilized to correct the tracking process so that it reduces the tracking failures.

The proposed approach improves on a well-known object tracking method based on the incremental PCA (ROSS et al., 2008). The proposed scheme learns from the data generated during object tracking and corrects the tracking mistakes with a resyncing mechanism. Also, a dynamic tracking error predictor is proposed to estimate how accurately the target object is being tracked. Furthermore, the tracking error predictor adapts itself in time and tends to be consistent in long video sequences (see Section. 4.3.6). Consequently, if the estimated tracking error is increasing, the tracking process is corrected by a resyncing mechanism based on CLM. In addition, it is also proposed an improvement of CLM named Weighted CLM (W-CLM) that utilizes the training data to assign a weight to each landmark (feature point) based on its consistency in time (see Section. 4.3.3). One of the possible applications of the proposed tracking method is the face and facial landmarks tracking, where CLM or W-CLM can be used to re-adjust the facial features locations (landmarks) when there is a potential tracking failure.

The proposed Adaptive Face Tracker with Resyncing Mechanism (AFTRM) op-

timizes the CLM search process without using the landmarks weights. Whereas, the also proposed method named Adaptive Face Tracker with Resyncing Mechanism with Weights (AFTRM-W) apply a landmark weight (calculated during the W-CLM training phase) to improve the landmark search process. Face tracking based on facial features can provide a cost-effective solution to a number of measurement applications, such as yawning detection, expression analysis, fatigue detection, and vigilance (SHIRMOHAMMADI; FERRERO, 2014). In this work, the tracked facial landmarks are evaluated in face tracking and an application (i.e., yawning detection) in a driving scenario.

Fig. 4.7 shows the block diagram of the proposed object tracking method applied to face tracking, and the blocks functions are explained below :

- Block 1: In the first frame, the initial target face, its affine parameters and the landmarks are localized using W-CLM (for details on W-CLM, see Section 4.3.3).
- Block 2: In order to track the target face in the subsequent frames, a finite number of affine parameters are drawn around the affine parameters of the initial/tracked target face in the previous frames (see details in Section 4.3.5).
- Block 3: Next, the affine parameters previously computed are used to warp the current frame candidate target face samples of size $u \times u$.
- Block 4: A test is performed to check if the eigenbases exist. The bases are build after a specific number (τ) of new target face samples have been gathered.
- Block 5: If the condition in block 4 is satisfied, the candidate target face samples are decomposed into patches ($v \times v$ and $v \leq u$), as the bases are build using patches (see Section 4.3.2).
- Block 6: The tracked target face is found among the candidate target face samples by maximizing a likelihood fuction (see details in Section 4.3.5 and Eq. 4.28).
- Block 7: If the condition in block 4 is not satisfied, the tracked target face is estimated by the mean condition (see Eq. 4.26).
- Block 8: The proposed error predictor checks if resyncing of the features is required to correct the tracking process (see details in Section 4.3.6).
- Blocks 9-10: If resyncing of the features is not necessary, the bases are updated if sufficient new tracked target face samples τ have been accumulated to build or update the bases (see details in Section 4.3.2).

- Blocks 12-13: In case the tracking error is higher than a certain threshold, the W-CLM is used to re-locate the facial landmarks and correct the tracking process (see details in Section 4.3.3).

- Block 15: The tracked target face and its affine parameters are used as seeds to continue tracking in the next frame if there are more frames to process.

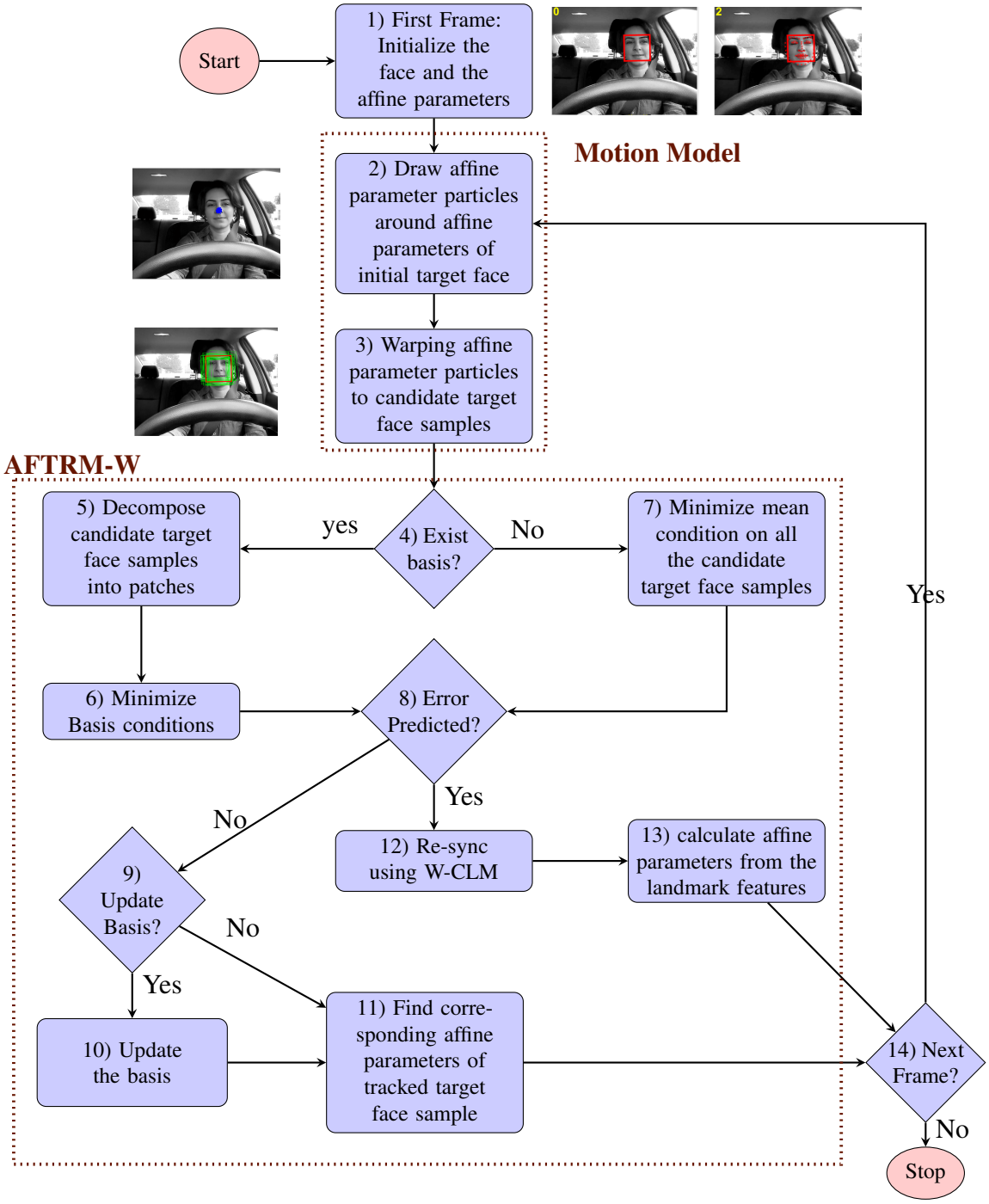
The proposed tracking algorithm can track non-rigid objects and detect early potential deviations from the target object. The incremental update of the tracking method is inspired by the incremental PCA approach (ROSS et al., 2008), however, the proposed method uses local texture information (patches of size $v \times v$) to build the basis rather than global texture (the tracked target object as a whole). A brief explanation of the efficient eigenbasis update used in this work is presented in section 4.3.2, followed by an explanation of the weighted CLM scheme and how it is used as a resyncing mechanism in section 4.3.3. The proposed tracking method is applied to the face and facial landmark tracking which is described in section 4.3.5. The account of the tracking error predictor and when to re-sync is given in section 4.3.6. The error predictor is ultimately used to correct the target tracking if the error is beyond a specific threshold. This threshold is selected dynamically, so it adjusts along the time to make the error predictor more consistent.

4.3.2 Incremental Update of Eigenbasis and Mean

The appearance of a non-rigid target object may change substantially along time due to intrinsic and extrinsic factors. For this reason, a tracking method should update the appearance model being used to accommodate local and global changes occurring over time, making the tracking process more robust. Ross et al. (ROSS et al., 2008) improved the Sequential Karhunen-Loeve (SKL) algorithm proposed by Levey et al. (LEVEY; LINDENBAUM, 2000), and applied this modified scheme to update the eigenbasis used in PCA. Each sample is represented by a column vector in order to construct and update PCA, that describes global texture information. However, in order to accommodate local changes, the proposed scheme construct and update basis using local patches of size $v \times v$ obtained from target object samples of size $u \times u$ where $u > v$. The SVD of the two set of observations A and B is represented by:

$$\begin{bmatrix} A & B \end{bmatrix} = U' \Sigma' V'^{\mathrm{T}}, \tag{4.20}$$

Figure 4.7: Block diagram of the proposed AFTRM-W face tracking method



and this can be calculated by using the already computed SVD of $A = U\Sigma V^T$ and the new set of observations B as:

$$\begin{bmatrix} A & B \end{bmatrix} = \left(\begin{bmatrix} U & \widetilde{B} \end{bmatrix} \widetilde{U} \right) \widetilde{\Sigma} \left(\widetilde{V}^T \begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix} \right), \tag{4.21}$$

where I is an identity matrix, $U^{'}$ = [U $\widetilde{B}$] $\widetilde{U}$ and $\Sigma^{'}$ = $\widetilde{\Sigma}$. Also, $\widetilde{B}$ is the component of B orthogonal to U, and $R = \begin{bmatrix} C & U^T B \\ 0 & \widetilde{B}^T B \end{bmatrix}$ is a square matrix of size $q + m$, and the SVD of R is given by $\widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$. The details on the incremental update of the PCA are given in Section 3.2.2.

Furthermore, the contribution of the old data is decreased by embedding a forgetting factor f. The application of forgetting factor makes the eigenbasis up to date to the current appearance of the tracked target object (face). Similarly, due to the utilization of the mean $\mu(t)$ in the detection of the tracked target face, the mean is also updated with a forgetting factor f for the same reason as the eigenbases. For details on the mean update, the reader is referred to Section 3.2.3. The equation of the mean $\mu(t)$ at time t is calculated as shown below:

$$\mu(t) = \frac{f \cdot n \cdot \mu_n + m \cdot \mu_m}{m + f \cdot n}, \tag{4.22}$$

where μ_n is the mean of the data matrix A and μ_m is the mean of the newly received data B, $t = m + n$, and f represents the forgetting factor. A benefit of having a forgetting factor is that the mean $\mu(t)$ at time t can still change in response to new observation, even if the total number of observations is very large.

4.3.3 Weighted Constrained Local Models (W-CLM) as Re-syncing Feature Detectors

Constrained Local Models (CLM) tend to be accurate facial feature detectors, on the other hand, they are known to have a slow convergence process, making their use in the face tracking quite challenging. Nevertheless, if CLM is used less often in comparison to other components of the tracking process, the tracking system could be viable for real-time operation. In this work, the proposed tracking scheme has been developed in the context of face tracking, and W-CLM is discussed within this context in the following two ways. Firstly, CLM/W-CLM is used to locate the face and the facial landmarks locations in the first frame of a video sequence. Secondly, CLM/W-CLM is used to resync the face model when the proposed tracking failure predictor suggests significant tracking inaccuracies. Consequently, the proposed method is self-driven and self-corrected in real-time.

The CLM utilizes the landmarks in the training data to build the CLM model, which makes it dependent on the landmark locations (to build the shape model) and the texture information around the landmark (to build the patch model). Due to the occlusion and/or invisibility of some landmarks in some training data, the CLM model can be affected. This is a very common scenario in non-rigid objects such as human faces (BEHAINE; SCHARCANSKI, 2012). The proposed method handles this problem by assigning weights to each landmark based on its consistency.

The proposed method called W-CLM utilizes CLM training data to evaluate the landmarks consistency by assigning higher weights to more consistent landmarks during the CLM search process. Multivariate Mutual Information (MMI) measures the mutual dependence between two or more random variables (CRUYS, 2011), and is utilized here to evaluate the consistency of each facial landmark. Firstly, MMI is computed independently for the feature vector of each facial landmark observed in a temporal window. Each feature vector contains the texture information in a window of size $\sqrt{l} \times \sqrt{l}$ around a facial landmark location in a given video frame. MMI evaluates the differences of the co-occurrence probabilities of n random variables describing the local texture, and can be used to indicate how consistent is the texture information around any particular landmark in the training images, which is used as a weight $\hat{w}_i \in [0, 1]$ of a facial landmark:

$$\hat{w}_i(x_1, x_2, , .., x_N) = log\frac{p(x_1, x_2, .., x_N)}{\prod_{i=1}^{n} p(x_i)}, \tag{4.23}$$

where, x_i is a column vector of size l, containing texture information around a particular landmark $i = 1, 2, ..., Z$ in a video frame at time t. The weights $\hat{w}_i$ of the landmarks are combined in a diagonal matrix to be used in the W-CLM search process.

In practice, CLM consists of two stages (modules), which are CLM model building and CLM search (LUCEY et al., 2009), which has been explained in Section 3.5. The proposed method builds the CLM model as it is explained in Section 3.5.1, and the weighted CLM search is discussed next.

4.3.4 CLM Weighted Search

The W-CLM search process combines the shape and patch models similarly to the CLM search explained in Section 3.5.2 to detect the facial landmarks of a face. Given a set of initial facial landmarks, the cropped patch around the current position of each landmark

is processed by the SVM based patch model, while preserving the shape constraints. Both these goals are combined using the following objective function with the corresponding weights of the landmarks and the Eq. 3.31 can be re-written as:

$$f(S_t) = \sum_{i=1}^{Z} \hat{w}_i \imath_i(x_i, y_i) - \beta \sum_{j=1}^{o} \frac{-h_j^2}{\lambda_j}, \tag{4.24}$$

where $\imath_i(x_i, y_i)$ is the SVM response of c^{th} feature template (i.e., cropped patch from the current position of landmark point) at the position (x_i, y_i) of landmark i and $\hat{w}_c$ represents the weight of the landmark i. The weight $\hat{w}_i$ describes how much effect this particular landmark will have in the fitting process. The term $\sum_{j=1}^{o} \frac{-h_j^2}{\lambda_j}$ is the shape constraint, where $h_j \in H$ is the corresponding eigenvector coefficient in the eigenvectors representation H, λ_j is its eigenvalue and o is the number of eigenvectors, a the and arameter $\beta \in [0\ 1]$ is a bias determining the compromise between shape fit and the SVM based patch model. If the term β is set to a high value, the shape difference is penalized heavily, which essentially tries to find a shape that is close to mean shape, which possibly sacrifices (or even, completely ignoring) the SVM shape model response.

On the other side, a very low value of β tries to find the landmark locations with high response, which possibly ignores the shape constraint. For this reason, it is sensible to compute the value of β from the training data. The W-CLM search process starts with initializing the facial landmarks, which can be the mean shape. For each facial landmark, a patch is cropped from its current position, and the patch model is used to find the output at each landmark. The term $\sum_{i=1}^{Z} \hat{w}_i \imath_i(x_i, y_i)$ in Eq. 4.24 is optimized using quadratic programming and can be readily solved using the Matlab *quadprog* function.

The best facial landmarks positions are found by combining these quadratic functions with their corresponding weights $\hat{w}_i$ similar to Eq. 3.31 while maintaining the shape constraint $\beta \sum_{j=1}^{s} \frac{-h_j^2}{\lambda_j}$ from CLM shape model or until the maximum number of iterations have been reached. An important key factor in the CLM search is the initialization. A good initialization of landmarks can lead to quick optimization. In case of a new sequence initialization is made on the mean shape of the shape model, however, if the previous frame's facial landmarks are available, utilizing them can improve the time of the optimization process.

4.3.5 Appearance Model and Tracked Target Face Selection

The candidate target faces are sampled using the motion model explained in Section 4.1. Now given the state of every candidate target face sample, the quality of the sample can be evaluated using the appearance model. The proposed methodology models the appearance of the tracked target face using a probabilistic PCA. A candidate target object (i.e., face) sample $\mathbf{I}(t)$ that is predicted by $\chi(t)$ is assumed to be generated from the subspace of the target object spanned by the eigenbases U and centered at the mean $\mu(t)$. The probability p of a candidate target object (i.e., face) sample being generated from this subspace is inversely proportional to the distance δ of the sample from the reference point (i.e., mean ($\mu(t)$)) of the subspace. This distance is comprised of the distance to the subspace (δt) and within the subspace distance (δw) of the projected sample to the subspace center ($\mu(t)$). The probability ($p_{\delta t}$) of a candidate target object sample being the tracked target object is given by the negative exponential value of the distance (δt):

$$\begin{aligned} p_{\delta t}(\mathbf{I}(t)|\chi(t)) &= \mathcal{N}(\mathbf{I}(t); UU^T + \epsilon \boldsymbol{I}) \\ &= exp(-\delta t), \end{aligned} \tag{4.25}$$

where, $\delta t = ||(\mathbf{I}(t) - \mu) - UU^T(\mathbf{I}(t) - \mu)||^2$, $\epsilon \boldsymbol{I}$ is the additive noise in the observation process with $\boldsymbol{I}$ being an identity matrix and $\epsilon \to 0$. It is worth-mentioning that if the basis are not available, the term U is set to 0, which changes the equation to:

$$p_{\delta w}(\mathbf{I}(t)|\chi(t)) = exp(-||(\mathbf{I}(t) - \mu)||), \tag{4.26}$$

is described as mean condition. This equation computes the distance of the candidate target object with the mean $\mu(t)$, calculated using Eq. 4.22, at time t. Now within the subspace, the probability ($p_{\delta w}$) of the projected candidate target object sample being the tracked target object can be given in terms of the negative exponential value of Mahalanobis distance δw of the candidate target object (face) sample from the projected mean ($\mu(t)$) at time t as follows:

$$\begin{aligned} p_{\delta w}(\mathbf{I}(t)|\chi(t)) &= \mathcal{N}(\mathbf{I}(t); \mu, U\Sigma^{-2}U^T) \\ & \quad exp(-\delta w), \end{aligned} \tag{4.27}$$

where, $\delta w = ||(\mathbf{I}(t) - \mu(t))^T U \Sigma^{-2} U^T (\mathbf{I}(t) - \mu(t))||^2$, $\sum$ is the singular value matrix. Finally, the likelihood of a candidate target object sample being the tracked target object sample is given by the combined probability of within space distance $p_{\delta w}$ and its distance from the subspace $p_{\delta t}$:

$$\begin{aligned} p(\mathbf{I}(t)|\chi(t)) &= p_{\delta t}(\mathbf{I}(t)|\chi(t)) p_{\delta w}(\mathbf{I}(t)|\chi(t)) \\ &= \mathcal{N}(\mathbf{I}(t); UU^T + \epsilon I)\mathcal{N}(\mathbf{I}(t); \mu, U\Sigma^{-2}U^T). \end{aligned} \tag{4.28}$$

The candidate target object sample that has the highest probability of being the tracked target object is selected. As the proposed method is applied to face tracking, the affine parameters $\chi(t)$ that are associated with this candidate target object sample are used to estimate the facial landmarks as shown below:

$$\Lambda_T(t) = \chi(t) \times [\Lambda(1); \vec{1}] \tag{4.29}$$

where $\Lambda(1)$ are the landmark locations in the initial target face and $\vec{1}$ is a unitary vector of length Z (total number of landmarks). In the start, eigenbasis U are initialized to be empty, and the first face serves as mean (μ) until the eigenbases are created, as eigenbases are built after specific number τ of tracked target face samples have been accumulated. Then, a finite number of affine parameter particles are drawn according to the motion and sampling model mentioned in Section 4.1. These affine parameter particles are used to warp the candidate target object samples that may contain face in the video frame at time t. For each candidate target face sample, the likelihood of being the tracked target face is calculated using the observation model explained in Section 4.1. The candidate sample with the highest probability of being the tracked target object is selected, and its associated affine parameters are used to estimate the facial landmark features. When the specific number of tracked target face samples (τ) has been accumulated which are counted using the $flag$ variable, incremental update is performed on the eigenbasis and the mean $\mu(t)$, with a forgetting factor. In the proposed method the update is performed each fifth frame $(\tau = 5)$.

The pseudo code of the procedure is given in Algorithm 2. The algorithm receives the following parameters as input: the current frame $I(t)$, the facial landmarks $\Lambda_T(t-1)$ and $\chi(t-1)$ in the previous frame, the eigenbases U and the corresponding eigenvalues Σ, the mean μ_n, the tracked target face from the previous frame $\mathbf{I}(t-1)$, a static variable $flag$ that counts the number of tracked target face samples gathered to update the bases

and a variable Υ that checks if there are more frames to process. Further, at each frame the tracking algorithm returns the facial landmarks $\Lambda_T(t)$ in the frame, the affine parameters $\chi(t)$ related to the tracked target face, the updated eigenvalues $\Sigma^{'}$ and eigenvectors $U^{'}$, the updated mean $\mu(t)$ and the tracked target face $\mathbf{I}(t)$ in the current frame at time t.

In order to track the target face in the subsequent frames, a finite number of affine parameters are drawn around the affine parameters of the initial/tracked target face in the previous frames. These affine parameters are used to warp the current frame candidate face samples. The probability of each candidate target face sample to be the tracked target face sample is computed, and the one with the highest probability is selected to be the tracked target face. The affine parameters associated with the tracked target face is used to estimate the facial landmarks in the current frame. If enough tracked target face samples have been accumulated, the eigenbases and the mean are updated.

Algorithm 2 Incremental Learning for Face Tracking Algorithm (ILFT).

procedure ILFT($I(t), \Lambda_T(t-1), \chi(t-1), \Sigma, U, \mu_o, \mathbf{I}(t-1), flag, \Upsilon$)
▷ $I(t)$ is the current frame, $\mathbf{I}(t-1)$, $\Lambda_T(t-1)$,$\chi(t-1)$, are the tracked target face sample, facial landmarks and the affine parameters of the previous frame and Υ is 1, if there is at least one more frame to process, otherwise Υ is 0.
    while ($\Upsilon = 1$) **do**
        $flag \leftarrow flag + 1$
        Draw a finite number of affine parameter particles around $\chi(t-1)$ using Eq. 4.2.
        Warp the candidate target face samples from $I(t)$ using these affine parameter particles.
        Compute the probability of each candidate target face sample being the tracked target face sample using Eq. 4.28.
        Select the candidate target face sample with highest probability as the tracked target face sample $\mathbf{I}(t)$.
        Estimate the facial landmarks using 4.29.
        if ($flag \geq \tau$) **then**
            $flag \leftarrow 0$
            Calculate $\Sigma^{'}$ and $U^{'}$ in Eq. 4.20.
            Update the mean ($\mu(t)$) using Eq. 4.22.
    return $\Lambda_T(t)$, $\chi(t)$,$\Sigma^{'}$, $U^{'}$,μ, $\mathbf{I}(t)$.

4.3.6 Tracking Error Predictor and Resyncing Mechanism

Visual tracking is prone to failure if the object changes, does a quick motion or changes appearance, and so on. Therefore, often tracking methods fail, and the tracking

error keeps on increasing, and the tracking fails indefinitely. Most of these methods fail to provide a self assessment of tracking (TERISSI; GÓMEZ, 2007; BABENKO; YANG; BELONGIE, 2009; ROSS et al., 2008; VIOLA; JONES, 2001; YUAN et al., 2013). The proposed method is based on an error predictor which tries to estimate the tracking error at runtime. Measures like the change in the eigenbases and differences in the tracking landmarks on the face in consecutive frames are tested. It was found in the experiments that a relevant measure to predict the tracking error is the difference in the tracking landmarks (feature points) represented by the Track Points Difference ($\Delta(t)$) at time t. This is verified using the correlation (ρ) with the tracking error (ε), and $\Delta(t)$ at time t is given by:

$$\Delta(t) = \frac{1}{Z}\sum_{i=1}^{Z}||\Lambda_T^{(i)}(t) - \Lambda_T^{(i)}(t-1)||^2, \tag{4.30}$$

where $\Lambda_T^{(i)}(t)$ is the location (x, y) of the landmark i at time t estimated by the proposed method. To further improve the tracking error prediction, the median and moving median filters can be applied to the $\Delta(t)$ noisy estimates (see details in Chapter. 5).

The next stage of the process is to predict the tracking error itself. This is done by checking if the value of $\Delta(t)$ in Eq. 4.30 is higher than a certain threshold. A constant threshold value is not suitable for real applications because $\Delta(t)$ may vary from one person to another because of face size, closeness to the camera and/or a number of landmarks used. This requires a dynamic threshold which can auto-adjust to different environments. For this reason, the median value ($\Gamma_T = Median(\Delta(T))$) is used as the threshold instead:

$$\Psi(t) = \begin{cases} 1, & \text{if } \Delta(t) \geq \Gamma_T, \\ 0, & \text{otherwise,} \end{cases} \tag{4.31}$$

where $\Delta(T) = \{\Delta(1), ..., \Delta(t)\}$, and $\Psi(t)$ is used to indicate if resyncing is required. It can be seen in the Chapter 5, that the proposed error predictor is highly correlated with the tracking error. Some landmarks on the tracked object can serve for checking the tracking error, while tracking continues. When the tracking predictor indicates a higher tracking error, the resyncing of the features using W-CLM is called to correct the tracking process by re-adjusting the tracked landmarks.

The pseudo code of the proposed tracking algorithm with resyncing mechanism is provided in Algorithm 3. For the first frame, the face and facial features are initialized

Algorithm 3 Adaptive Face Tracker with Resyncing Mechanism (AFTRM).

procedure AFTRM($I(t), \Lambda_T(t-1)$), $\mathbf{I}(t-1)$ ▷ $I(t)$ is the current frame, $\mathbf{I}(t-1)$ is the tracked target face sample and $\Lambda_T(t-1)$ represents the facial landmarks of the previous frame.
2: $flag \leftarrow 1$
while $\Upsilon = 1$ **do**
4: Track $\mathbf{I}(t)$ and estimate $\Lambda_T(t)$ using Algorithm 2.
Calculate $\Delta(t)$ using Eq. 4.30.
6: Calculate Ψ using Eq. 4.31.
if $\Psi = 1$ **then**
8: Update $\mathbf{I}(t)$ and $\Lambda_T(t)$ using W-CLM ▷ see Section. 4.3.3.
Re-Initialize subspace U and mean $\mu(t)$.
10: Calculate affine $\chi(t)$ parameters of $\Lambda_T(t)$.
return $\Lambda_T(t)$, $\chi(t)$, $\mathbf{I}(t)$.

using the W-CLM. From the second frame, it calls Algorithm 2 to track the face and the facial features until it fails. When the prediction of tracking failure is made, W-CLM is used to re-sync, which re-locates important features of the moving target (e.g., facial landmark features). These features are then used for further processing such as finding new affine parameters $\chi(t)$ and the tracked target face bounding box $\mathbf{I}(t)$ in the current frame. Moreover, a new subspace is created starting from this frame, and the old data is discarded because it is least relevant and can be removed safely.

5 EXPERIMENTAL RESULTS

This chapter presents the experimental results of the proposed tracking methods. The proposed methods are evaluated on the qualitative and the quantitative basis in comparison with the state of the art tracking methods. Due to the different parameter settings of the proposed tracking methods, and the qualitative outputs, the qualitative experimental results of the MMDL-FT and MMDL-FTU are presented in the Section 5.2.1, and AFTRM and AFTRM-W are presented in Section 5.2.2. Next, the quantitative results are presented and discussed in Section 5.3. Next, the yawning detection is used as a case study for the evaluation of the proposed methods in real facial analysis problem, discussed in the Section 5.4.

5.1 Experimental Settings

The proposed tracking algorithms were implemented in Matlab 2015a on an IBM PC compatible with 3.40GHz i7-6700 CPU with 16GB internal memory. For experimental evaluation, the YawDD dataset (ABTAHI et al., 2014), is used which is freely available for educational purposes. The dataset contains videos of drivers performing various facial expressions, which includes neutral, talking/singing, and yawning. The database is created to be used to design and test the algorithms for yawning detection. The total of 119 participants from different age groups with the minimum age of sixteen years are involved. The videos from 29 participants are recorded using camera installed on the dash, and for other 90 participants, the camera is installed under the front mirror. These participants posses various skin, hair, and eye color, and belonging to Caucasian, African, Middle-eastern, and Asian ethnicities. The facial characteristics include prescription glasses, sunglasses, women using a scarf or not using a scarf, men with mustache or beard on the face and men without having a mustache or beard. The camera is installed on the dash or under the front mirror of the car recording videos of the male and female drivers in outdoor conditions. The videos are taken in real and varying illumination conditions. The Canon A720Is digital camera with the resolution set at 640x480 pixels and 24-bit true color (RGB) was used to record videos at 30 frames per second. This recording setting of the video collection results in a video that has similar characteristics to the video produced by real driver monitoring systems (ABTAHI et al., 2014). Figure 5.1 provides some examples of the diversity in the dataset.

Figure 5.1: Some example frames from the dataset with drivers performing different actions.

(a) (b)

(c) (d)

(e) (f)

Apart from the dataset and machine configuration, each operating mode has its optimal settings, which are presented in their corresponding sections 5.2.1 and 5.2.2 respectively.

Table 5.1: Key characteristics of the selected videos from the dataset for detailed evaluation.

Video Sequence	Video Name	Nº of Frames	Challenging Factors
1	7-MaleGlasses.avi	2398	Fast movement
2	5-FemaleNoGlasses.avi	2149	Appearance change
3	6-MaleGlasses.avi	1633	Glasses and bad illuminations
4	4-FemaleNoGlasses.avi	1496	Occlusion and appearance change
5	3-FemaleGlasses.avi	3057	Similar background and glasses
6	9-MaleNoGlasses.avi	2020	bad illumination and face size

5.1.1 Quantitative Evaluation Strategy

The proposed face tracking algorithms are quantitatively evaluated using Center Location Error (CLE), that measures the distance between center locations of the tracked target face with the manually labeled center location of the target face that is used as the groundtruth. Furthermore, for detailed evaluation, six videos have been chosen which contain background and varied illumination. Additionally, person-specific characteristics, such as face changes, head motion, and glasses are also included. Furthermore, the total number of frames, names in the dataset and some particular characteristics of the specific video are provided in Table 5.1. These six videos have been annotated manually, which includes the target face and landmarks ($Z = 68$) on the face, nose and the eyes. The proposed face tracking methods are tested to verify if they can track the facial landmarks consistently on these videos. Hence, the error was measured by the root mean squared error (RMSE) between the estimated landmark locations (Λ_T) and the manually-labeled groundtruth (Λ_G) locations of the landmarks as follows:

$$\varepsilon(t) = \frac{1}{Z}\sum_{i=1}^{Z}||\Lambda_G^{(i)}(t) - \Lambda_T^{(i)}(t)||_2, \tag{5.1}$$

where $\varepsilon(t)$ represents the tracking error of the current frame at time t, whereas i is the i^{th} landmark and $\Lambda_G^{(i)}$, and $\Lambda_T^{(i)}$ represent the ground truth and estimated location in (x, y) of the i^{th} landmark.

5.1.2 Choice of Batch Size

In the object tracking methods that learn the appearance of the tracked target object incrementally, the batch size plays an important role. Batch size describes that after how many frames the appearance model is updated. Different batch sizes have been tested to optimize the performance as shown in Table 5.2. The second and third row represents the average RMSE (ε_M) in a completed video sequence using the method mentioned in the first column, and the last row represents the number of times the methods consult the CLM for correction. In the case of AFTRM and AFTRM-W, the test to predict error is made after a specific number of frames (i.e., batch size τ), for the method to be consistent. For this reason, it is expected that fewer tests are made to predict error with higher batch size, leads to less resync using CLM. Similarly, Table 5.3 provides the number of frames per second for the proposed method of its operating modes. It can be seen that the number of frames increases from left to the right, which indicates bigger the batch size, more number of frames are processed per second. This is due to the update of the dictionaries and resyncing scheme, which is time-consuming. However, on the other side, a smaller batch size has a smaller error. Therefore, a trade-off between the batch size and the acceptance of error is necessary.

The same phenomenon of batch size τ with the ε_M and number of resyncs r is shown in the Figure 5.2 (a, b, c, d, e, f) and Figure 5.3 for different batch sizes ($1 \leq \tau \leq 16$). Figure 5.2 $(a - f)$ shows examples of individual videos, whereas Figure 5.3 shows the plot of six videos together. The size of the triangle indicates the batch size, which means after how many frames the resync of the features is performed (larger the size of the triangle, bigger batch size). It can be seen that larger batch size (big triangles) requires less number of resync, but it confers higher error and vice versa. It can also be noted that small triangles tend to lie on the upper left (upper for a large number of resyncs and left confers to smaller error) of the plot, which shows that more resyncs are required, and the error is low.

The experiments shown in Figure 5.2, Table 5.2, and Figure 5.3 indicate that frequent updates (small batch size) on the basis of the proposed method has less tracking error than for large batch size. The reason for this behavior is that it updates the most recent appearance of the face and also the resync (if required) of the features are performed after a specific number of frames. The optimal trade-off is the batch size that minimizes

both the number of resyncs (r) and the tracking error ε, which is defined as:

$$\begin{aligned} &\arg\min \forall_{\tau=1}^{n}\, c(r_\tau, \varepsilon_\tau), \\ &s.t.\;\; c(r_\tau, \varepsilon_\tau) = (1-\kappa) r_\tau + \kappa \varepsilon_\tau, \end{aligned} \tag{5.2}$$

where c indicates a cost function, n is the total number of batch sizes (n=16 in the current experiments), and κ is a bias between the tracking error ε and number of resyncs r (κ=0.5 in the current experiments).

Table 5.2: Average RMSE ε_M and number of times resync is activated for different batch sizes τ.

Method	Batch Size (τ)									
	1	2	3	4	5	6	7	8	9	10
MMDL-FT	10.39	**7.72**	*9.81*	14.43	14.25	13.79	16.76	40.34	27.53	21.67
MMDL-FTU	**5.46**	*6.27*	6.31	8.29	7.03	6.43	21.78	36.48	29.04	30.44
AFTRM	**5.65**	7.03	*6.89*	6.92	7.03	7.27	7.23	7.32	7.62	7.77
AFTRM-W	**3.02**	3.58	*3.19*	3.48	3.65	3.98	4.33	4.06	4.56	5.88
N^o of resync	1002	474	279	231	194	171	161	146	*117*	**108**

5.2 Experimental Qualitative Results

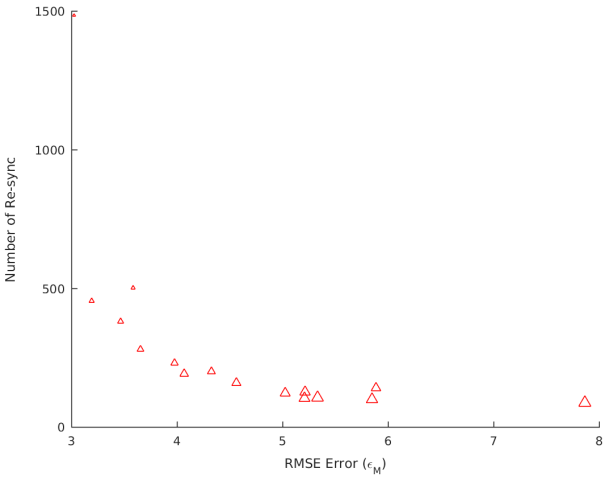
This section presents the qualitative results of the proposed face tracking method in its two operating modes, i.e., MMDL-FTU operating mode and AFTRM-W operating mode. Each operating mode has its specific parameter settings and the qualitative outputs. For this reason, the qualitative experimental results of the MMDL-FTU operating mode

Table 5.3: Average frames per second (fps) and number of times resync is activated for different batch sizes τ.

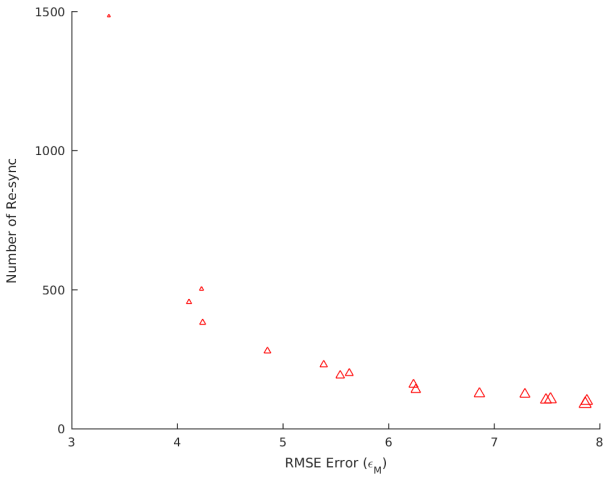
Method	Batch Size (τ)									
	1	2	3	4	5	6	7	8	9	10
MMDL-FT	3.38	3.48	4.50	4.48	5.50	5.51	6.52	6.35	7.31	8.33
MMDL-FTU	3.67	3.69	4.88	4.80	4.67	4.93	5.32	5.26	6.62	7.57
ILFT	15.27	17.12	16.68	17.91	18.33	16.11	16.37	18.79	18.80	18.89
AFTRM	0.09	0.21	0.25	0.31	0.29	0.38	0.53	0.62	0.79	0.98
AFTRM-W	0.07	0.23	0.20	0.23	0.33	0.45	0.56	0.54	0.83	1.10
N^o of resync	1002	474	279	231	194	171	161	146	*117*	**108**

Figure 5.2: Batch size effect on error (ε) and number of resyncs (r).

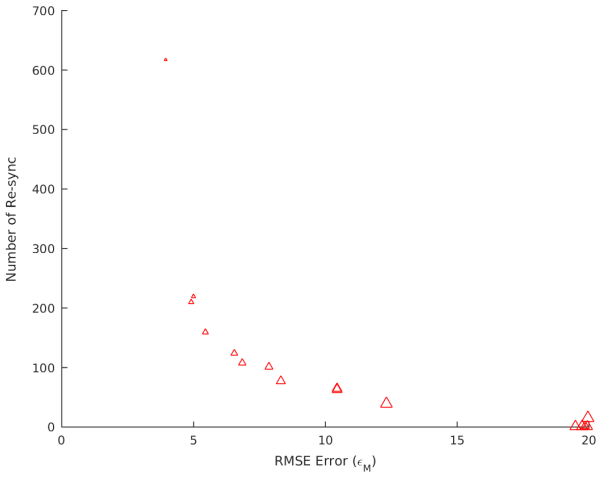
(a) video i;



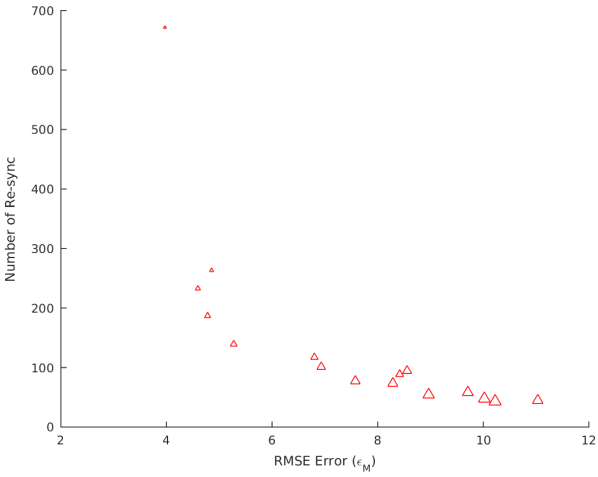
(b) video ii;



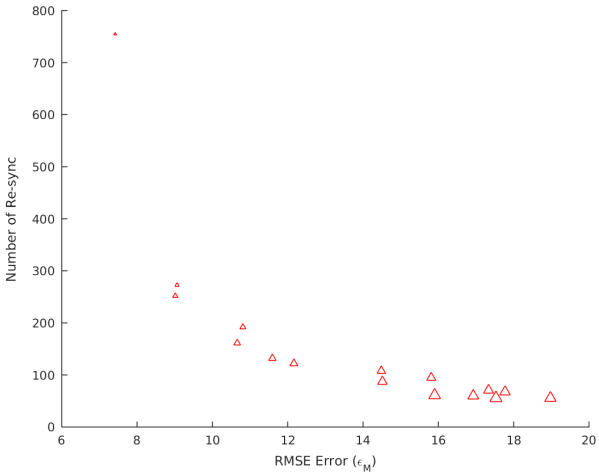
(c) video iii;



(d) video iv;



(e) video v;



(f) video vi.

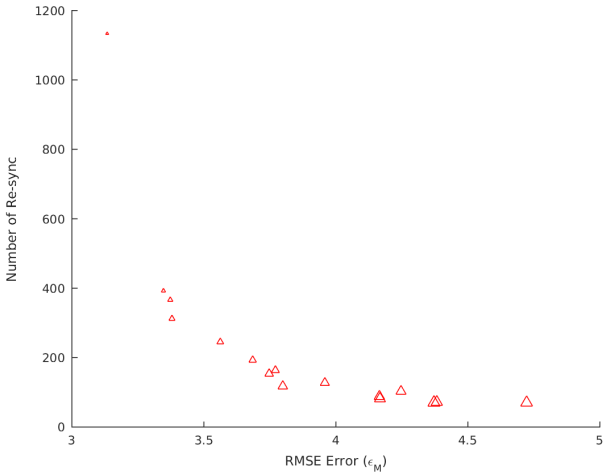


Figure 5.3: Batch size effect on error (ε) and number of resyncs (r) in multiple videos (normalized to [0,1]).

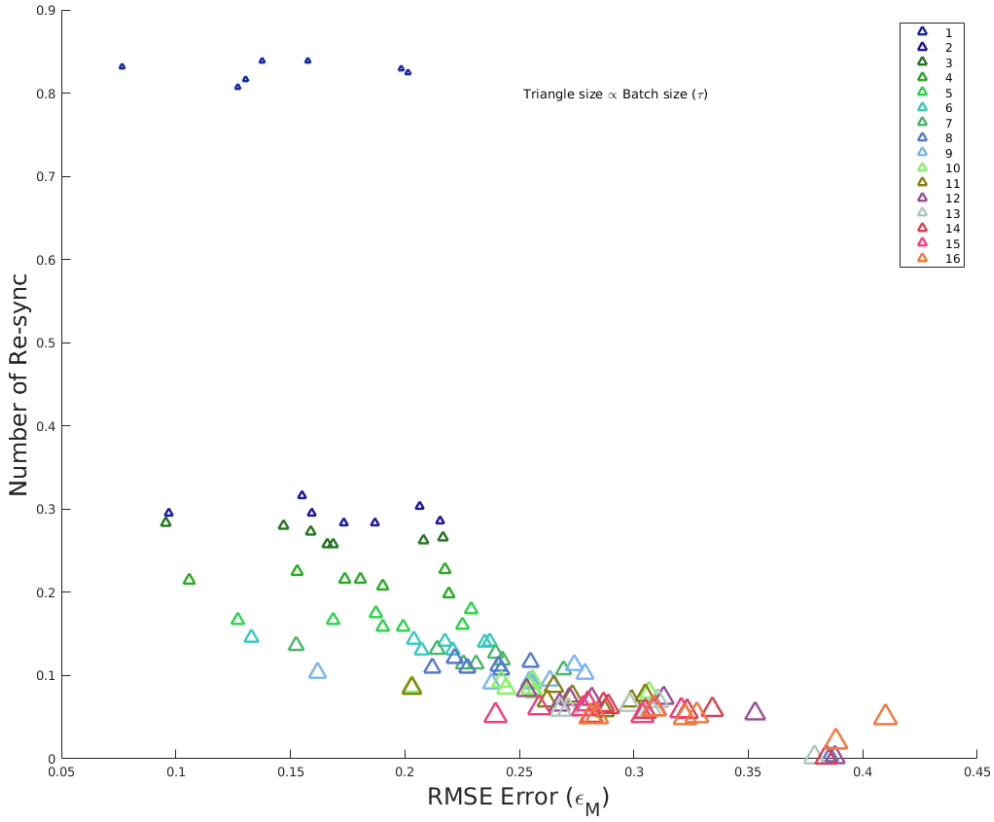
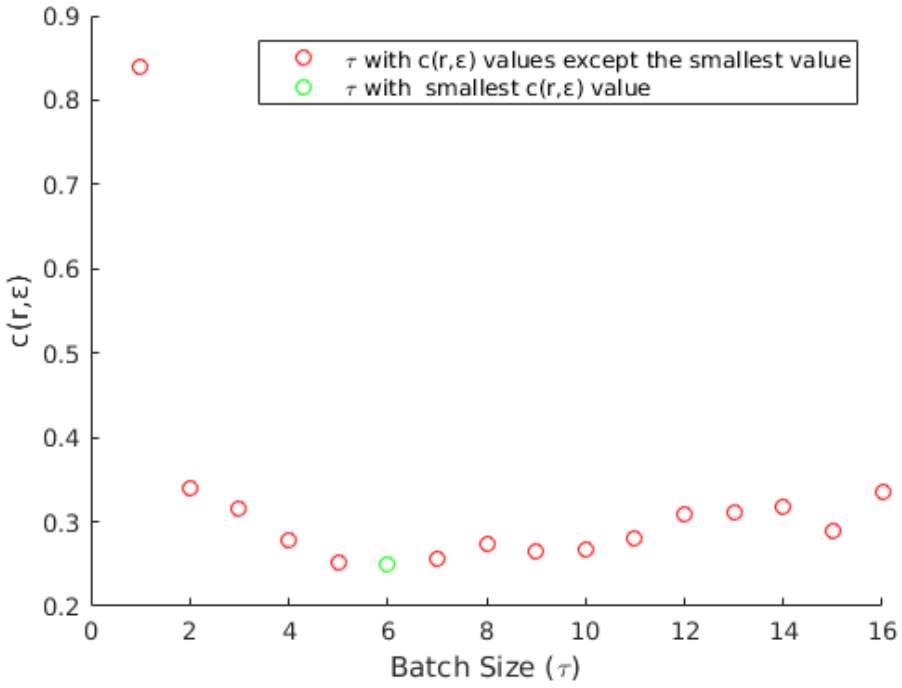


Figure 5.4: An example plot of cost function $c(r_\tau, \varepsilon_\tau)$ and batch size τ (τ = [1 16]).



are presented in the Section 5.2.1, and the qualitative experimental results of AFTRM-W operating mode are presented in Section 5.2.2.

5.2.1 Qualitative Evaluation of the MMDL-FT and MMDL-FTU Face Tracking Method

In order to evaluate the MMDL-FT and MMDL-FTU, the following specific parameter settings in addition to the settings mentioned before are used in order to track the target face. To increase computational efficiency, the initial/tracked target face is resized to $32 \times 32, (u = 32)$ for learning the dictionaries. As explained earlier, two dictionaries - a reconstruction dictionary and a classification dictionary- are created. To keep both the dictionaries representing the current tracked face appearance at time t, the dictionaries update is performed after each set of three frames $(\tau = 3)$ with a forgetting factor f (f=0.95). Various values of Ω (the weight associated with the classification and reconstruction dictionaries), in the range $[0\ 1]$ have been tested, and $\Omega = 0.8$ was used in the experiments. The proposed method runs six frames per second with a patch size of $8 \times 8, (v = 8)$ and the configuration above. Besides, 400 affine parameters sample values are drawn to obtain and test the candidate target face samples (Eq. 4.2). The YawDD dataset (ABTAHI et al., 2014) was used in the experiments, which includes videos of drivers with various facial expressions and head poses in varied illumination conditions in real driving scenarios, such as neutral expression, talking, laughing, singing, yawning, and so on. The camera is installed on the dash or under the front mirror.

For the qualitative evaluation of the proposed MMDL-FTU, the video frames are presented in the form of images. Each resulting image contains a video frame with the tracked target face enclosed in a bounding box and the tracked facial landmarks plotted on the tracked target face in red color, whereas, the facial landmarks plotted on the face in yellow color are the manually labeled landmarks that are used as the ground-truth facial landmarks.

Figure 5.5 shows some examples from the experimental results obtained using the MMDL-FT operating mode of the proposed method. It can be seen that the proposed method performs well in different scenarios, including the tilted face (see Figure 5.5a), face distant from the camera due to the height and sitting position of the driver (see Figure 5.5b), partial occlusion of the face (see Figure 5.5c), change in the face size compared to the other drivers (see Figure 5.5d), visual angle (see Figure 5.5e) and illumination

changes (see Figure 5.5f. The results in the Figure 5.5 suggests that MMDL based face tracked is able to track the target face and facial landmarks in the challenging conditions.

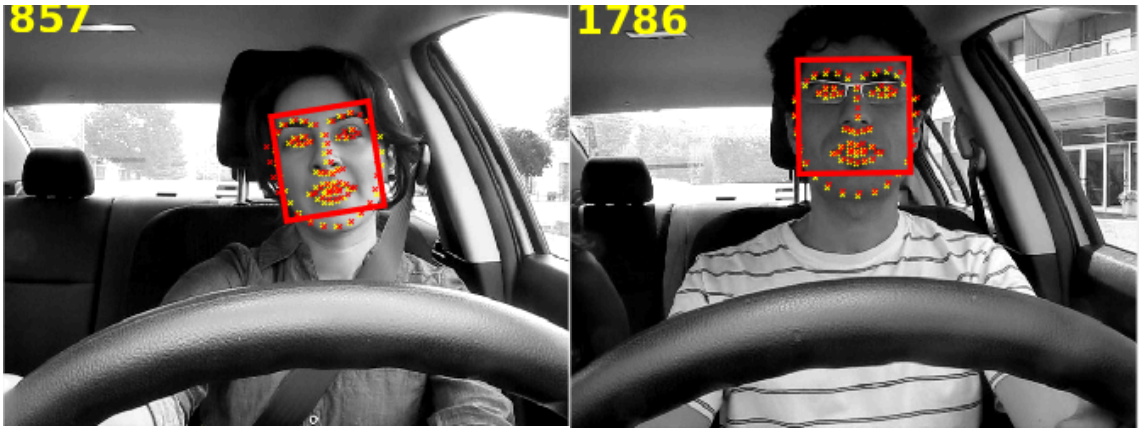
The same affine parameters are used for warping the bounding box and the facial landmarks of the tracked target face. However, sometimes the bounding box has a tilt giving an illusion that it is not correctly detected, which is actually due to the affine parameters angle to capture the face. This is due to the rotation angle $\theta(t)$ and skew direction $\phi(t)$ parameters, and this helps in finding the correct locations of the landmarks (see Figure 5.4e). In the MMDL-Face Tracker (MMDL-FT), the dictionaries are updated using the tracked target face samples collected without checking their quality, as proposed by Ross et al (ROSS et al., 2008).

Figure 5.6 provide RMSE plots of the proposed MMDL-FT, MMDL-FTU methods and the comparative methods that include Incremental Learning for Robust Visual Tracking (ILRVT) (ROSS et al., 2008) and Approximate Structured Output Learning for Constrained Local Models (CLM) (ZHENG; STURGESS; TORR, 2013). Also, Figure 5.7 shows the plot of the average RMSE of a complete tested video. It is noted that the mean tracking error increases over time in most of the videos as shown in Figure 5.7. The increase of the mean tracking error in the long videos is associated with the change in the tracking conditions during tracking. This issue is addressed by using a resyncing scheme in Section 4.3.

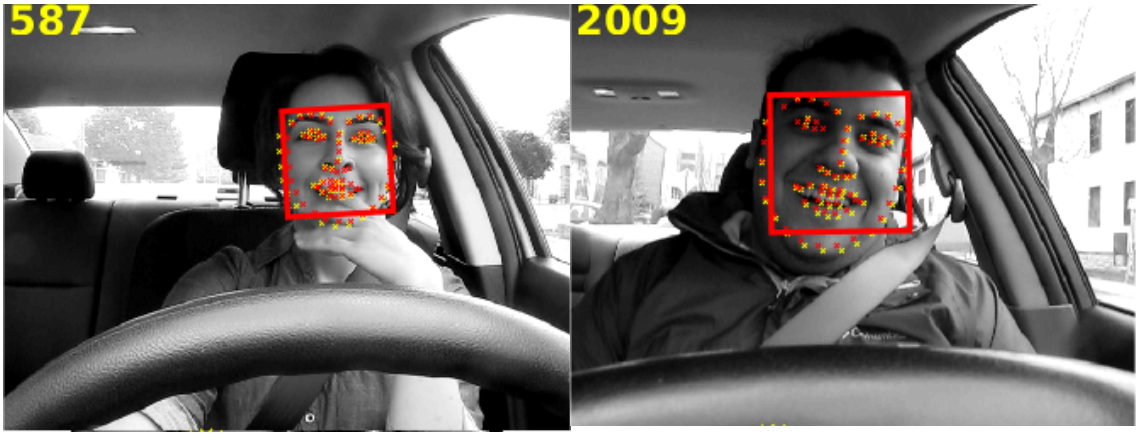
The experimental results shown in Figure 5.5, Figure 5.6,Figure 5.7 indicate that the proposed MMDL-FT and MMDL-FTU methods tend to perform better than the comparative methods, and MMDL-FTU shows slightly better performance. It can be seen that the MMDL based face tracker performs better than the comparative methods. As the dictionaries are updated frequently, so it is capable of performing better in challenging conditions. When the appearance of the tracked target face changes along time, the dictionaries are adapted to the current appearance of the target face by adding new samples to update the dictionary and also by down-weighting the contribution of the old training samples. This makes the tracker capable of representing the tracked target face better in different tracking conditions. Another important aspect of MMDL based tracker is using two dictionaries, i.e., reconstruction and classification dictionaries which makes it robust to the changes in illumination and background of the scene. The appearance change is handled by the reconstruction dictionary, and the background changes are captured by the classification dictionary that discriminates the target face from the background irrespective of the background change along time. The combination of two dictionaries to track

Figure 5.5: Results of the proposed MMDL-FTU method for different face condition evaluated in the tests, red = tracked landmarks, yellow = ground-truth landmarks.

(a) Tilted face; (b) Distance to the camera



(c) Partial occlusion; (d) Face size change;

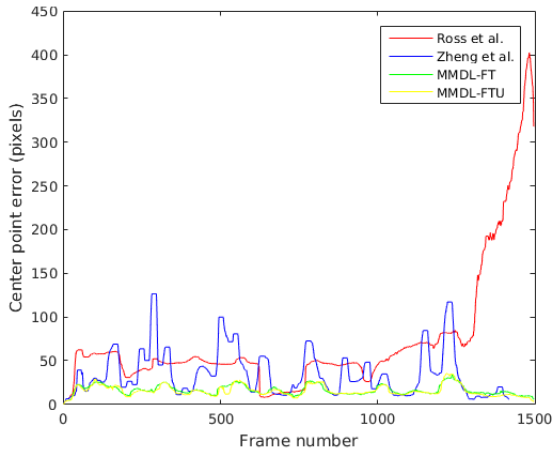


(e) Visual angle; (f) Illumination change.

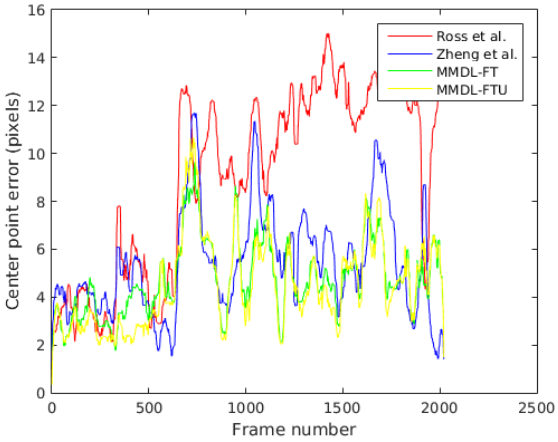
the target face improves the robustness and increases confidence score. This suggests that the utilization of the test samples to update the dictionary improve the face tracking results, and also improvement can be made in the quality assessment of the sample. Hence use only the samples which are sufficiently accurate. The experimental results obtained for face tracking suggest the potential of dictionary learning for a non-rigid object (e.g.,

Figure 5.6: RMSE across frames of 68 landmarks of the complete tested videos.

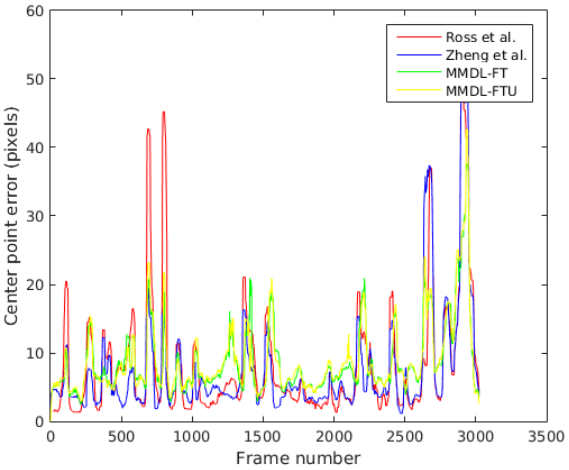
(a) video i;



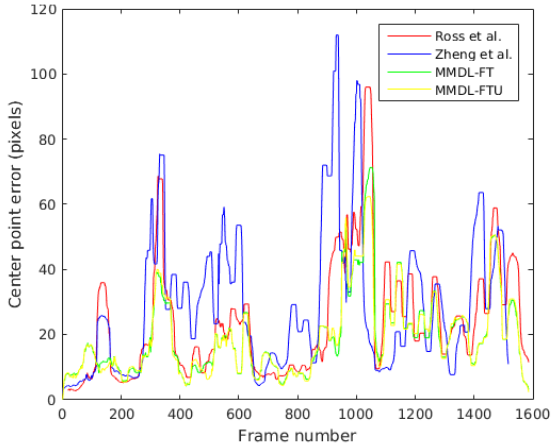
(b) video ii;



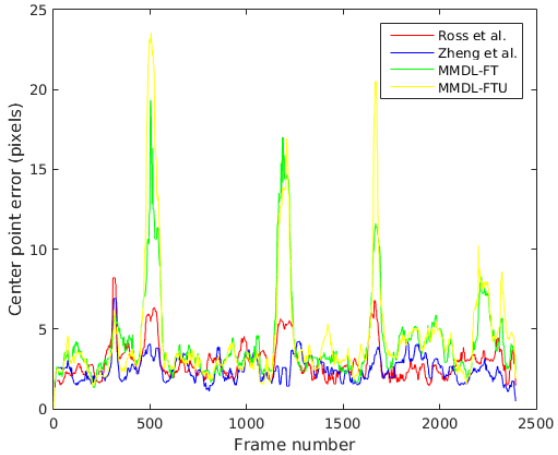
(c) video iii;



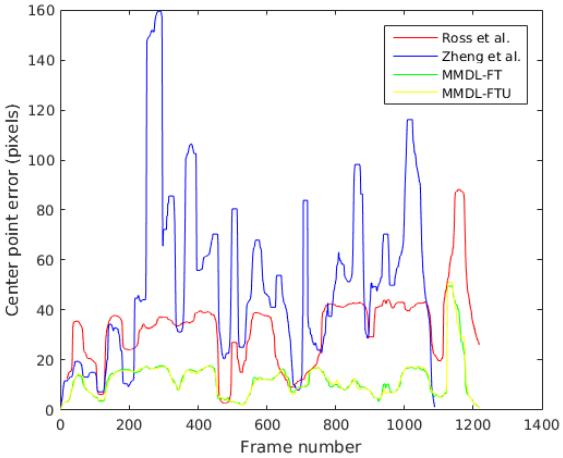
(d) video iv;



(e) video v;



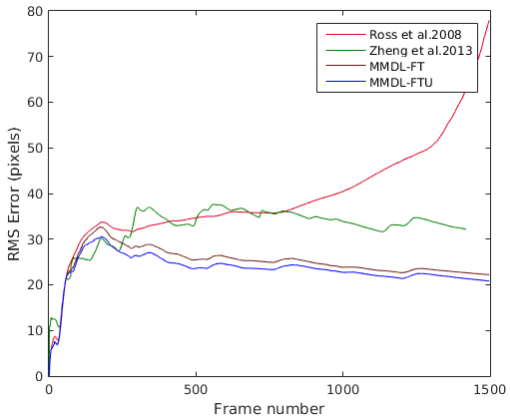
(f) video vi.



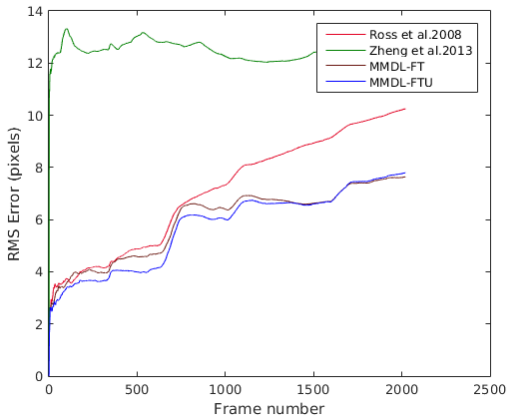
face) tracking algorithms, and provide insights for further improvements.

Figure 5.7: Mean Error across frames of all landmarks of the complete tested videos.

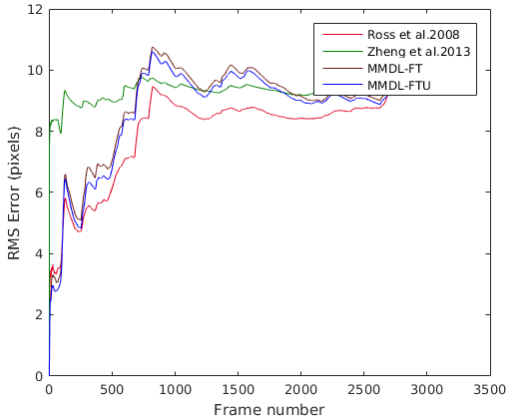
(a) video i;



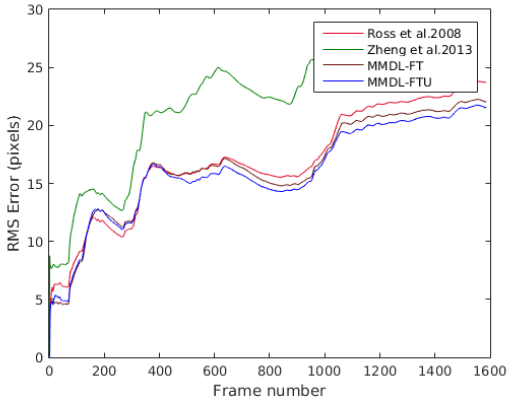
(b) video ii;



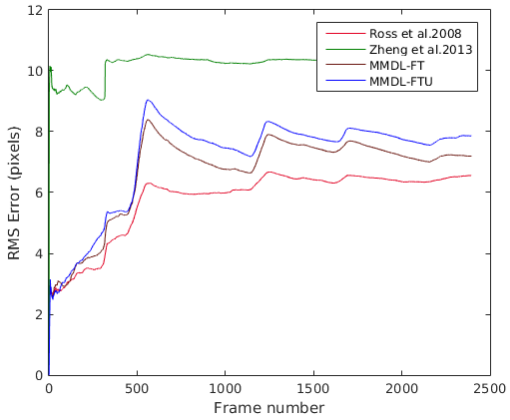
(c) video iii;



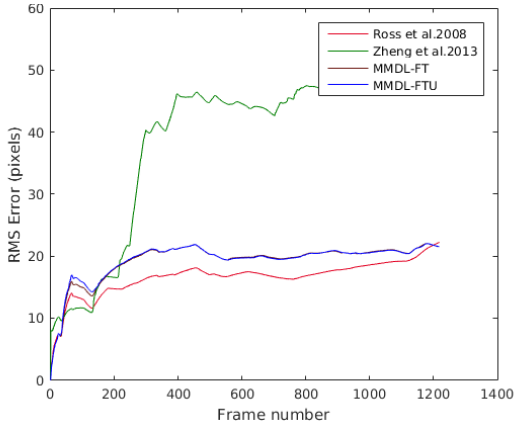
(d) video iv;



(e) video v;



(f) video vi.



5.2.2 Qualitative Evaluation of the AFTRM and AFTRM-W Face Tracking Methods

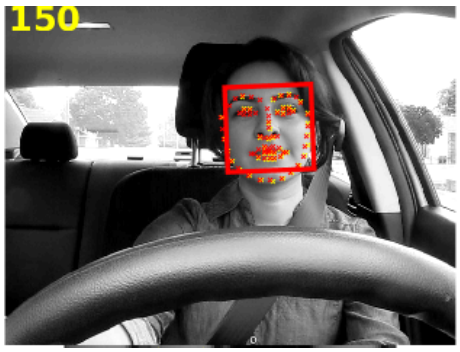
For the experimental evaluation of the AFTRM and AFTRM-W, the following setting of the parameters is used. To build and update the eigenbases, the candidate/tracked target object sample (e.g., face) is resized to $u \times u$ (u=32) for computational efficiency. In this work, the bases are built using the local patches of size $v \times v$ (v=8), the number of eigenvectors is set to γ (γ=16), and the eigenbases are updated every five frames ($\tau = 5$). In order to down-weight the contributions of the old samples, the forgetting factor (f) is set to 0.95.

In the proposed AFTRM and AFTRM-W methods, the batch size τ is set to 5, for efficient execution of the tracking algorithm and also performs adequately accurate. For the qualitative evaluation of the proposed tracking scheme, each image is displayed with three layers, that are separately explained in Figure 5.8. The first layer (Figure 5.8a) is a video frame that contains the tracked target face enclosed in a bounding box and the tracked landmarks (features) are plotted on the tracked target face in red color, whereas, the landmarks plotted on the face in yellow color are the ground-truth landmarks; the second layer (Figure 5.8b) is comprised of four sub-image blocks, the first block shows the mean face ($\mu(t)$), the second is the tracked face with groundtruth landmarks laid on it, followed by the reconstruction error and the last block shows a reconstructed face with error removed; finally, the third layer (Figure 5.8c) shows some important eigenvectors, most important is shown first and so on.

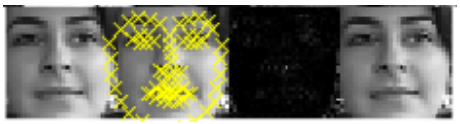
Some visual results showing a failure of the proposed tracking method without the error predictor and resync procedure are shown in Figure 5.9 and Figure 5.10. It should be noted that the figures only show results in which the proposed method without resync fails. It shows only the challenging scenarios for ILFT method (AFTRM operating mode without error prediction and resyncing mechanism) in which this method fails. Figure 5.9a, shows the initialization of the tracking process. The eigenbases are not shown as there are not enough number (τ) of the tracked target face samples to build the basis yet. In Figure 5.9b, as the the occlusion of the face happens, the tracker is not able to capture well the occluded face, in consequence, the tracker is not able to detect the face and facial landmarks correctly. The deformation in the face also affects the tracking procedure as can be seen in Figure 5.9c. As the human face is a highly deformable object and when the deformation is not captured correctly, it is difficult to perform a facial analysis. The

Figure 5.8: Example figure showing face bounding box, facial landmarks, mean face, tracked target face, reconstruction error, reconstructed face and most important eigenbases.

(a) Face bounding box and estimated landmarks (In red color), Groundtruth landmarks (in yellow color);



(b) From left to right: mean face, tracked target face, reconstruction error, reconstructed face;



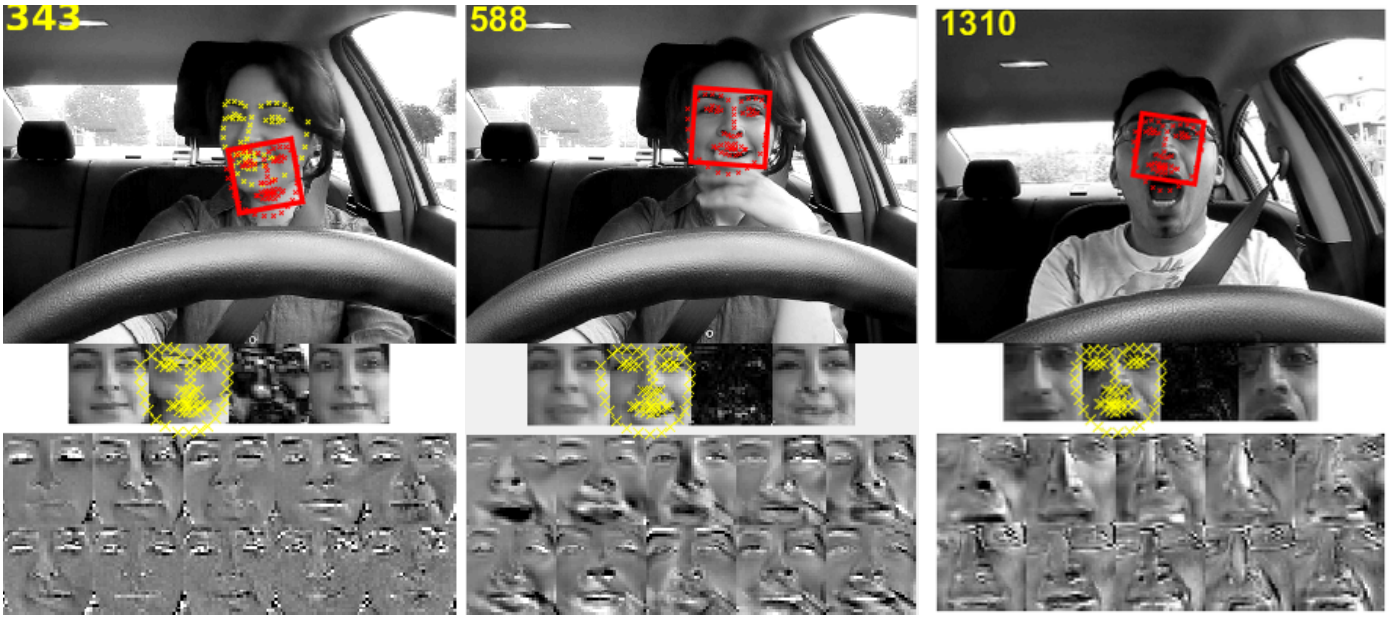
(c) Most important eigenbases.

face movement, appearance or shape changes, illumination changes along time affects the tracker, and hence it may result in complete wrong detection. The behavior just explained results in failure such as can seen in Figure 5.9d. The face size varies from person to person and also the distance from the camera can make the face size look different due to the back and forth movement of a person. This also affects the tracker, as can be seen in Figure 5.9e-a small sized face and Figure 5.9f-large face size and also contains tilt. Similarly, Figure 5.10 gives some more examples that may cause the failure of the tracker to detect the face correctly such as distance of the face from the camera (Figure 5.10a), the combination of tilt and face deformation (Figure 5.10b), face angle to the camera (Figure 5.10c), combination of the face angle with another face behind the tracked target face (Figure 5.10d), combination of face movement and deformation (Figure 5.10e), etc.

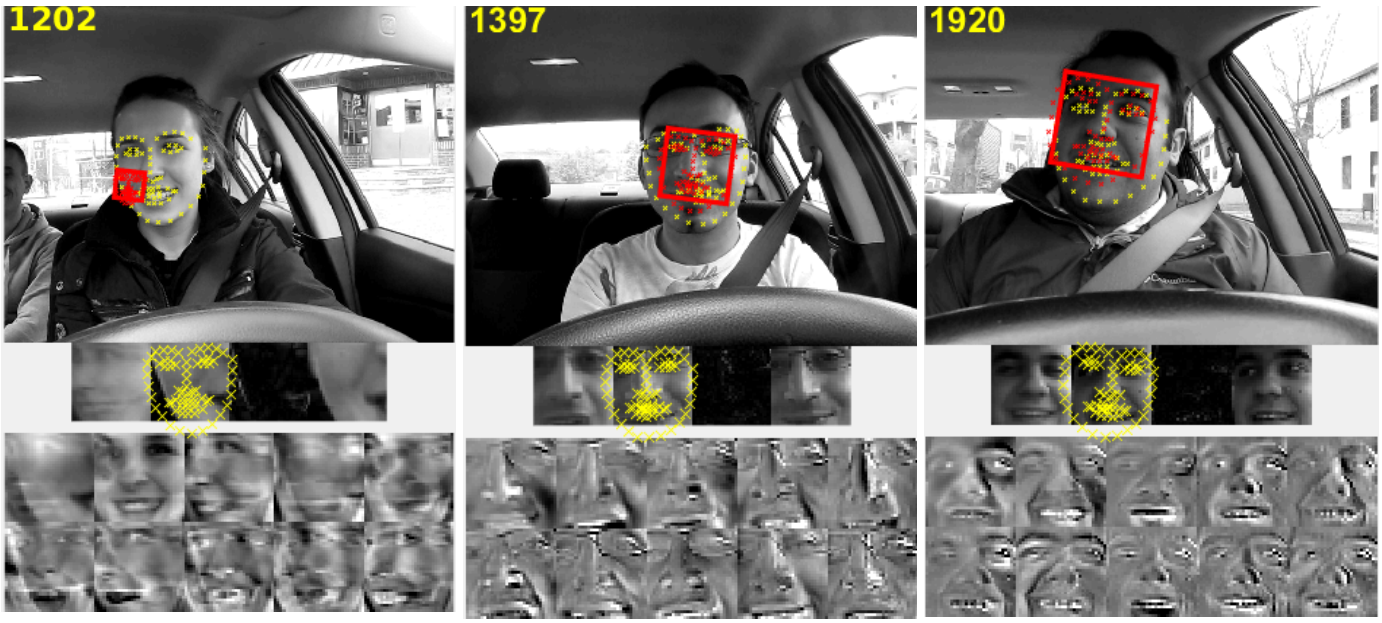
Figure 5.9 and Figure 5.10 indicates that in challenging conditions, the error is increased. Furthermore, once the tracking error is introduced, it keeps on increasing.

Figure 5.9: Results of the Incremental Learning for Face Tracking Algorithm without Resync (ILFT).

(a) face movement; (b) partial occlusion; (c) face deformation;



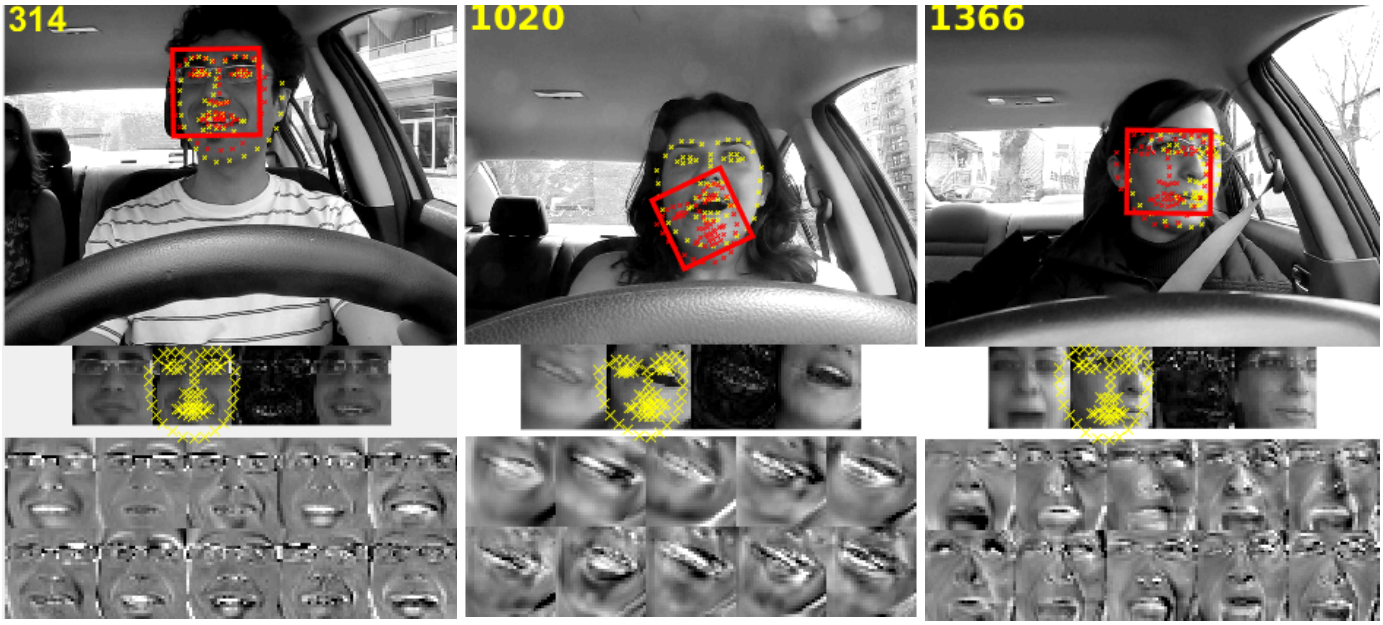
(d) face movement; (e) face size; (f) tilted face.



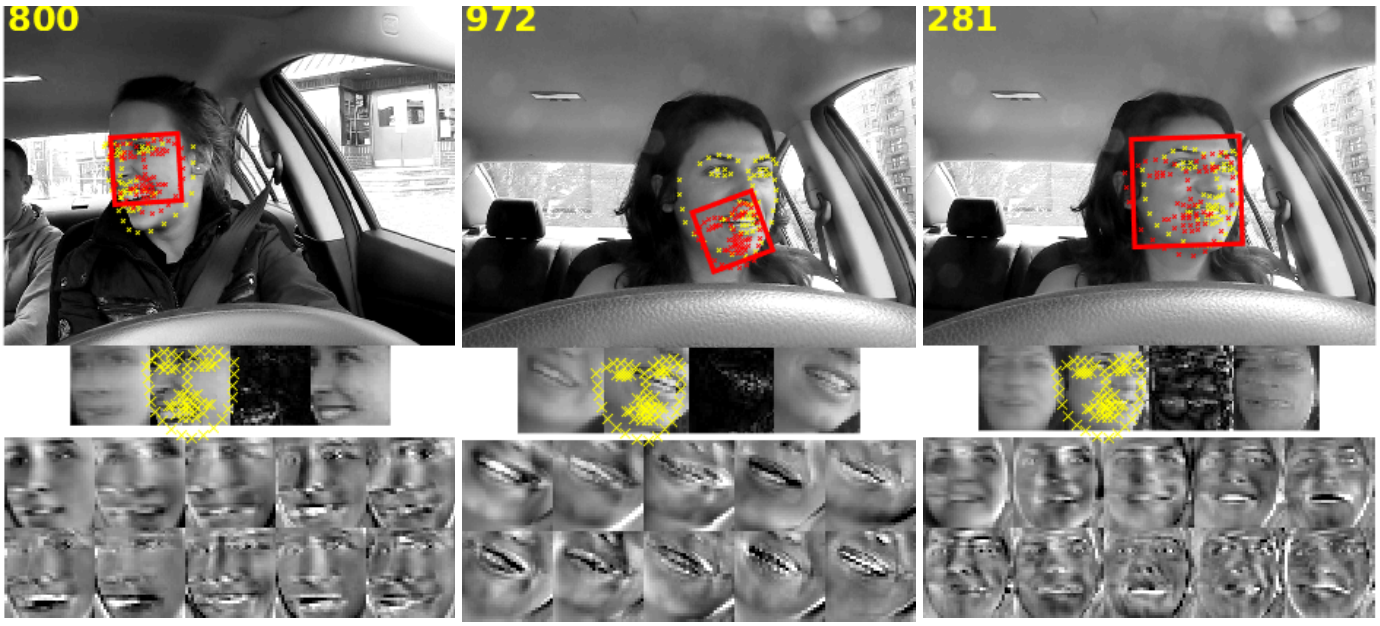
The reason for this behavior is that the tracker is blind to the tracking error and does not contain any information about the tracking error. Also, as the method utilizes the test data to update the basis in order to adapt with the environment changes, this may affect the tracking quality if the samples used for training does not contain accurate information. In this method, the tracked target samples that are used as training data are not assessed for the quality, and the possibility of utilizing error-prone data to update the basis is always there. In result, the eigenbases are built on slightly wrong tracked target face samples, as

Figure 5.10: Results of the Incremental Learning for Face Tracking Algorithm without Resync (ILFT).

(a) face distant from camera; (b) tilt and deformation; (c) angled face;



(d) angled face with another face behind; (e) face movement and deformation; (f) face movement.

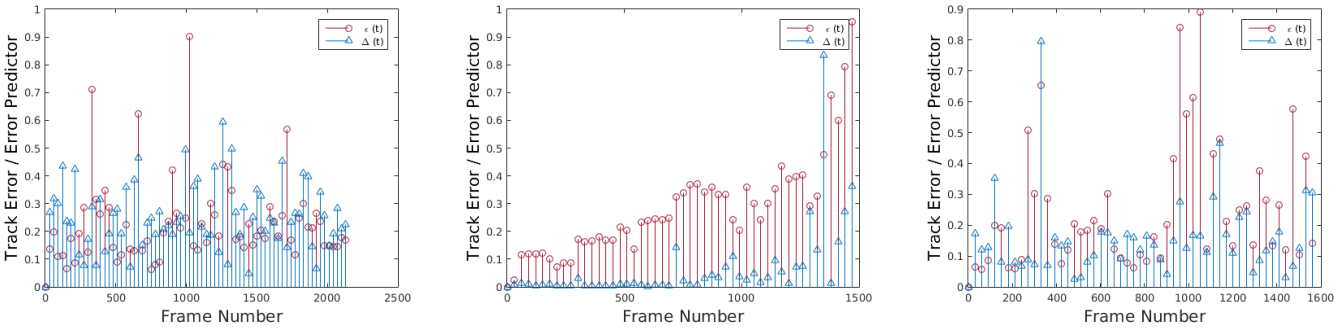


the error was introduced but not known to the tracker. However, not updating the basis is a waste of the useful data that is available during tracking and also the tracker does not make any improvement in learning the environment. Nonetheless, these issues can be addressed, if the tracker has the knowledge of the tracking error.

In this method, an error predictor is proposed that can be used to indicate when the error is high, and a resyncing of the important features (in current experiments, facial landmarks) can be performed in order to correct the tracking procedure (explained in

Figure 5.11: Plot of $\Delta(t)$ and $\varepsilon(t)$.

(a) Correlation= 0.2166; (b) Correlation= 0.5457; (c) Correlation= 0.1347.

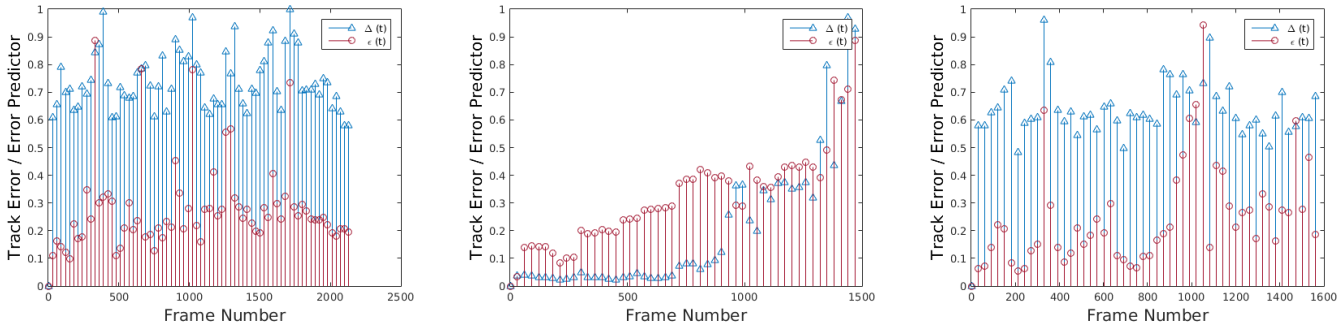


Section 4.3.6). Fig. 5.11, shows an example of the prospective error predictor, which is the root mean square (RMS) distance of the Z landmarks in the consecutive frames and the RMS error $\varepsilon(t)$ of the tracked landmarks. The plots in Fig. 5.11(a,b and c) suggest some correlation (ρ) between the tracking error ($\varepsilon(t)$) and the tracked points difference ($\Delta(t)$). However, the data is noisy, and the correlation is low. The median filter is usually used to remove noise and spikes in the data. Due to the spiky nature of $\Delta(t)$ and the tracking error $\varepsilon(t)$, an one dimensional median filter of fifth order $\overline{\Delta}(t)$=$\{\Delta(t)-\tau, \Delta(t)-\tau+1, ..., \Delta(t)\}$) is applied over a sliding window of τ frames to smooth consistently $\Delta(t)$, increasing the correlation between $\varepsilon(t)$ and $\Delta(t)$, as shown in Fig. 5.12(a,b and c). It can be seen that the filtered $\overline{\Delta}(t)$ and $\overline{\varepsilon}(t)$ have higher correlation because the data is smoothed and has fewer spikes. To further improve the tracking error prediction, a median filter of fifth order is applied over a sliding window of τ previous values of $\overline{\Delta}(t)$ i.e., ($\hat{\Delta}(t)$=$\{\overline{\Delta}(t)-\tau, \overline{\Delta}(t)-\tau+1, ..., \overline{\Delta}(t)\}$). The correlation between the $\hat{\Delta}(t)$ and $\hat{\varepsilon}(t)$ is improved, which can be seen in Fig. 5.13(a, b and c). This behavior of $\hat{\Delta}(t)$ and $\hat{\varepsilon}(t)$ suggests that the tracking process can be improved automatically. Using the proposed error predictor, the tracking quality can be analyzed, and the resyncing of the tracking point locations can use W-CLM when necessary.

Some results of the proposed tracking algorithm AFTRM-W are shown in Figure 5.14 and Figure 5.15, that utilizes the proposed predictor to estimate the error and resync the important features using W-CLM when the error is estimated to be high. It can be seen that the tracking works correctly in the long sequences even if there is a movement in the face (Figure 5.14(a and d)), when the face is partially occluded (Figure 5.14(b) face deformation (Figure 5.14(c)), change in face size (Figure 5.14($e-f$)), tilted face (Figure 5.14(f)), distant face from the camera (Figure 5.15(a)) or combination of these factors such as: tilt and deformation (Figure 5.15(b)), angled face with another face in

Figure 5.12: Plot of $\overline{\Delta}(t)$ and $\overline{\varepsilon}(t)$.

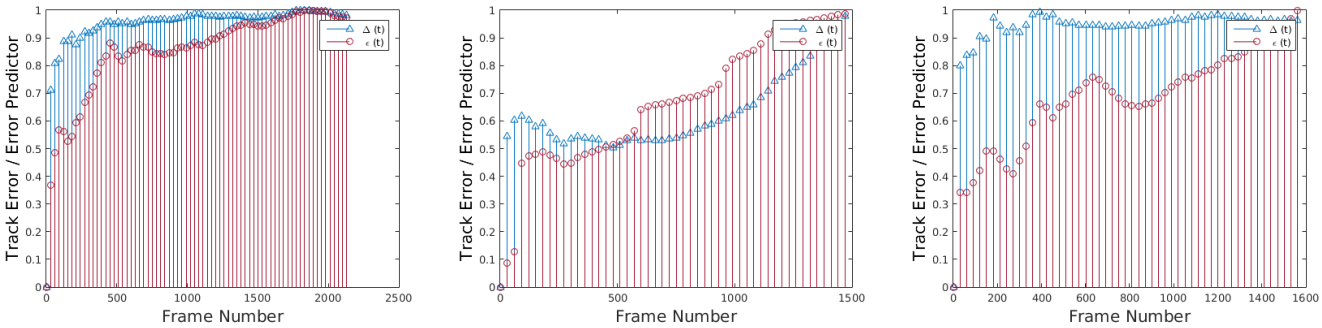
(a) Correlation=0.4644; (b) Correlation= 0.7889; (c) Correlation= 0.3148.



the frame (Figure 5.15(d)) or angled face with face deformation (Figure 5.15(e)). Figure 5.14(b) and Figure 5.14(c) show examples of recent resyncing using CLM, therefore the old eigenbases are disregarded. When the resyncing is performed, the old data and the eigenbases are disregarded because they are not able to represent the current appearance of the tracked target face as indicated by the proposed error predictor. Furthermore, the proposed method is capable of handling sufficient front, side and vertical movement in the face.

Figure 5.13: Plot of $\hat{\Delta}(t)$ and $\hat{\varepsilon}(t)$.

(a) Correlation=0.91037; (b) Correlation= 0.7592; (c) Correlation=0.7673.

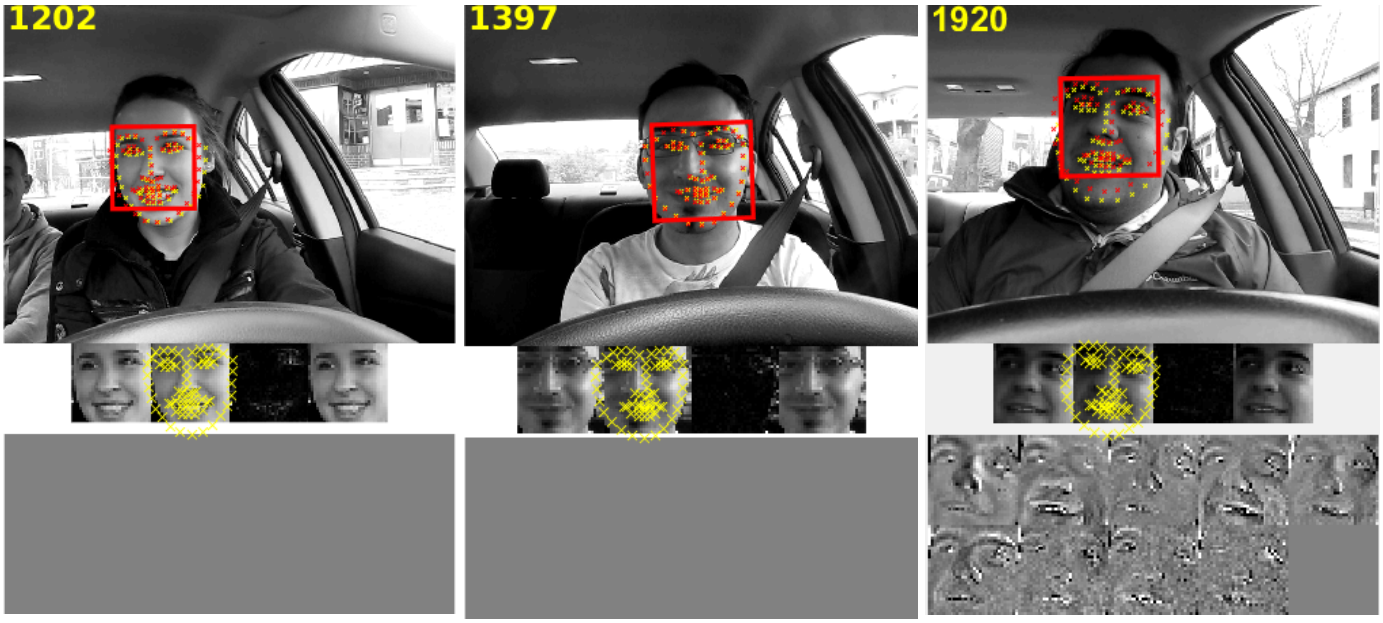


5.3 Quantitative Evaluation and Discussion of the Proposed MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W Face Tracking Methods

The proposed tracking methods are compared with the state of the art method using the root mean squared error (RMSE) of the estimated landmark locations with the manually-labeled landmarks locations used as the groundtruth. The quantitative comparison of the proposed MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W (AFTRM with

Figure 5.14: Results of the proposed Adaptive Face Tracker with Resyncing Mechanism (AFTRM-W).
(a) face movement; (b) partial occlusion; (c) face deformation;

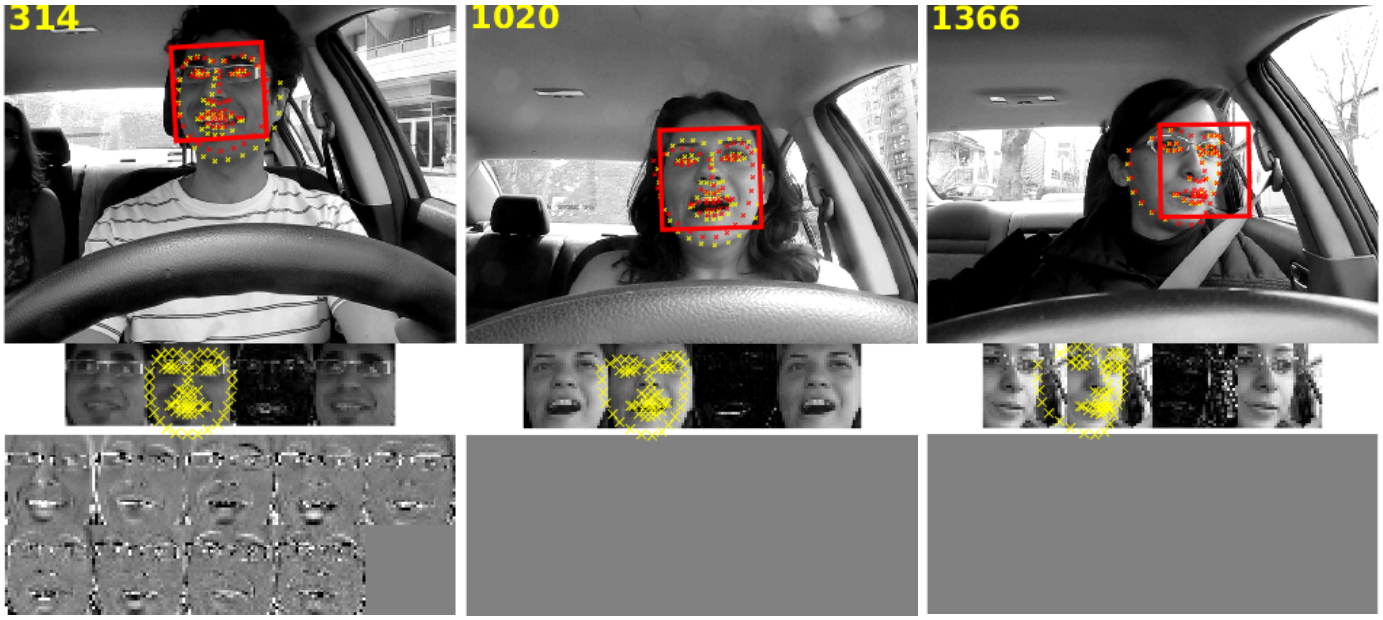
(d) face movement; (e) face size; (f) tilted face.



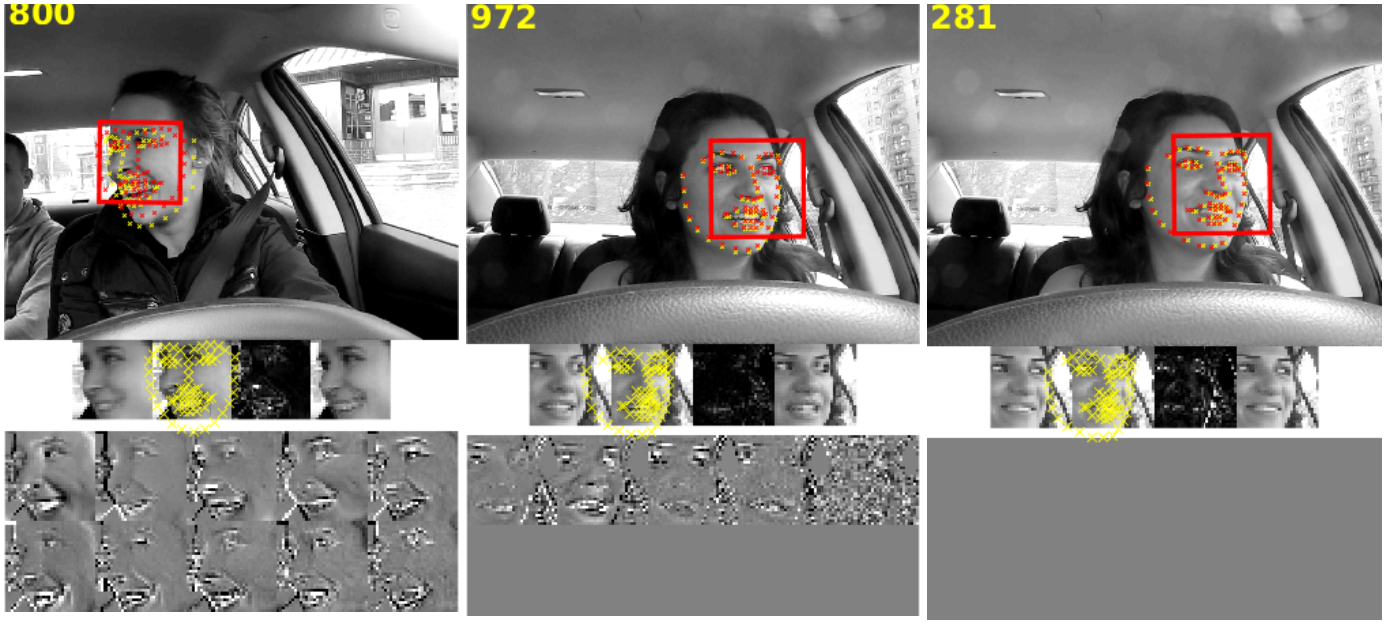
the weighted CLM), with the Incremental Learning for Robust Visual Tracking (ILRVT) (ROSS et al., 2008), incremental learning tracking based on Independent Component Analysis (ILICA), Incremental Cascaded Continuous Regression (iCCR) (SÁNCHEZ-LOZANO et al., 2016) and Approximate structured output learning for CLM (ZHENG; STURGESS; TORR, 2013).

The results in terms of RMSE of the 68 facial landmarks tracked using the proposed MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W methods with respect to the

Figure 5.15: Results of the proposed Adaptive Face Tracker with Resyncing Mechanism (AFTRM-W).
(a) distant face; (b) tilt and deformation; (c) angled face;



(d) angled face with another face behind; (e) face movement and deformation; (f) face movement.



ground-truth facial landmarks are compared with the comparative methods are shown in Table 5.4. Each column indicates average tracking error (ε_M) for the whole video sequence, using the method specified in the first row. The last row illustrates the average tracking error of all the tested videos. It should be noted that for incremental learning approaches, the parameters of the comparative methods (if required) are set to the default values as proposed in the particular method. The **bold** value indicates the best result and the second best results are in *italic* font. Similarly, Table 5.5 compares the CLE of the pro-

posed MMDL-FTU with the state-of-the-art methods based on all the videos of YawDD dataset with the camera installed on dash.

It can be seen from the Table 5.4 and Table 5.5 that proposed MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W outperform the other methods, whereas, AFTRM-W has much-improved performance than AFTRM. This is due to the weighting mechanism, as consistent landmarks receive higher weight and thus improves the quality of the resyncing mechanism using W-CLM search process. The methods proposed by zheng et al. (ZHENG; STURGESS; TORR, 2013) and sanchez et al. (SÁNCHEZ-LOZANO et al., 2016) have close results to the proposed MMDL-FT, MMDL-FTU, AFTRM method for some videos, whereas, AFTRM-W has performed better than all the other methods on five videos out of six videos and had much smaller tracking error. The ILRVT method (ROSS et al., 2008) performed well for some videos but fails for others. MMDL-FT and MMDL-FTU perform better than the comparative methods other than the proposed AFTRM and AFTRM-W methods. The high tracking error presented by these methods is because once the error is introduced in these approaches, it keeps on increasing and ultimately the tracking fails. The proposed methods AFTRM and AFTRM-W have less tracking error and does not fail to track the face indefinitely in any of the tested video sequences because of its tracking error predictor and resyncing mechanism. For this reason, the proposed tracking methods can be used for consistent tracking of the face and the facial feature in long videos, which can be used to detect expression, fatigue, and learn the cognitive behavior of human using facial features. Also, the face tracking methods perform better on female videos, because of the smooth texture compared to male videos, that may have different styles of beard, mustache, etc.

Table 5.4: Average RMSE comparison of MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W with comparative methods (the best results are in bold).

Video	1	2	3	4	5	6	Average
(TERISSI; GÓMEZ, 2007)	38.43	26.93	50.38	66.44	66.12	16.75	34.24
(ROSS et al., 2008)	21.43	10.56	183.72	30.12	6.23	12.17	44.04
(ZHENG; STURGESS; TORR, 2013)	33.93	11.46	12.41	17.05	12.26	14.02	16.86
(SÁNCHEZ-LOZANO et al., 2016)	16.42	11.48	10.33	22.07	14.49	9.84	14.10
MMDL-FT	10.12	7.19	**7.63**	22.02	8.06	30.75	14.29
MMDL-FTU	*9.73*	*6.50*	*7.76*	16.62	7.76	19.37	11.29
AFTRM	15.01	9.22	13.78	*15.31*	*5.91*	*7.53*	*11.12*
AFTRM-W	**6.54**	**3.56**	10.65	**5.27**	**4.85**	**3.62**	**5.65**

Table 5.5: Center Location Error (CLE) comparison of MMDL-FT, MMDL-FTU, AFTRM and AFTRM-W with comparative methods (the best results are in bold).

Video	Male videos	Female videos	Average
(TERISSI; GÓMEZ, 2007)	25.92	18.37	22.15
(ROSS et al., 2008)	14.74	11.33	13.03
(ZHENG; STURGESS; TORR, 2013)	13.02	10.14	11.58
(SÁNCHEZ-LOZANO et al., 2016)	14.11	10.17	12.14
MMDL-FT	10.61	8.70	9.65
MMDL-FTU	10.36	8.68	9.52
AFTRM	*8.81*	*7.54*	*8.18*
AFTRM-W	**5.31**	**4.24**	**4.78**

5.4 Evaluation of the Proposed Face Tracking Method in Yawning Detection

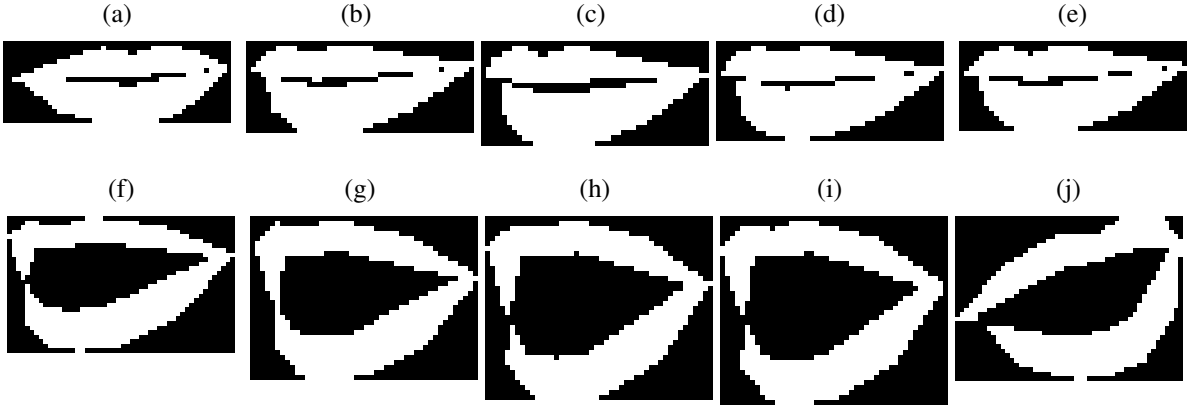
In the experiments, yawning detection is used as a case study to evaluate the correctness and effectiveness of the proposed tracking method in a real facial analysis problem, where the local face appearance is changing. The proposed method takes inspiration from Omidyeganeh et al. (OMIDYEGANEH et al., 2016) method of yawning detection. The method in (OMIDYEGANEH et al., 2016) is based on the backprojection theory and detects yawning based on the pixels counts in the binary mouth blocks of the current and reference frames. To convert into a binary image, the pixel values greater than a certain threshold Γ_0 receive a value of 1 (referred as a white pixel), and 0 (referred as a black pixel) otherwise.

The proposed method improves the method in Omidyeganeh et al. (OMIDYEGANEH et al., 2016) in two ways. Firstly, the proposed method uses only the pixels which are in the lips to measure the mouth openness in a binary image (see Fig. 5.16, only the pixels inside the white region are used), as compared to (OMIDYEGANEH et al., 2016) which uses a rectangular mouth block and includes some pixels outside the lips to detect yawning.

Secondly, Yawning is detected in each video frame if the following three conditions are satisfied:

1. the ratio of the number of black pixels in the current frame (NBC) and the number

Figure 5.16: Illustration of a closed mouth (a,b,c,d,e) and yawning sequence (f,g,h,i,j).



of black pixels the reference frame (NBR) is greater than Γ_1:

$$\frac{NBC}{NBR} > \Gamma_1, \tag{5.3}$$

2. the ratio of the number of black and the number of white pixels (NWC) in the current frame is greater than Γ_2:

$$\frac{NBC}{NWC} > \Gamma_2, \tag{5.4}$$

3. the ratio of a vertical distance between the midpoints (VD) and the distance between the corner points (HD) of the mouth is greater than Γ_3:

$$\frac{VD}{HD} > \Gamma_3, \tag{5.5}$$

where NBC and NBR is the total number of black pixels in the current and the reference frames mouth respectively, whereas, NWC is the number of white pixels in the current frames mouth, HD is the horizontal distance between mouth corners and VD is the vertical distance between the center points of lips. The first frame is used as a reference in the proposed scheme and is assumed to contain a closed mouth. Some examples of the yawning and non-yawning sequences are given in Figure 5.16, where Figure 5.16 $(a-e)$ shows a non-yawning mouth blocks and Figure 5.16 $(f-j)$ shows some yawning mouth blocks.

The proposed yawning detection is evaluated in terms of;

1. the True Positive Rate (TPR) which is the rate of True Positives (TP) detected as

yawning, and is given by:

$$TPR = \frac{TP}{TP + FN}; \tag{5.6}$$

2. the True Negative Rate (TNR) which is the rate of True Negatives (TN) correctly detected as non-yawning, and is given by:

$$TNR = \frac{FP}{FP + TN}; \tag{5.7}$$

3. the False Positive Rate (FPR) is the rate of yawning falsely detected as non-yawning:

$$FPR = \frac{FP}{TP + FN}; \tag{5.8}$$

4. the False Negative Rate (FNR) is the rate of non-yawning falsely detected as yawning:

$$FNR = \frac{FN}{FP + TN}; \tag{5.9}$$

5. and the Correct Detection Rate (CDR) is defined as:

$$CDR = \frac{TPR + TNR}{TPR + TNR + FPR + FNR}. \tag{5.10}$$

Table 5.6, exhibits a comparison on the YawDD dataset (ABTAHI et al., 2014) of the proposed method using data provided by AFTRM and AFTRM-W, with state of the art methods in yawning detection, including Chiang et al (CHIANG et al., 2003), Bouvier et al. (BOUVIER et al., 2008) and Omidyeganeh et al. (OMIDYEGANEH et al., 2016). It can be seen that the proposed method tends to outperform the comparative methods. It should be further noted that the proposed method has a higher TPR, which indicates the effectiveness of the proposed method. The threshold values for Γ_1, Γ_2 and Γ_3 are set to 1, 0.5 and 2.5, respectively.

Table 5.6: Yawning Detection Results (the best result are in bold).

Method	TPR	TNR	FPR	FNR	CDR
(CHIANG et al., 2003)	0.3990	0.4562	0.6010	0.5438	0.4276
(BOUVIER et al., 2008)	0.6764	0.5437	0.3236	0.4563	0.6101
(OMIDYEGANEH et al., 2016)	0.6578	**0.7733**	0.3419	**0.2266**	0.7155
MMDL-FT	0.7342	0.6435	0.2658	0.3565	0.6888
MMDL-FTU	0.7913	0.7432	0.2087	0.2568	0.7672
AFTRM	*0.8120*	0.7222	*0.1879*	0.2777	0.76703
AFTRM-W	**0.9307**	0.7551	**0.0693**	0.2449	**0.8429**

6 CONCLUDING REMARKS

This thesis proposes an adaptive face and facial landmark tracking scheme. The proposed face tracker contains two operating modes which selects the tracked target face among the candidate target face samples given by the motion model. Both the operating modes are based on feature learning techniques that utilize the useful data accumulated during the face tracking and implements an incremental learning framework to adapt to the current appearance of the tracked target face over time. To accumulate the training data, the quality of the test sample is checked before its utilization in the incremental and online training scheme. Also, a novel error prediction scheme is proposed that is capable of estimating the tracking error during the execution of the tracking algorithm.

The appearance model of the tracked target face is updated after a specific number of frames (batch size τ). The experimental results indicate that frequent updates (small batch size) on the appearance model of the proposed method has less tracking error than for large batch size. The reason for this behavior is that the proposed method updates the appearance of the tracked target face and also the resync (if required) of the features are performed after a specific number of frames τ. However, frequent updates in the appearance model or resyncing of features using CLM makes the method slower, and process less number of frames per second. To solve this problem, an optimal trade-off is estimated using a cost function to minimize the tracking error and number of updates/resyncs.

The first operating mode, namely MMDL-FTU, proposes a multi-model dictionary learning for face tracking, and the second operating mode, namely AFTRM-W, proposes a face tracking algorithm by combining an online learning technique with a model-based technique as a resyncing mechanism to improve the performance of the tracking when the tracking error is estimated to be high. Both the operating modes share the same motion model that models the motion and estimates the candidate target face samples, among which the tracked target face is selected based on the appearance model.

The proposed MMDL-FTU operating mode of the proposed method is a new scheme for tracking a target face, which is adaptive to the face appearance changes (e.g., changes in facial expressions and pose). MMDL-FTU operating mode relies on online learning of a multi-model dictionary, that helps to detect the target face against a complex background. Furthermore, the proposed method takes advantage of data collected during face tracking to learn the multi-model dictionary incrementally (i.e., to adaptively learn a dictionary for face detection/reconstruction and another dictionary for face/background

discrimination), updating the dictionary when the face appearance changes.

The AFTRM-W operating mode of the proposed method improves on a well known visual tracking approach based on incremental PCA using a re-syncing mechanism and is applied to face tracking. A novel error predictor is proposed that estimates the tracking error while face tracking. The high correlation of the proposed error predictor with the actual tracking error indicates its effectiveness. At times when the estimated tracking error is increasing, the performance is improved by using a resyncing mechanism that re-locates important features of the moving target (e.g., facial features in face tracking). This tracking scheme can be applied to non-rigid objects. In this work, it is applied to track faces; therefore, CLM/W-CLM is used as the resyncing procedure when divergence occurs.

Also, an improvement in the CLM is proposed, namely W-CLM. The W-CLM method utilizes the training data to assign weights to each landmark. The weights are computed using MMI that measures the consistency of the texture information around each landmark in the training set. These weights are used in the W-CLM search process to improve the localization of the landmarks.

Additionally, the tracked face is used to detect the facial landmarks which ultimately can be used for further facial analysis applications. These facial landmarks are used to detect yawning as a test case to validate the effectiveness of the proposed face tracker. It is also proposed an improvement in the yawning detection method, which utilizes the facial landmarks to estimate the features for yawning detection.

The proposed face tracker is evaluated using CLE of the tracked target face and RMSE of the facial landmarks with the comparative methods which are representative of the state of the art. The experimental results suggest that both the operating modes (MMDL-FTU and AFTRM) of the proposed face tracker can provide a competitive face tracking results in comparison to methods that are representative of the state-of-the-art. Also, the proposed improvement in yawning detection tends to increase the TPR and CDR of the yawning detection, which is evident from the experimental results.

In the future, we would like to extend the proposed method to another non-rigid object tracking. Similarly, the motion model can also be improved by drawing particles in the direction of movement of the object and modeling object specific characteristics while estimating the candidate samples instead of using a Gaussian distribution. The third possible improvement can be made in the representation of the object in a feature space that can provide more knowledge about local appearance changes.

6.1 Publications

- Mona Omidyeganeh, Shervin Shirmohammadi, Shabnam Abtahi, Aasim Khurshid, and Muhammad Farhan, Jacob Scharcanski, Behnoosh Hariri, Daniel Laroche, Luc Martel, "Yawning detection using embedded smart cameras". IEEE Transactions on Instrumentation and Measurement, IEEE, v. 65, n. 3, p. 570–582, 2016.

- A. Khurshid and J. Scharcanski, "Incremental multi-model dictionary learning for face tracking," 2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Houston, TX, USA, 2018, pp. 1-6. (KHURSHID; SCHARCANSKI, 2018).

6.2 Submitted Article

- Aasim khurshid, Jacob Scharcaski, "A New Adaptive Object Tracker with Applications", submission: Transaction on Instrumentation and Measurement.
  Submission date: 27-09-2018.

REFERENCES

ABTAHI, S. et al. Yawdd: a yawning detection dataset. In: ACM. **Proceedings of the 5th ACM Multimedia Systems Conference**. [S.l.], 2014. p. 24–28.

BABENKO, B.; YANG, M.-H.; BELONGIE, S. Visual tracking with online multiple instance learning. In: IEEE. **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**. [S.l.], 2009. p. 983–990.

BEHAINE, C.; SCHARCANSKI, J. Enhancing the performance of active shape models in face recognition applications. **IEEE Transactions on Instrumentation and Measurement**, IEEE, v. 61, n. 8, p. 2330–2333, 2012.

BLACK, M. J.; JEPSON, A. D. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. **International Journal of Computer Vision**, Springer, v. 26, n. 1, p. 63–84, 1998.

BLAKE, A.; ISARD, M. The condensation algorithm-conditional density propagation and applications to visual tracking. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 1997. p. 361–367.

BOUVIER, C. et al. Open or closed mouth state detection: static supervised classification based on log-polar signature. In: SPRINGER. **International Conference on Advanced Concepts for Intelligent Vision Systems**. [S.l.], 2008. p. 1093–1102.

BRADSKI, G. R. Real time face and object tracking as a component of a perceptual user interface. In: IEEE. **Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on**. [S.l.], 1998. p. 214–219.

CARNEIRO, G. et al. Detection and measurement of fetal anatomies from ultrasound images using a constrained probabilistic boosting tree. **IEEE Transactions on Medical Imaging**, v. 27, n. 9, p. 1342–1355, Sept 2008. ISSN 0278-0062.

CASCIA, M. L.; SCLAROFF, S. Fast, reliable head tracking under varying illumination. In: IEEE. **Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.** [S.l.], 1999. v. 1, p. 604–610.

CASCIA, M. L.; SCLAROFF, S.; ATHITSOS, V. Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 22, n. 4, p. 322–336, 2000.

CAUWENBERGHS, G.; POGGIO, T. Incremental and decremental support vector machine learning. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2001. p. 409–415.

CHEN, G.-Y.; TSAI, W.-H. An incremental-learning-by-navigation approach to vision-based autonomous land vehicle guidance in indoor environments using vertical line information and multiweighted generalized hough transform technique. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 28, n. 5, p. 740–748, 1998.

CHEN, S.; DONOHO, D.; SAUNDERS, M. **Atomic decomposition by basis pursuit: SIA M Review, 43, no. 1, 129–159**. 2001.

CHENG, X. et al. Visual tracking via sparse representation and online dictionary learning. In: SPRINGER. **International Workshop on Activity Monitoring by Multiple Distributed Sensing**. [S.l.], 2014. p. 87–103.

CHENG, Y. Mean shift, mode seeking, and clustering. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 17, n. 8, p. 790–799, 1995.

CHIANG, C.-C. et al. A novel method for detecting lips, eyes and faces in real time. **Real-time Imaging**, Elsevier, v. 9, n. 4, p. 277–287, 2003.

CHOI, I. H.; KIM, Y. G. Deep manifold embedding active shape model for pose invarient face tracking. In: **2018 IEEE International Conference on Big Data and Smart Computing (BigComp)**. [S.l.: s.n.], 2018. p. 578–581.

CHRYSOS, G. G. et al. A comprehensive performance evaluation of deformable face tracking "in-the-wild". **International Journal of Computer Vision**, Springer, p. 1–35, 2014.

COLLINS, R. T.; LIU, Y.; LEORDEANU, M. Online selection of discriminative tracking features. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 27, n. 10, p. 1631–1643, 2005.

COOTES, T. F. et al. Active appearance models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 23, n. 6, p. 681–685, 2001.

COOTES, T. F. et al. Active shape models-their training and application. **Computer Vision and Image Understanding**, Elsevier, v. 61, n. 1, p. 38–59, 1995.

COOTES, T. F.; TAYLOR, C. J. et al. **Statistical models of appearance for computer vision**. [S.l.]: Technical report, University of Manchester, 2004.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

CRISTINACCE, D.; COOTES, T. F. Feature detection and tracking with constrained local models. In: **BMVC**. [S.l.: s.n.], 2006. v. 1, n. 2, p. 929–938.

CRUYS, T. Van de. Two multivariate generalizations of pointwise mutual information. In: **Proceedings of the Workshop on Distributional Semantics and Compositionality**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (DiSCo '11), p. 16–20. ISBN 9781937284022.

DANELLJAN, M. et al. Learning spatially regularized correlation filters for visual tracking. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 4310–4318.

DONOHO, D. L.; ELAD, M. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 100, n. 5, p. 2197–2202, 2003.

ELAD, M.; AHARON, M. Image denoising via sparse and redundant representations over learned dictionaries. **IEEE Transactions on Image processing**, IEEE, v. 15, n. 12, p. 3736–3745, 2006.

ESSA, I. et al. Modeling, tracking and interactive animation of faces and heads//using input from video. In: IEEE. **Computer Animation'96. Proceedings**. [S.l.], 1996. p. 68–79.

ESSA, I. A.; PENTLAND, A. A vision system for observing and extracting facial action parameters. In: **CVPR**. [S.l.: s.n.], 1994. p. 76–83.

ESSA, I. A.; PENTLAND, A. P. Coding, analysis, interpretation, and recognition of facial expressions. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 19, n. 7, p. 757–763, 1997.

FANELLO, S. R. et al. Multi-class image classification-sparsity does it better. In: **VISAPP (1)**. [S.l.: s.n.], 2013. p. 800–807.

FLORES, E.; SCHARCANSKI, J. Segmentation of melanocytic skin lesions using feature learning and dictionaries. **Expert Systems with Applications**, Elsevier, v. 56, p. 300–309, 2016.

GERSHO, A.; GRAY, R. M. **Vector quantization and signal compression**. [S.l.]: Springer Science & Business Media, 2012.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition**. Washington, DC, USA: IEEE Computer Society, 2014. (CVPR '14), p. 580–587. ISBN 978-1-4799-5118-5.

GUO, H. A simple algorithm for fitting a gaussian function [dsp tips and tricks]. **IEEE Signal Processing Magazine**, v. 28, n. 5, p. 134–137, Sept 2011. ISSN 1053-5888.

JOSE, J. P.; POORNIMA, P.; KUMAR, K. M. A novel method for color face recognition using knn classifier. In: **2012 International Conference on Computing, Communication and Applications**. [S.l.: s.n.], 2012. p. 1–3. ISSN 2325-6001.

JUNG, C.; SCHARCANSKI, J. Wavelet transform approach to adaptive image denoising and enhancement. **Journal of Electronic Imaging**, v. 13, n. 2, p. 278–285, 2004.

KENDALL, D. G. A survey of the statistical theory of shape. **Statistical Science**, JSTOR, p. 87–99, 1989.

KHURSHID, A.; SCHARCANSKI, J. Incremental multi-model dictionary learning for face tracking. In: **2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)**. [S.l.: s.n.], 2018. p. 1–6.

KOELSTRA, S.; PANTIC, M.; PATRAS, I. A dynamic texture-based approach to recognition of facial actions and their temporal models. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 32, n. 11, p. 1940–1954, 2010.

KREUTZ-DELGADO, K. et al. Dictionary learning algorithms for sparse representation. **Neural computation**, MIT Press, v. 15, n. 2, p. 349–396, 2003.

KWON, H. J. et al. Multiple face tracking method in the wild using color histogram features. In: **2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)**. [S.l.: s.n.], 2017. p. 051–055.

LANITIS, A.; TAYLOR, C. J.; COOTES, T. F. A unified approach to coding and interpreting face images. In: IEEE. **Computer Vision, 1995. Proceedings., Fifth International Conference on**. [S.l.], 1995. p. 368–373.

LEVEY, A.; LINDENBAUM, M. Sequential karhunen-loeve basis extraction and its application to images. **IEEE Transactions on Image processing**, IEEE, v. 9, n. 8, p. 1371–1374, 2000.

LEWICKI, M. S.; SEJNOWSKI, T. J. Learning overcomplete representations. **Neural computation**, MIT Press, v. 12, n. 2, p. 337–365, 2000.

LIU, H.; LI, S.; FANG, L. Robust object tracking based on principal component analysis and local sparse representation. **IEEE Transactions on Instrumentation and Measurement**, IEEE, v. 64, n. 11, p. 2863–2875, 2015.

LOSING, V.; HAMMER, B.; WERSING, H. Incremental on-line learning: A review and comparison of state of the art algorithms. **Neurocomputing**, Elsevier, v. 275, p. 1261–1274, 2018.

LUCEY, S. et al. Efficient constrained local model fitting for non-rigid face alignment. **Image and Vision Computing**, Elsevier, v. 27, n. 12, p. 1804–1813, 2009.

MALCIU, M.; PRÊTEUX, F. A robust model-based approach for 3d head tracking in video sequences. In: IEEE. **Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on**. [S.l.], 2000. p. 169–174.

MALLAT, S. G.; ZHANG, Z. Matching pursuits with time-frequency dictionaries. **IEEE Transactions on signal processing**, IEEE, v. 41, n. 12, p. 3397–3415, 1993.

MICHAL, A.; MICHAEL, E.; ALFRED, B. K-svd: Design of dictionaries for sparse representation. **SPARS**, v. 5, p. 9–12, 2005.

OLSHAUSEN, B. A.; FIELD, D. J. Natural image statistics and efficient coding. **Network: computation in neural systems**, Taylor & Francis, v. 7, n. 2, p. 333–339, 1996.

OMIDYEGANEH, M. et al. Yawning detection using embedded smart cameras. **IEEE Transactions on Instrumentation and Measurement**, IEEE, v. 65, n. 3, p. 570–582, 2016.

PEYRÉ, G. Sparse modeling of textures. **Journal of Mathematical Imaging and Vision**, Springer, v. 34, n. 1, p. 17–31, 2009.

RANGANATHA, S.; GOWRAMMA, Y. P. An integrated robust approach for fast face tracking in noisy real-world videos with visual constraints. In: **2017 International Conference on Intelligent Computing and Control (I2C2)**. [S.l.: s.n.], 2017. p. 1–5.

REN, Z. et al. A face tracking framework based on convolutional neural networks and kalman filter. In: **2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)**. [S.l.: s.n.], 2017. p. 410–413.

ROSS, D. A. et al. Incremental learning for robust visual tracking. **International Journal of Computer Vision**, Springer, v. 77, n. 1-3, p. 125–141, 2008.

ROWEIS, S. Em algorithms for pca and spca. In: **in Advances in Neural Information Processing Systems**. [S.l.]: MIT Press, 1998. p. 626–632.

RUBINSTEIN, R.; ZIBULEVSKY, M.; ELAD, M. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. **Cs Technion**, v. 40, n. 8, p. 1–15, 2008.

RUECKERT, D. et al. Nonrigid registration using free-form deformations: application to breast mr images. **IEEE transactions on medical imaging**, IEEE, v. 18, n. 8, p. 712–721, 1999.

SAFFARI, A. et al. On-line random forests. In: **2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops**. [S.l.: s.n.], 2009. p. 1393–1400.

SALHI, A. et al. Face detection and tracking system with block-matching, meanshift and camshift algorithms and kalman filter. In: **2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)**. [S.l.: s.n.], 2017. p. 139–145.

SÁNCHEZ-LOZANO, E. et al. Cascaded continuous regression for real-time incremental face tracking. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2016. p. 645–661.

SANTNER, J. et al. Prost: Parallel robust online simple tracking. In: IEEE. **Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on**. [S.l.], 2010. p. 723–730.

SARAGIH, J. M.; LUCEY, S.; COHN, J. F. Face alignment through subspace constrained mean-shifts. In: **2009 IEEE 12th International Conference on Computer Vision**. [S.l.: s.n.], 2009. p. 1034–1041. ISSN 1550-5499.

SEE, Y. C. et al. Investigation of face recognition using gabor filter with random forest as learning framework. In: **TENCON 2017 - 2017 IEEE Region 10 Conference**. [S.l.: s.n.], 2017. p. 1153–1158.

SHIRMOHAMMADI, S.; FERRERO, A. Camera as the instrument: the rising trend of vision based measurement. **IEEE Instrumentation & Measurement Magazine**, IEEE, v. 17, n. 3, p. 41–47, 2014.

SNAPE, P. et al. Face flow. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2015. p. 2993–3001.

SOH, H.; DEMIRIS, Y. Incrementally learning objects by touch: Online discriminative and generative models for tactile-based recognition. **IEEE transactions on haptics**, IEEE, v. 7, n. 4, p. 512–525, 2014.

SOLDERA, J.; DODSON, K.; SCHARCANSKI, J. Face recognition based on geodesic distance approximations between multivariate normal distributions. In: **2017 IEEE International Conference on Imaging Systems and Techniques (IST)**. [S.l.: s.n.], 2017. p. 1–6.

SOLDERA, J. et al. Facial biometrics and applications. **IEEE Instrumentation Measurement Magazine**, v. 20, n. 2, p. 4–30, April 2017. ISSN 1094-6969.

TAHIR, N. M. et al. Feature selection for classification using decision tree. In: **2006 4th Student Conference on Research and Development**. [S.l.: s.n.], 2006. p. 99–102.

TERISSI, D.; GÓMEZ, J. C. Facial motion tracking and animation: An ica-based approach. In: **Proceedings of 15th European Signal Processing Conference, Poznan, Poland, September**. [S.l.: s.n.], 2007. p. 3–7.

TOIVANEN, M.; LAMPINEN, J. Incremental object matching and detection with bayesian methods and particle filters. **IET computer vision**, IET, v. 5, n. 4, p. 201–210, 2011.

VATER, S.; IVANCEVIC, R.; LEóN, F. P. Integration of precise iris localization into active appearance models for automatic initialization and robust deformable face tracking. In: **2017 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2017. p. 2617–2621.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. **Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on**. [S.l.], 2001. v. 1, p. I–511.

WANG, L. et al. Ik-svd: dictionary learning for spatial big data via incremental atom update. **Computing in Science & Engineering**, IEEE, v. 16, n. 4, p. 41–52, 2014.

WANG, L. et al. Comparative analysis of image classification methods for automatic diagnosis of ophthalmic images. v. 7, p. 41545, 01 2017.

WANG, Y.; LUCEY, S.; COHN, J. F. Enforcing convexity for improved alignment with constrained local models. In: **2008 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2008. p. 1–8. ISSN 1063-6919.

WENG, J.; EVANS, C. H.; HWANG, W.-S. An incremental learning method for face recognition under continuous video stream. In: IEEE. **Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on**. [S.l.], 2000. p. 251–256.

XIE, Y. et al. Discriminative object tracking via sparse representation and online dictionary learning. **IEEE Transactions on Cybernetics**, IEEE, v. 44, n. 4, p. 539–553, 2014.

YANG, F. et al. Dynamic texture recognition by aggregating spatial and temporal features via ensemble svms. **Neurocomputing**, v. 173, p. 1310 – 1321, 2016. ISSN 0925-2312.

YILMAZ, A.; JAVED, O.; SHAH, M. Object tracking: A survey. **ACM computing surveys (CSUR)**, ACM, v. 38, n. 4, p. 13, 2006.

YUAN, Y. et al. Visual object tracking based on appearance model selection. In: IEEE. **Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on**. [S.l.], 2013. p. 1–4.

ZHENG, S.; STURGESS, P.; TORR, P. Approximate structured output learning for constrained local models with application to real-time facial feature detection and tracking on low-power devices. In: IEEE. **Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on**. [S.l.], 2013. p. 1–8.