Daniel Chichorro de Carvalho

Development of a Facial Feature Detection and Tracking Framework for Robust Gaze Estimation

Dissertação de Mestrado em Engenharia Eletrotécnica e de Computadores
05/2018

· U C ·

Universidade de Coimbra

Development of a Facial Feature Detection and Tracking Framework
for Robust Gaze Estimation

Daniel Chichorro de Carvalho

Department of Electrical and Computer Engineering
Faculty of Sciences and Technology
University of Coimbra

Supervisor: Prof. Doctor Jorge Nuno de Almeida e Souda Almada Lobo[a,b]
Co-Supervisor: Prof. Doctor João Filipe de Castro Cardoso Ferreira[a,c]

Dissertation submitted for the Degree of Master of Science

[a]Institute of Systems and Robotics, Coimbra
[b]University of Coimbra
[c]School of Science and Technology
Nottingham Trent University

May 2018

Acknowledgements

There are a number of people who contributed in some form to the development of this thesis. First and foremost, I would like to acknowledge the help of my supervisors, Professors João Filipe Ferreira and Jorge Lobo, for their guidance and vision, as well as their patient mentoring. I would also like to extend my gratitude to my girlfriend, Joana, who always pushed me to do my best work, walking next to me when the road was bumpy and cheering me on with every small victory. To my parents, who allowed me to pursue my activities, and were always there to give advice and support. To my friends at the laboratory, who shared in my long nights of coding. To my squad of friends at the department, who I could always count on to share a laugh with at the Programming Club. To all my friends at TAUC, especially president Peres, for their understanding and support. To my fellow musicians at the Academic Orchestra, for providing a place for me to feel inspired. Lastly, to everyone that accompanied me through these past years as a student.

Abstract

Remote gaze estimation is the process of attempting to find the gaze direction, or ultimately, the point of fixation of a human subject, such as humans do to engage in communication. This is achieved by using information from the head pose and geometry of facial feature landmarks. This is useful for a number of applications, namely human-machine-interfacing. However, facial feature detection and tracking remains today one of the most important foci of Computer Vision, and no solution for a general-purpose facial feature detector and tracker exhibits satisfactory robustness under in-the-wild conditions. Most approaches towards this goal rely on high-resolution scenarios and overly near-frontal head poses, and no work was found accounting for both near-frontal and profile views during tracking. This work proposes a proof-of-concept that attempts to achieve better performance on distant subjects, and incorporating tracking of facial features in profile views. The designed method is based on a comparison of the shape geometry obtained from independent feature detectors to different shape models of the human face, capturing different head poses. Tracking is performed by the use of the KLT algorithm, and partial or full re-detections are triggered by a confidence-based design. A proof-of-concept is presented showing that the proposed solution satisfactorily addresses main functional requirements. Future work is suggested to further improve performance and address remaining issues.

Resumo

A estimação remota do olhar é o processo de tentar descobrir a direção do olhar de um humano, e ultimamente, o seu objeto de fixação, tal como os humanos o fazem para comunicar, partindo de informação de imagem capturada remotamente. Este processo é conseguido utilizando informação relativa à orientação da sua cabeça e à geometria dos pontos de referência faciais. Tem muitas aplicações, nomeadamente para Interface Homem-Máquina. No entanto, a deteção e seguimento de pontos faciais é um dos focos mais importantes de Visão por Computador, e nenhuma solução exibe uma robustez satisfatória em condições gerais. A maior parte das abordagens a este objetivo partem de cenários com alta resolução e poses de cabeça essencialmente frontais, e nenhuma abordagem encontrada contempla tanto caras frontais como de perfil. Este trabalho propõe um sistema que tenta alcançar uma melhor performance em pessoas mais distantes, e incorporando seguimento de pontos faciais em caras de perfil. O método desenvolvido baseia-se numa comparação da geometria obtida por detetores independentes de pontos faciais com diferentes modelos geométricos da cara humana, que capturam diferentes orientações da cabeça. O seguimento é feito pelo algoritmo KLT, e deteções parciais ou totais são lançadas por um design baseado em confianças. Uma demonstração conceptual é feita, mostrando que a solução proposta atinge de forma satisfatória os requisitos principais. Trabalho futuro é sugerido para melhorar a performance e lidar com as restantes questões.

Table of contents

List of figures

Chapter 1

Introduction

1.1 Motivation

Remote gaze estimation is a particular aspect of Computer Vision that gathers a lot of focus. It has many challenges associated, such as distance and occlusion, but is useful for many applications, namely psychophysical experimentation, video surveillance, and Human-Machine-Interfacing (HMI). It is also found to be crucial for communication engagement [18], which makes it a desirable functionality in artificial perception and cognition systems, such as socially-interactive robots.

Social robots are a prominent topic in Human-Robot Interaction (HRI), and they are expected to heavily hit the market in the coming years [25][23]. Particularly, it is expected that AI-driven conversational robots will soon take over the telephone information service [25]. However, verbal communication is only a subset of human social interaction, as humans use visual cues to express emotion and intent [22], and some studies show that nonverbal communication can often be more important than verbal content to the receptor of the message [14]. Capturing these subtle visual hints of information, such as gestures, body and head movements (e.g. nodding, waving) and eye gaze, and processing them requires a robust framework, integrating multiple layers of data acquisition and interpreting, from image processing to feature extraction, to shape interpretation, emotion classification, gaze estimation and speech recognition. The focus of this work is to tackle the first two steps, and develop a way to reliably track the facial feature geometry of a person in a video feed.

There are many different scenarios that hinder an accurate detection over all image frames in a video sequence. Among the most difficult scenarios are low resolution/quality images, distance of the person from the camera - which can translate to poor resolution, occlusion of parts of the face, head wear such as glasses, facial hair, and obviously, varying head poses and illumination changes, such as shadows or low ambient lighting [36, 37]. Additionally, most facial feature detection databases only account for the frontal or near-frontal view case, which renders fully automatic facial feature extraction non-existent for now [36].

The aim of this work is to serve as a follow-up to the project CASIR - Coordinated Attention for Social Interaction with Robots[1], improving the facial feature detection and adding a tracking module.

This project focused on researching artificial attention systems, in particular the coordination of multi-sensory stimuli and goal-directed attention, in the context of social interaction. In their designed architecture CASIR-IMPEP, they use an integrated framework combining multi-sensory information (auditory, 2D and 3D visual saliency and dynamics, memory) with appropriate weighting, to determine a gaze shift action to perform, similar to the process of *joint attention*, one of the primal human social interactions [17].

One of the areas developed in the project is robust gaze estimation. From physiological investigation done in this subject, it is known that the human brain uses the head pose to provide an estimate of gaze, particularly when the eyes are not visible due to distance or occlusions (e.g. sunglasses). However, when the eyes are visible, the head pose still presents a useful cue to improve gaze prediction. It is consensual that gaze estimation in humans results of a combination of the two subprocesses: head pose estimation, and eye gaze estimation [18]. The head pose estimation stage requires a specific geometry of facial features to construct a pose vector, according to the ones that are visible. This geometry is demonstrated in figure 1.1. The centre between each eye's corners (not the irises), as well as the nose tip and mouth corners, are used to compute the head pose of the individual. The eye gaze is computed by the positions of the irises relative to their eye's corners.

1.2 Objectives

The main goal of this project is to develop the proof-of-concept for a facial feature detection and tracking framework, robust in realistic conditions, capable of dealing with occlusion, varying pose, distance and resolution. The minimum requirement is to develop a solution that is an improvement from the off-the-shelf approach taken by the project CASIR for this purpose. Since their architecture is probabilistic, this work should provide confidence measures for the tracked points (see figure 1.2).

This framework is implemented in Matlab, and is supposed to be ported to OpenCV to be installed in the CASIR-IMPEP ROS architecture.

[1]Project web-page: http://mrl.isr.uc.pt/projects/casir.

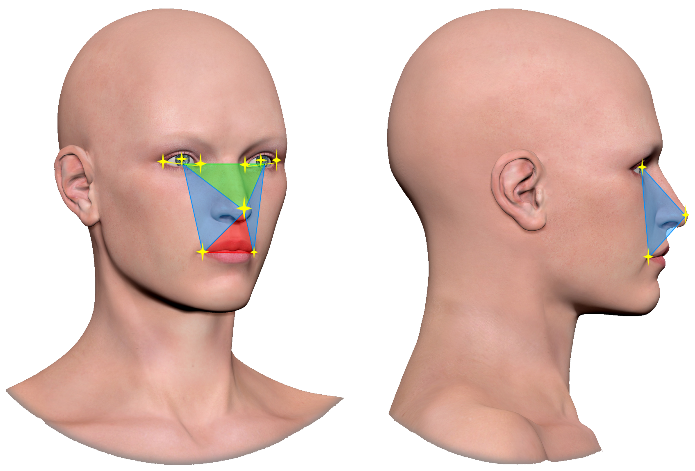


Fig. 1.1 The facial features required in the CASIR project. To compute the head pose, the eyes' position is represented by the centre of their corners, not precisely the iris. The iris is used for eye gaze estimation.

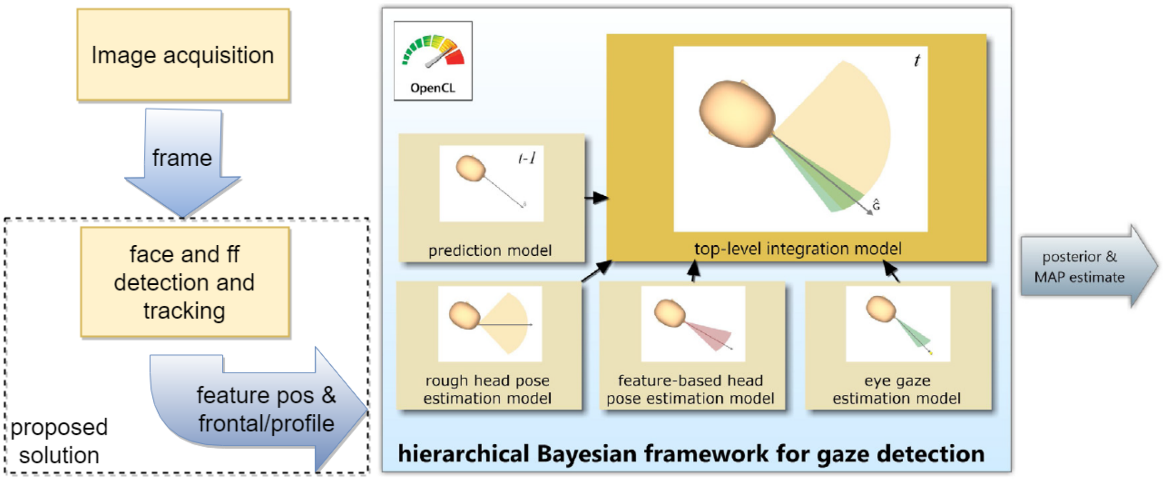


Image acquisition
frame
face and ff detection and tracking
feature pos & frontal/profile
proposed solution
OpenCL
t-1
prediction model
t
top-level integration model
rough head pose estimation model
feature-based head pose estimation model
eye gaze estimation model
hierarchical Bayesian framework for gaze detection
posterior & MAP estimate

Fig. 1.2 The proposed work integrated in the CASIR pipeline for gaze estimation (adapted from [18])

1.3 Related Work

The area of face and facial feature point detection gathers a lot of attention, with a number of surveys [36, 37, 39] available, from which to draw insight, as well as a number of facial feature localisation image databases to test with. However, the problem of facial feature tracking does not yet have the same focus, as the first facial landmark tracking benchmark database was presented in 2015 [27], and the first survey of this matter was released in the current year [9]. Although many applications concerning facial behaviour analysis require continuous long-term tracking of features, most work up until recently did not touch this issue [9].

From past to present, an outline of the general direction of techniques is presented in this section. Yang et. al. [38] proposed a set of techniques to be used in face and facial feature tracking. They used an adaptive skin colour model to track the face region, and image thresholding techniques to detect eye pupils, nostrils and mouth corners. For the detection of the eyes, they first use an iterative thresholding method on the upper half of the detected face, until two regions are found that correspond to the geometric prediction. Images of those areas are extracted, and those images are fed to a neural network trained to predict the gaze direction of the eyes, in degrees. Since the nose tip is a difficult feature to accurately detect using simple image processing algorithms, they used the nostrils of the individual, as well as the mouth corners, to estimate the head pose and, consequently, the gaze direction. The nostrils are much better defined regions in terms of image intensity, being easily captured by thresholding methods, although they are more prone to occlusion with varying head poses. The solution of [38] for tracking the features consists of updating the search windows and redetecting them using the same methods. This is not a robust solution, as difference in lighting or head pose can easily hinder a good detection, particularly in the case of eye feature detection [15].

Bourel et. al. [7] proposed a method for robust tracking of facial features, by making use of the Kanade-Lucas-Tomasi point tracker [20, 32], adapting it to their framework. In this case, they were focused on detection and tracking of three anchor points on each eyebrow, the nostrils, and mouth corners. The detection technique is based on similar image thresholding strategies, and anchored on the nostrils. From the nostrils, they narrow down the search windows by utilising the geometry of a frontal face, and perform detection on the smaller windows. Their pipeline [7] is illustrated in figure 1.3. When the tracker fails to track any feature point, the system will attempt to redetect it, and update the tracker before moving on to the next frame. This approach greatly increases robustness of the solution, and paves the way for more real-time applications, allowing for the inclusion of better detection strategies without hindering the real-time performance of the algorithm. However, since the face is

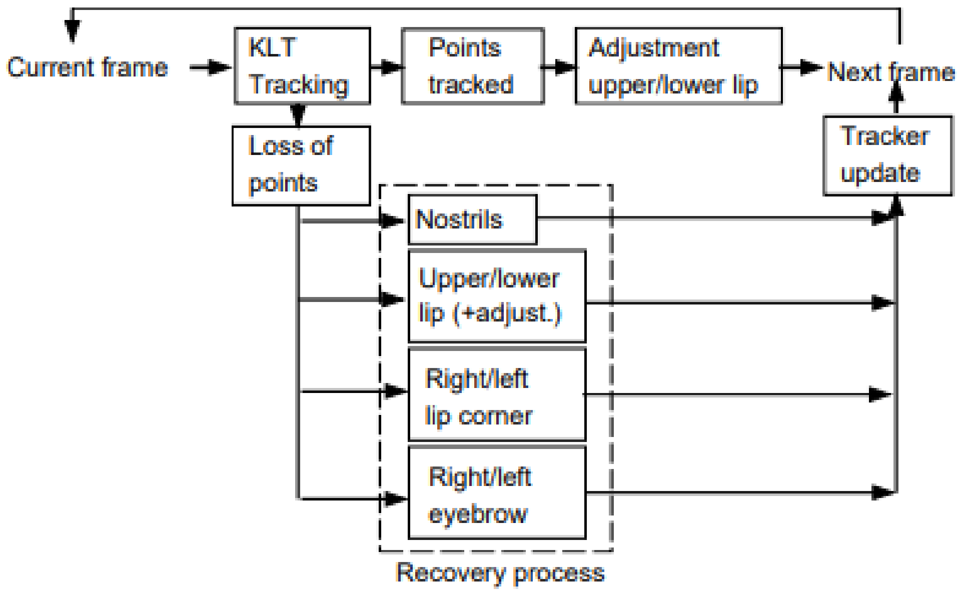


Fig. 1.3 The framework diagram for Bourel et al's solution for robust facial feature tracking (from [7])

assumed to be near-frontal, the use of this algorithm "in-the-wild", where very dynamic head poses and situations occur, is not sufficient for a robust detection and tracking of facial features.

Meanwhile, the focus started turning into analysing the geometry of the detected points, and using techniques that search for the shape description of the face, rather than each feature independently. Active Shape Models (ASM) [11] and Active Appearance Models (AAM) [10] were notorious techniques for their widespread use in facial feature tracking, but their shortcoming is requiring a robust initialization to be able to perform successfully. However, as a measure of confidence in the detected points, statistical models of shape played a huge role [12][33][24]. These approaches allow for more robustness to varying head poses, as they integrate the change in geometry as the head pose varies into the detection, pulling in outlier points, producing more cohesive results. In this sense, the use of Point Distribution Models (PDM) becomes a standard technique.

However, there are a number of situations that add a great deal of difficulty to the problem of reliably detecting and tracking faces in unconstrained scenarios. Most notable of them is the issue of extreme pose and partial occlusion of the face [9, 36]. This poses a more complex problem, requiring more intelligent solutions.

The need to tolerate missing features, and predict their position based on the present configuration leads to strategies like Kalman filtering, as attempted by [34]. A lot of machine

Method	Distance	Pose	Occlusion
[38, 31]	Low	Frontal	No strategy
[7]	Low	Near-frontal	No strategy
[33, 24]	Medium	Half-profile	Model estimation
[34]	Medium	Half-profile	Kalman filtering

Table 1.1 Capabilities of the described methods, in terms of robustness to distance, variety of head poses, and occlusion of features.

learning-intensive strategies are now the norm, with many international competitions pushing forward the state-of-the-art [27, 26].

The solution for facial feature detection utilised in CASIR builds upon Haar classifiers [18]. First, a set of classifiers trained for different head pose configurations are applied simultaneously, and according to the result, different Haar detectors are launched inside reasonable ROIs. The eye ROIs returned are then segmented into their features, namely the corners and iris. The corners are considered to be approximately in the vertical lines of the Haar detection, at centre height. The iris is segmented as the centre of the ROI. The nose tip and mouth corners are detected through similar methods, explained in sections 2.2.3 and 2.2.4. What CASIR's solution lacked is a way to evaluate the extracted features' positions, by comparing them to a model of some sort.

In the current times, most methods for facial feature extraction and tracking rely on Deep Learning techniques, such as Convolutional Neural Networks (CNNs) [9, 37], which have been progressively dominant in many different areas of Computer Vision.

Since facial feature point detection has been a popular topic in Computer Vision, a comprehensive evaluation of all the methods available would be time-consuming, and outside of the spectrum of this work. However, from the surveys analysed, important insight was drawn. Table 1.1 compares the above described methods for facial feature tracking, in terms of their applicability to specific in-the-wild scenarios.

1.4 Proposed Work

The main contribution of this work to project CASIR is a significant upgrade to their facial feature detection approach, mainly by providing a way to assess the coherence of the detected features, adding some robustness to the detection. Also, to improve the time performance of the system, feature tracking is to be implemented.

In fact, as can be seen from table 1.1, none of the surveyed methods explicitely deal with full profile views, nor properly handle large distances to camera (i.e. over 2 metres). This is reflected in the fact that this use case is not captured in in popular databases such as the 300VW database [2][9]. One key contribution of the proposed method is the integration of profile view feature detection and tracking.

1.5 Structure of Dissertation

This dissertation is divided in four chapters: Introduction, Materials and Methods, Implementation and Results, and Conclusions and Future Work.

In the second chapter, the foundation for the developed work is presented, going from all stages of the pipeline, starting on face detection methodologies, reviewing some methods for individual feature extraction, analysing the theory behind statistical shape analysis and exploring visual tracking techniques.

In the third chapter, the development of the present work is described, and the results are presented. This section goes from the choice of environment work in to the testing databases utilised, going through the intricacies of each step in the framework pipeline.

In the fourth chapter, some conclusions of this work are given, and some possibilities of future work are discussed.

[2]See more details on available databases and corresponding datasets in section 3.4.1

Chapter 2

Materials and Methods

2.1 Face Detection

Face detection in Computer Vision and image processing is the process of retrieving a region from an image that contains a human face. This is useful for countless applications, including photography, surveillance, human recognition, and is the basis for many different areas of HMI. The most popular method for face detection is commonly referred to as the Viola-Jones detector, proposed in [35]. It has been regarded as the 'industry-standard' technique for many years, with few upgrades since. It combines the use of integral images - a preprocessing of frames to accelerate calculation of sums over rectangular areas - with Haar-like features, and using the AdaBoost algorithm to train a cascade of classifiers, each classifier having a larger rejection rate than the previous one.

Haar-like features are sets of adjacent rectangles that provide a way to encode information in an area, as opposed to performing calculations with every individual pixel in that area. This way, processing is far quicker, allowing for real-time computations. The features used in this method are described in figure 2.2, and are comprised of three types: *two-rectangle features*, which consist on the difference between the sum of pixels in two adjacent rectangle regions, *three-rectangle features*, which consist on the difference between the sum of pixels in a centre rectangle and the combined sum of pixels in the adjacent rectangles. *Four-rectangle features* consist on the difference between sums of pixels in diagonally adjacent rectangles [35].

The integral image is computed as follows:

$$I'(x,y) = \sum_{x'<x, y'<y} I(x',y')$$

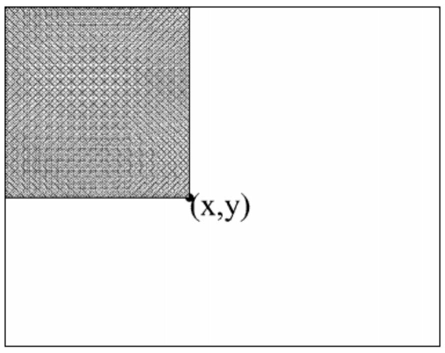


Fig. 2.1 Each pixel in the integral image corresponds to the sum of every pixel above and to the left of the pixel in the original image (adapted from [35]).

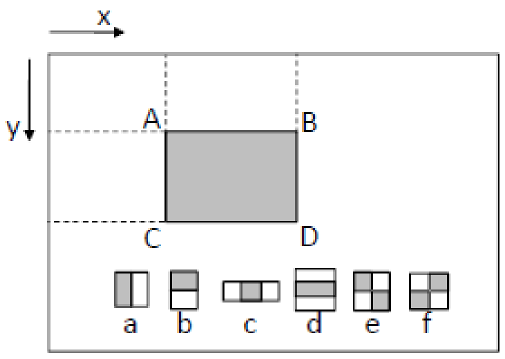


Fig. 2.2 Calculation of the sum of a rectangle using the integral image, and examples of Haar-like features considered. The features consist on the difference between the summed pixel intensity in the grey area and the white area (adapted from [39]).

where I' is the integral image and I the original image.

Through the use of this preprocessing stage, the sum of the pixels inside a rectangle ABCD in the original image can be calculated very quickly, using only four array references [39]:

$$\sum_{(x,y)\in ABCD} I(x,y) = I'(D) + I'(A) - I'(B) - I'(C)$$

A set of classifiers using the Haar-like features are trained by a boosting algorithm, namely AdaBoost (Adaptive Boosting). Boosting is a strategy for combining weak learners into a stronger learner.

A weak learner or classifier is a predictor that only has to guess slightly better than random guessing, i.e. its correct classification rate just has to be over 50% [21] (for binary classification). Through boosting, a strong learner is obtained by a linear combination of weak learners [21]. An example of a weak learner could be a decision stump, trained to guess if there is a face in an area simply based on the value of a feature like the ones in figure 2.2 being higher or lower than a threshold. Since only one feature wouldn't be able to capture all possible characteristics of a face, the classifier obviously wouldn't be very accurate. However, it could perform just slightly better than random, and being a very simple classifier, it can be weighted based on its prediction success rate and integrated into a combination of other similar decision stumps, leading to an overall stronger classifier.

In Viola-Jones, boosting is used to greatly reduce the number of features to consider, since otherwise there would be too many to draw information from, impeding a real-time implementation [35]. Boosting selects the features with greater predicting power, leading to better and faster classifiers. The trained classifiers are then cascaded in an attentional structure, with the classifiers with the highest rejection percentage on top of the cascade. The rationale behind this method is that the negative sub-windows (those not containing a face) will be rejected early on in the process, making the algorithm more efficient.

2.2 Facial Feature Detection

The facial features that I am exploring in this work are the ones that are required in the CASIR-IMPEP architecture, which are used for eye gaze and head pose estimation. These are the eye pupils and corners (also known as *cantus*), the nose tip, and the mouth corners. Following upon the work in [38] and [7], I attempt to track the nostrils as well, as they provide ancillary information regarding the geometry of the face, and are, in some cases, easier features to track.

2.2.1 Eye Pupils

A survey on eye detection techniques [15] reveals there are three categories: *shape-based*, *appearance-based* and *hybrid* methods. Shape methods are usually constructed from local feature points or contours, and use a prior model of eye shape, as opposed to appearance-based methods, that use the appearance directly. Hybrid methods use a combination of the two approaches.

There are also many different kinds of methods for pupil detection, but mostly unfit for the purpose of this work, as they require an active system to perform detection (e.g. an infrared camera, a head mounted system, or specific illumination setups to induce a glint in the eyes that is easy to capture) [15].

In [38], simple darkest pixel finding on two regions extracted by iterative thresholding on the upper region of a skin-color model segmentation.

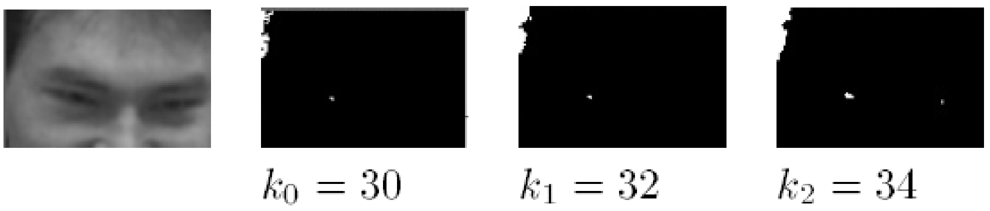


Fig. 2.3 Iterative thresholding used in [38] to detect the eye regions (from [38]).

A popular technique for detecting the pupil is through the use of projection functions [40, 5], most commonly the Integral Projection Function (IPF). In [40], both the IPF and the Variance Projection Function (VPF) are combined into the Generalized Projection Function, which exhibits a better performance than the former techniques. However, relatively high contrast images and a minimum resolution are required for these methods to work adequately. In the next chapter, the solution that was chosen is explained.

2.2.2 Eye Corners

In [6], a number of eye corner detection techniques are analysed, and the best-performing uses a neural network to generate an eye window from the face window, and inside it the eye corners are extracted using a method relying on the Harris corner detector [16]. After the pupil is detected, the eye region is subdivided, providing two smaller search windows on each side, on which the Harris corner detection method is run in the original subwindow, and a Gaussian smoothed version of it. The corner candidates are filtered based on their detection on both images.

The chosen approach in this work also makes use of the Harris corner detector, but uses a thresholding method in the subwindow, using an adaptive threshold (Bradley's method [8]), selecting the leftmost or rightmost candidate, depending on the eye corner.

2.2.3 Nose Tip

The nose tip is a tricky feature to detect by using 2D local detectors. Most methods for nose tip detection are based on more holistic techniques, applying a shape matching, usually with 3D information, for example in the method utilized in CASIR's architecture [21]. Also, the nose tip is the most pose-dependent feature, varying significantly with the yaw angle of the face.

To tackle this, two approaches were taken to detect the nose tip, one in the frontal view, and another in the profile view. To detect the nose tip in the frontal view, a Haar cascade detector, similar to the Viola-Jones method, is applied to the face bounding box, returning a smaller window containing the nose, and most frequently centred on its tip. Since the results of using such a simple detection technique are satisfactory, no further processing is done.

In the profile case, nose tip detection is performed by the use of Active Contours, also called 'snakes', as well as a skin-colour segmentation of the face region. The contour mask is initialized in the centre of the face bounding box, to capture the skin tone of the user. The contour searching is run for 200 iterations, and the outermost point in the contour is chosen as the nose tip.

In CASIR, the nose tip is detected through the method by Macesanu et. al. [21], which utilises the initial face ROI to compute a smaller search window, roughly a third of the face bounding box's width and height. Then, the resulting area is converted to the Lab colour-space, contrast-adjusted, and a logical AND is performed between the *b* colour-channel with a morphological gradient filter applied and the unaltered *a* colour-channel. From the resulting image, k-means clustering is used to segment the nose region, and the nose tip is extracted.

2.2.4 Mouth Corners

Most surveyed approaches for mouth corner detection rely on image processing techniques, such as edge filtering [38], intensity thresholding [7] or colour difference [29].

The approach taken by the CASIR team was based on the method in [29]. It consists of using skin colour segmentation in the *Lab* colour-space (*L* consists of luminance, and *a* and *b* are the colour channels), combining two techniques to segment the face image, nearest-neighbour colour segmentation, and colour-based k-means clustering. Then, the lips are found by fitting an ellipse to the resulting clusters. After that, corner detection in the ellipse's extrema provides the location of the mouth corners.

Bourel et. al's [7] solution for mouth corner detection is fairly simple, consisting of a search for dark zones in windows defined relative to the position of the nostrils (the nostrils serve as anchor points).

The solution applied in [38] uses an integral projection to detect the line separating both lips, and then applying a horizontal edge filter to capture the lip zones, and selecting the extrema of those regions as the mouth corners. This method is too specific, as it assumes a closed mouth configuration, as well as a frontal head pose.

2.3 Shape Model

Through the use of independent feature detectors, even if they are run on small search windows of the face image, it is common to lose the shape geometry the features form. It is then necessary to use a method for evaluating the plausibility of the present configuration, to give the framework some insight on the position of the points. This can be achieved by comparing the tracked points to a model that describes the face geometry. This approach is very popular, first gaining tract with the proposal of Active Shape Models by Cootes et. al. [11], but quickly developing to Active Appearance Models [10], which is one of the most featured methods for facial feature tracking. Many other approaches use strategies deriving from ASMs and AAMs [24, 13, 33, 19].

This model can be constructed from a number of training images with annotations of the desired points. The points from each image are concatenated together, normalised to the same centre and scale.

2.3.1 Procrustes Analysis

An important topic in Computer Vision is registration. It consists of conforming two or more coordinate systems to a common system, by estimating the required geometric transformation parameters. In this case, the input face shape is to be fitted to a model of the face shape, to determine if the tracked points present a valid configuration of a face. A method that achieves this is Procrustes Analysis [3]. Procrustes was a burglar in ancient Greece who offered hospitality to passing travelers in a miraculous bed that could fit any guest. However, to make the guests fit the bed, he would either stretch them or cut off their limbs. Albeit a bit obscure, this is a good analogy as to what the least-squares minimization tool achieves.

Particularly, the Orthogonal Procrustes problem is the transformation of matrix $\mathbf{A}$ to fit $\mathbf{B}$ minimizing the residual $\mathbf{E} = \mathbf{AT} - \mathbf{B}$, where $\mathbf{T}$ is an orthogonal rotation matrix. This problem does not contemplate a translation between the two matrices, and it also does not address the scale difference expected in our application, therefore it is not enough to suit our needs.

A more useful tool would be the Extended Orthogonal Procrustes minimization [3], which takes into account a translation vector $\mathbf{t}$ and a scale factor $\mathbf{s}$. The method is described as the least-squares minimization of the residual matrix:

$$\mathbf{E} = \mathbf{cAT} + \mathbf{jt}^T - \mathbf{B} \tag{2.1}$$

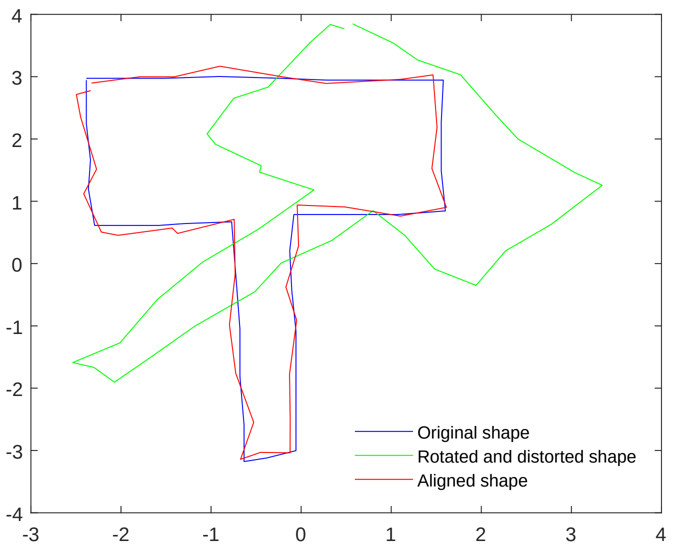


Fig. 2.4 An example of the Orthogonal Procrustes problem. The first shape is rotated, random noise is applied to it, and it is then aligned to the original shape using the OP method.

where both $\mathbf{A}$ and $\mathbf{B}$ are $p \times k$, $\mathbf{j}$ is a $p \times 1$ unit vector, $\mathbf{t}$ is a $k \times 1$ translation vector, and c is a scalar value. Obviously, p represents the number of landmarks in the shape, and k the number of dimensions (in 2D, $k = 2$).

However, this method assumes all the points have the same confidence associated to them, and tries to align them without prioritising. Since from the current work it is shown that the different feature detectors have different accuracies, it is a good idea to apply weights to the set of points, prior to attempting to align them. This can be described by the **Weighted Extended Orthogonal Procrustes** (WEOP) problem [3], modelled by the following equation:

$$\mathbf{E} = \mathbf{W}(c\mathbf{AT} + \mathbf{jt}^T - \mathbf{B}) \tag{2.2}$$

where $\mathbf{W}$ is a diagonal matrix of weights, with dimension $p \times p$. These weights can be any number, as their scale doesn't influence the end goal of minimizing the residual $\mathbf{E}$. This method prioritises the fitting of the points with the highest weights, as can be illustrated in figure 2.6.

The sum of squares in a matrix X can be represented by $tr\{X^T X\}$, with the tr{} operator representing the trace of the matrix. Therefore, equation 2.2 can be represented as:

$$minimize \quad tr\{(c\mathbf{AT} + \mathbf{jt}^T - \mathbf{B})^T \mathbf{W}^T \mathbf{W}(c\mathbf{AT} + \mathbf{jt}^T - \mathbf{B})\} \tag{2.3}$$

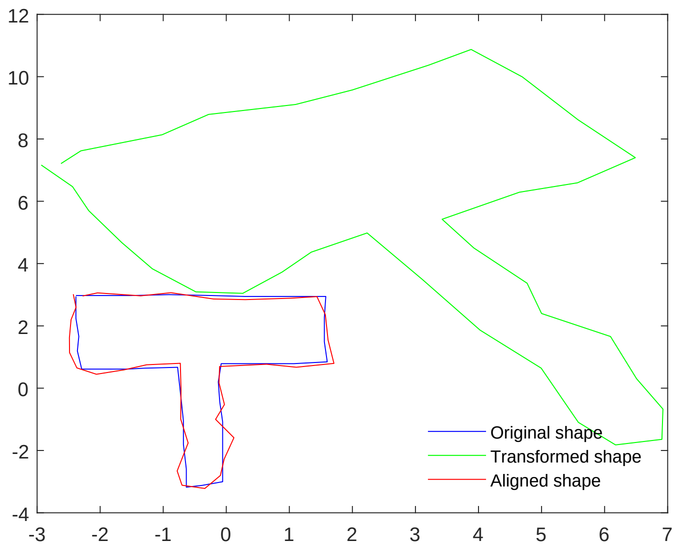


Fig. 2.5 Similar example, this time performing also a scaling and translation, before being realigned to the original shape using the EOP method.

where $\mathbf{W}^T = \mathbf{W}$, as $\mathbf{W}$ is a diagonal matrix. Since $\mathbf{T}$ is an orthogonal transformation matrix, the orthogonality condition must be met:

$$\mathbf{T}^T\mathbf{T} = \mathbf{T}\mathbf{T}^T = \mathbf{I} \tag{2.4}$$

Through the rationale in [3], a Lagrangean function with these two conditions can be defined to lead to the parameters $\mathbf{T}$, c and t, by setting its derivative to zero, and finding the parameters through the use of Singular Value Decomposition. By defining:

$$\mathbf{A}_w = \mathbf{W}\mathbf{A},\ \mathbf{B}_w = \mathbf{W}\mathbf{B},\ \text{and}\ \mathbf{j}_w = \mathbf{W}\mathbf{j}$$

Equation 2.3 can be rewritten as:

$$minimize \quad tr\{(c\mathbf{A}_w\mathbf{T} + \mathbf{j}_w\mathbf{t}^T - \mathbf{B}_w)^T(c\mathbf{A}_w\mathbf{T} + \mathbf{j}_w\mathbf{t}^T - \mathbf{B}_w)\} \tag{2.5}$$

Through the following SVD, the unknown parameters can be found [3]:

$$\text{svd}\{\mathbf{A}_w^T(\mathbf{I} - \frac{\mathbf{j}_w\mathbf{j}_w^T}{\mathbf{j}_w^T\mathbf{j}_w})\mathbf{B}_w^T\} = \mathbf{U}\Sigma\mathbf{V}^T \tag{2.6}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal eigenvector matrices, and Σ is a diagonal eigenvalue matrix. Finally:

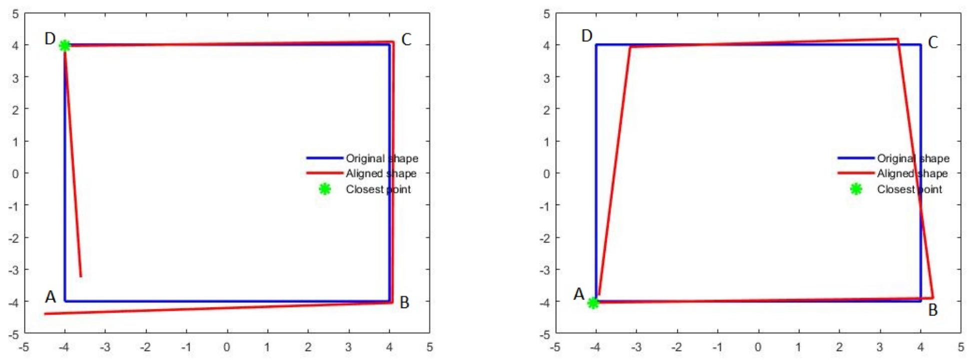


Fig. 2.6 The effect of different weighting of the shape points. In the left, the weights were the following, in ABCDA' order: [0.1, 0.1, 0.1, 1, 0.1]. In the right, the weights were [1 0.1 0.1 0.1 0.1]. As can be seen, the algorithm prioritises the minimization of the distance of the points with the highest weight.

$$\mathbf{T} = \mathbf{U}\mathbf{V}^T \tag{2.7}$$

$$c = \frac{\mathrm{tr}\{\mathbf{T}^T\mathbf{A}_w^T(\mathbf{I} - \frac{\mathbf{j}_w\mathbf{j}_w^T}{\mathbf{j}_w^T\mathbf{j}_w})\mathbf{B}_w\}}{\mathrm{tr}\{\mathbf{A}_w^T(\mathbf{I} - \frac{\mathbf{j}_w\mathbf{j}_w^T}{\mathbf{j}_w^T\mathbf{j}_w})\mathbf{A}_w\}} \tag{2.8}$$

$$\mathbf{t} = (\mathbf{B}_w - c\mathbf{A}_w\mathbf{T})^T\frac{\mathbf{j}_w}{\mathbf{j}_w^T\mathbf{j}_w} \tag{2.9}$$

Having the transformation parameters calculated, all that's left is applying the transformation to the matrix **A** to complete the alignment process.

2.3.2 Principal Component Analysis

If a dataset consisting of annotations of facial features from training images is considered, the number of variables is the dimensionality of the images (2, since only 2D information is being used) times the number of landmarks per image. For demonstration purposes, 10 landmarks are assumed. That means the data comprising the dataset has 20 dimensions. However, since the data isn't random, there must be interdependencies between the landmarks, e.g. the eye pupils are always between their respective eye corners. Therefore, a dimensionality reduction must be found, so that the differences in the shapes in the dataset can be visualized and analysed clearly.

Principal Component Analysis (PCA) [30] is a statistical tool that is applied in many different fields to reduce dimensionality, capturing the axes that hold the most variation in a dataset. It can be thought of as fitting an n-dimensional ellipsoid to the data, in which the largest axis of the ellipsoid is the one in which there is more variance in the data.

To perform PCA on a dataset, first the data must be normalised, i.e., subjected to a common referential. Simply, to centre the dataset at the origin, all it takes is to subtract the mean from itself. After that, to impose a similar scale to all the observations, Procrustes Analysis can be used, as explained in the previous section. Normalising the scale isn't necessary to perform PCA, but it can be expected that the first principal component obtained, the one that describes the most variance in the dataset, will capture scale, and this information is irrelevant for the application of this work.

Considering a data matrix **X**, which is $m \times n$, where m correspond to observations, and n the number of variables. Assuming **X** has already been centred at the origin and the scale of observations was already normalised. The first step is to compute the covariance matrix of **X**.

The covariance matrix **C** is calculated as follows:

$$\mathbf{C} = \frac{1}{m-1}\mathbf{X}^T\mathbf{X}$$

Since the matrix $\mathbf{X}^T\mathbf{X}$ is symmetric, we can perform Singular Value Decomposition on it, to extract its eigenvectors and eigenvalues. In fact:

$$\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}^T$$

Where Σ is a diagonal matrix of eigenvalues, and **U** and **V** are orthonormal eigenvector matrices. The eigenvectors of the covariance matrix **C** are the principal components of the dataset **X**, and are in descending order of described variance. To visualize the effect of the principal components, a set of vectors can be constructed from the mean of the dataset, as follows:

$$\mathbf{x}_i = \hat{\mathbf{x}} + k\mathbf{v}_i$$

where $\mathbf{x}_i$ is a vector similar to an observation in **X**, $\hat{\mathbf{x}}$ is the column mean of **X**, k is an arbitrary scalar, and $\mathbf{v}_i$ is the i-th eigenvector of **C**. The value of k can be arbitrary, but for good visualization its values should lie between $-\lambda_i$ and λ_i, with λ_i being the eigenvalue corresponding to the eigenvector $\mathbf{v}_i$. In section 3.2.3, some examples of this visualization are shown.

2.4 Tracking methods

2.4.1 Kanade-Lucas-Tomasi (KLT) Algorithm

The Kanade-Lucas-Tomasi tracking algorithm was based on the work by Lucas and Kanade [20] on image registration, developed by Tomasi and Kanade [32], and later Shi and Tomasi improved the feature selection method, and introduced a dissimilarity measure to the tracker [28].

An image sequence can be described as the function of three variables $I(x,y,t)$, where the x and y variables are discrete and bounded. If the time intervals are short enough, we can safely assume that the image $I(x,y,t)$ and $I(x,y,t+\tau)$ are highly correlated, as they describe the same scene from slightly different viewpoints. This allows for the assumption of pattern movement in the image:

$$I(x,y,t+\tau) = I(x-\xi, y-\eta, t)$$

which basically means that an image taken shortly after can be obtained by moving every point in the image by a suitable transformation $\mathbf{d} = (\xi, \eta)$, called the *displacement* of point $\mathbf{x} = (x,y)$ between instants t and $t+\tau$. This displacement can be usually a function of x,y and t [32]. However, the function that describes displacement might not be well defined, as all images suffer from noise. Therefore, if we define $I(x,y,t+\tau)$ as $J(\mathbf{x})$, and $I(\mathbf{x}-\mathbf{d}) = I(x-\xi, y-\eta, t)$, and dropping the time variable, then the image model is:

$$J(\mathbf{x}) = I(\mathbf{x}-\mathbf{d}) + n(\mathbf{x})$$

where n represents the noise.

The goal of feature tracking is to find this displacement vector, in a way which minimizes the dissimilarity of the window between two consecutive frames. This can be done by minimizing the residual:

$$\varepsilon = \int_W [I(\mathbf{x}-\mathbf{d}) - J(\mathbf{x})]^2 w d\mathbf{x}$$

where W is a window around the point $\mathbf{x} = (x,y)$, and w is a weighting function of the pixels inside that window, which can be set to 1, in the simplest case, but can also, for example, be a Gaussian function, giving more importance to the centre of the window W.

This minimization of the residual ε can be done in many ways, but for a small displacement vector (smaller than window W) the residual), the algorithm uses the method in [20].

Chapter 3

Implementation and Results

The general strategy of this work is to build a system similar to [7], but adding several crucial functionalities:

- **Profile feature detection:** The system must be able to reliably track features in profile views, as its main use case scenario is when the user is not looking directly at the camera. Since varying pose-tolerant approaches don't fully account for the full profile case, a specially designed module for profile feature detection must be used.;
- **Shape model verification:** Since the KLT tracking algorithm can lose track of the points in random ways, due to occlusion, poor illumination, or rapid movements provoking a motion blur in the image, the shape of the face can get deteriorated, up to a point where the system has no confidence in the configuration of the tracked points, triggering a re-detection. To have this awareness, the system must include some prior info about a mean face shape, to be able to compare with, and assess the difference.;
- **Confidence-based re-detection:** The KLT algorithm attributes a confidence measure to each point tracked. However, since this measure is based on the similarity between the previous and current neighbourhoods, if points stray from the shape at a slow pace, it may come to a situation where the points are absurdly far from each other, while maintaining a good confidence measure in the KLT algorithm. Analogous to the previous topic, the system must be able to determine when a single feature is differing from the expected shape configuration. Therefore, each feature point must have its own confidence measure, and through that measure, the system may trigger partial re-detections.

3.1 Implementation Infrastructure

The selected environment for the development of this work was MATLAB, making use of the Computer Vision and Image Processing toolboxes. The framework was tested using a laptop running Windows 10, with an Intel Core i5-5200U CPU running at 2.20GHz and an Nvidia GeForce 920M graphics card.

To test the framework, an application with a graphical user interface (GUI) was created, using a webcam as a live video feed. A version of the framework was also created to work with video files instead of the video feed, but without the user interface.

To move from Matlab to the CASIR-IMPEP architecture, the code should be ported to C/C++, using OpenCV. Then, it can be integrated in the ROS framework, and installed in the architecture.

3.2 Conceptual Design

The main conceptual diagram for the proposed work is shown in figure 3.1. The algorithm can be summarised in three steps: Face detection, Facial feature detection, and feature tracking. In the last phase, the system will attempt to figure out when another re-detection is needed for a specific feature, or if the overall confidence has dropped too low, a full re-detection can be triggered. This hybrid detection/tracking pipeline gives the algorithm robustness, being able to reinitialize itself completely if the points have been lost.

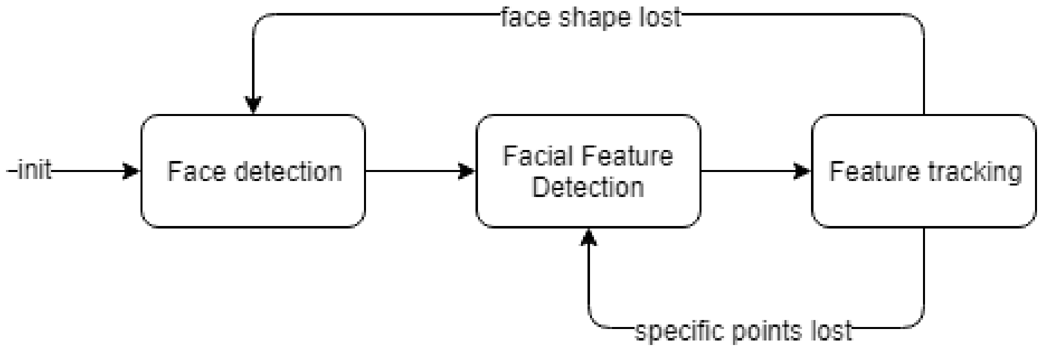


Fig. 3.1 High-level diagram for the proposed work.

3.2.1 Face Detection phase

In the first step, the goal is to obtain a bounding box containing the face to be tracked. To achieve this, the Viola-Jones [35] algorithm is applied, using the implementation in the

Computer Vision toolbox for Matlab, the *Cascade Object Detector* (COD) system object, using a 'FrontalFaceLBP' classification model.

This algorithm returns a set of bounding boxes corresponding to possible faces. Since it is not full-proof, sometimes a face might not be correctly detected, and some false positives can be returned. To sort through the bounding boxes, a separate COD is applied, using the 'EyePairSmall' classification model. The largest bounding box in which the second COD detects an eye pair is chosen as the face bounding box to feed to the next step in the algorithm. If the frontal face COD doesn't return any bounding boxes, a separate profile face COD is used to attempt to find profile faces. Since this COD was trained using the same view, if no faces are found, the image is flipped horizontally and the COD is relaunched.

If there was a previous detection when the algorithm is run, i.e. if the tracker was already running, and a full re-detection was triggered, the system selects a search window around the last face detection, and only if nothing is found in that window, does the algorithm revert to use the whole frame as input to the COD. This is to help prevent sudden changes in the tracked face, e.g. if a closer person comes in front of the camera, the system doesn't just switch to the new person because its face bounding box is larger. This approach is more sensible in a HMI perspective: the system will only switch subjects if the current one becomes invisible.

The procedure for obtaining a face bounding box is described in figure 3.2.

3.2.2 Facial feature detection

The total number of features to track is 9, corresponding to both eyes' corners and pupils, the nose tip, and mouth corners. As ancillary points, the nostrils were considered, as, appearance-wise, they are relatively well defined, and so can be easily detected, providing similar info to the nose tip. First I will consider the frontal face case.

Eye features

Eye features are comprised of eye corners and eye pupils. The first step to their detection is to locate the region containing these features. For that purpose, the frontal face bounding box is divided into four quadrants, and two CODs with classification models 'LeftEye' and 'RightEye' are launched in the corresponding upper quadrants of the face. From the returned bounding boxes, the feature detectors are run.

The eye pupil detector utilised comes from a solution by Asadifard and Shanbezadeh [4], as their detector was method showed good results in obtaining the position of the pupil, both in high and low resolution images. The only difference from my implementation is the use

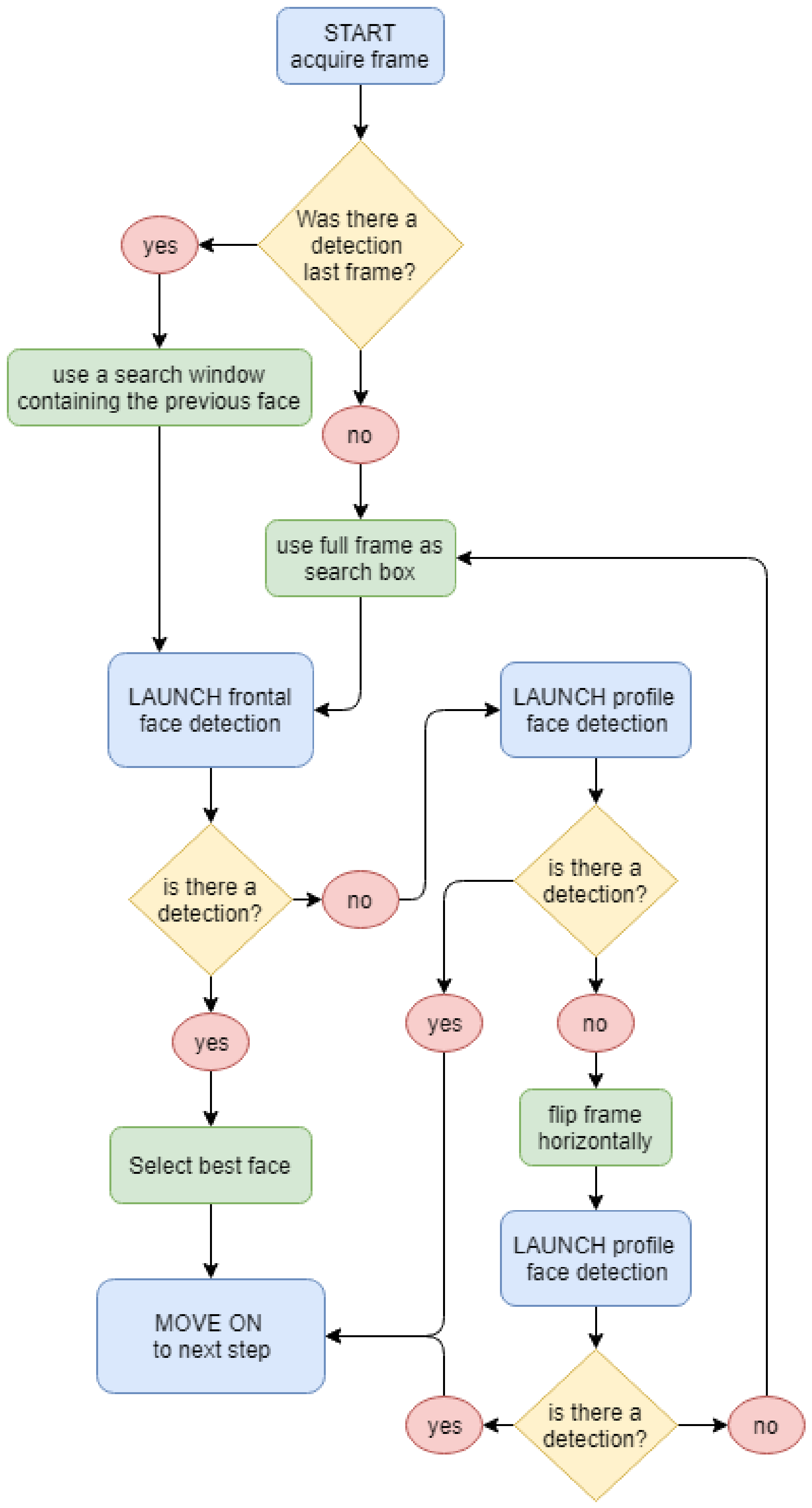

START
acquire frame
Was there a detection last frame?
yes
no
use a search window containing the previous face
use full frame as search box
LAUNCH frontal face detection
LAUNCH profile face detection
is there a detection?
no
is there a detection?
yes
yes
no
Select best face
flip frame horizontally
LAUNCH profile face detection
MOVE ON to next step
yes
is there a detection?
no

Fig. 3.2 Procedure for the acquisition of a face bounding box.

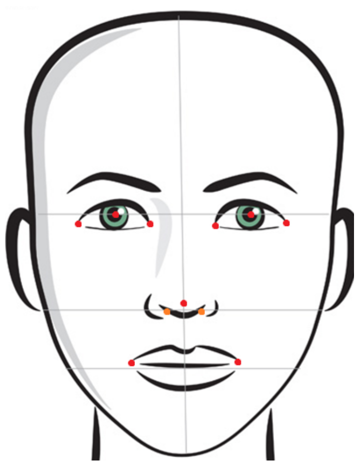


Fig. 3.3 The 9-point mark-up used in this work. Nostrils are considered as ancillary points.

of smaller bounding boxes for the pupil detection, due to further reducing of the quadrant by the Haar detection. However, this improves the robustness of the algorithm, by mitigating the effect of the eyebrows in the detection.

This method employs the use of histogram thresholding, by setting the threshold of the binarization to a percentage of the cumulative distribution function (CDF) of the histogram. In this particular case, the first thresholding is set to 5% of the image histogram, to capture the darkest pixels, which most commonly belong to the pupil. After, erosion is performed on the binarized image, and the largest connected component is isolated. Then, the minimum intensity pixel is found in that connected component, and the average pixel intensity is calculated in a 10x10 neighbourhood of that pixel. Then, after some erosion on the original eye ROI, the 15x15 neighbourhood around the pixel with minimum intensity is thresholded with the average intensity calculated before. The centroid of the resulting threshold is selected as the pupil candidate.

Eye corners are trickier to detect, as in low resolutions most information in the image gets lost, such as edges, or distinguishable dark spots, and so most techniques only work reliably in short distances to the camera, or , analogously, with higher resolutions. To tackle this issue, the leftmost and rightmost quarters of the eye ROI are selected, and corner detection is run on the resulting ROIs. If no corners are returned, the corner quality threshold in the Harris method is decreased until a corner is found. If still no corner is found, that means the ROI has no distinguishable features, and the centre of the ROI is selected as a candidate for the eye corner.

A few examples of the result of detection of eye features are shown in figure 3.5.

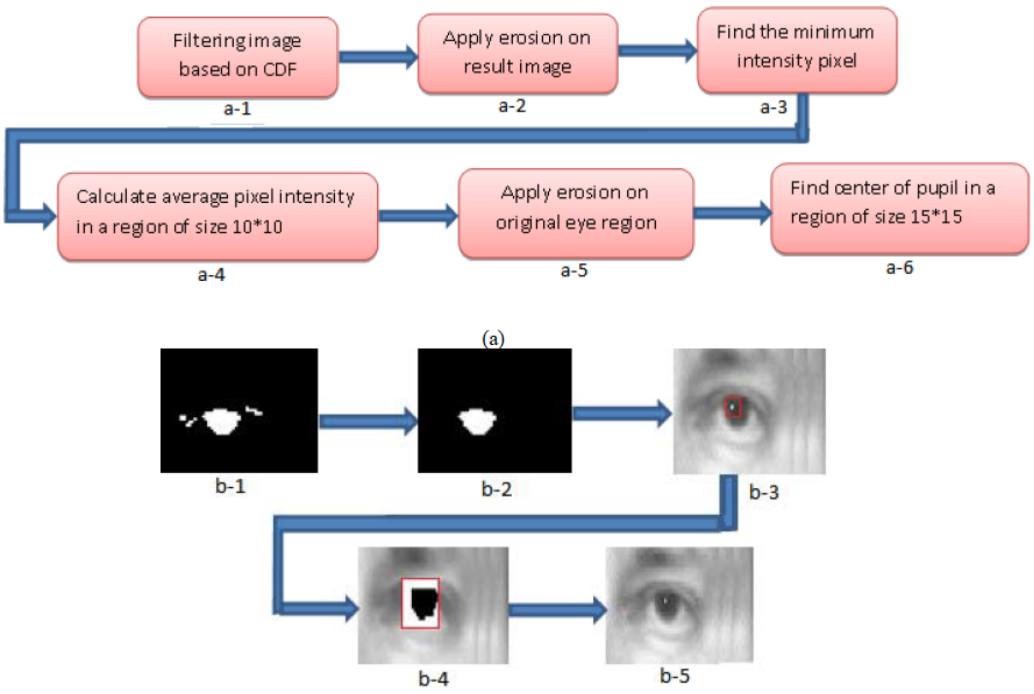

Filtering image based on CDF
a-1
Apply erosion on result image
a-2
Find the minimum intensity pixel
a-3
Calculate average pixel intensity in a region of size 10*10
a-4
Apply erosion on original eye region
a-5
Find center of pupil in a region of size 15*15
a-6
(a)
b-1
b-2
b-3
b-4
b-5

Fig. 3.4 The method for pupil detection used in this work (adapted from [4]).

Nose features

To detect the nose region, a search window in the y-centre of the face bounding box is selected, and a nose COD is run on it. If the detector returns a bounding box, it is considered as the nose candidate, and its nose tip is almost always at the centre of that region, so it is selected. The nostrils, not being necessary for the CASIR-IMPEP integration, can provide similar information of the face configuration. These can be detected with simple thresholding of the region just below the centre of the nose ROI, and extracting the centre of mass of the two largest blobs. Applying a contrast stretching proved useful in highlighting the nostrils before thresholding. Examples are shown in figure 3.6.

Mouth features

The mouth feature extraction starts, like the other features, by running a mouth COD in a smaller search window, with half the width of the face bounding box, and a quarter of its height. It is positioned slightly below the centre of the face. However, it was found that the Matlab trained COD returns a bounding box that often doesn't fully contain the mouth. To deal with this issue, some padding is added around the returned ROI. After, the bounding box is divided in two, and edge detection is launched on both subwindows. In the

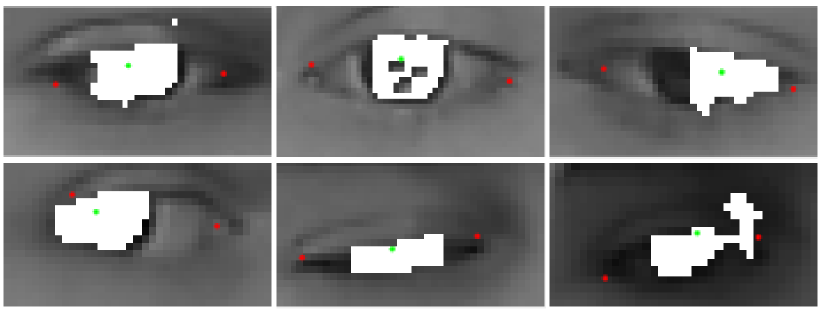


Fig. 3.5 A few examples of the detected eye regions and their features extracted. Although the eye corners aren't always accurate, the iris extraction method performs well.

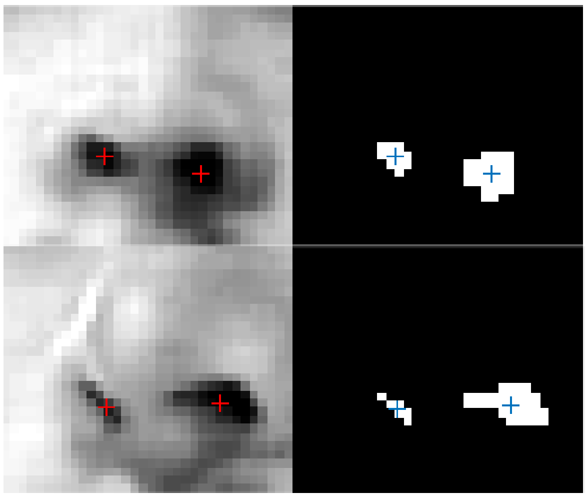


Fig. 3.6 Two frames showing the nose regions detected, centred on the nose tip, and with the detected nostrils shown.

resulting images, Harris corner detection is launched, and the mouth corners are selected as the leftmost and rightmost candidates, for the left and right mouth corners, respectively.

Profile features

In profile views, the features look quite different from a frontal view. Therefore, the independent detectors need to be able to address this issue. Particularly, the nose has the most distinct appearance, as in a full profile view, the nostrils aren't visible, and the nose tip is a much more salient point from the rest of the face.

The strategy for extracting the features in a profile view starts by assuming that only one of the eyes and mouth corners are visible, depending on the result of the face detection phase.

Since the eye, nose and mouth CODs are trained for frontal views, they cannot be used in the profile case, and so, an anchor point must first be found, and individual search windows

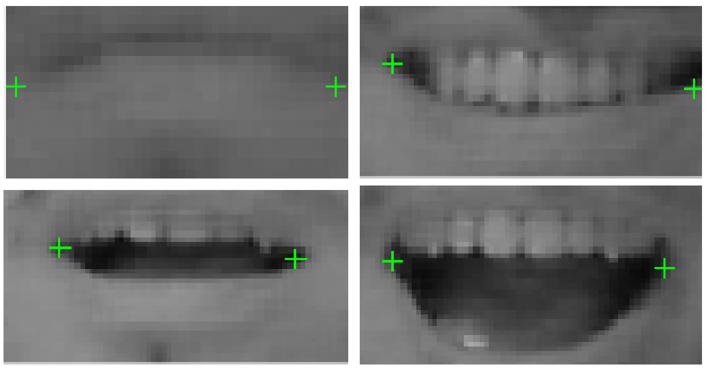


Fig. 3.7 A few examples of mouth and mouth corner detection, with different expressions.

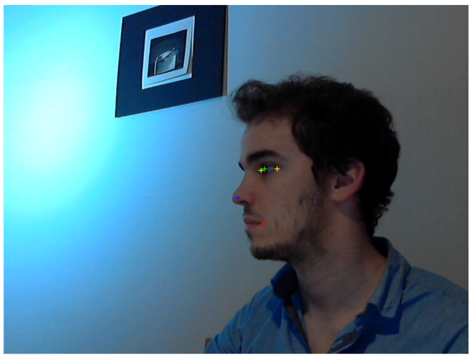
must be determined to extract the other features. Due to its saliency and central position in a face, the nose tip is chosen as that anchor point. It is detected using a mixture of skin colour segmentation and Active Contour search. First, a mask is selected, in the middle of the face bounding box. This assures that, if the detection really contains a face, the mask should only have skin, particularly the skin region close to the nose, between the eye and mouth, and shouldn't contain any facial hair. By restricting the initial mask to just skin, the active contour search will only extend up to the tip of the nose. Then, active contour search is performed for 200 iterations, using the Matlab function *activecontour*. It implements active contour search with two methods: the 'Chan-Vese' method and the 'edge' method. From empirical testing, the 'Chan-Vese' was determined to perform better to detect the nose contour.

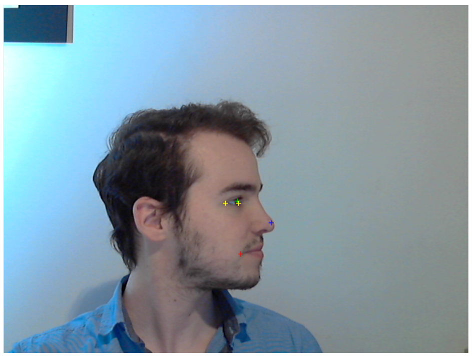
However, since the number of iterations is not set adaptively, sometimes the active contour search can extend beyond the figure of the nose. To adapt to this, a logical AND operation between the segmentation resulting from the active contour search and a segmentation based on skin colour is performed. The skin colour segmentation utilised has no author, and was adapted from Stack Overflow [ski]. It performs segmentation by making use of the Lab colour-space (similar to [29]) and the *YCbCr* colour-space. After combining the information from the active contour search and the skin colour segmentation, the leftmost pixel is selected as the nose tip.

After the nose tip is detected, search windows for the eye and mouth are defined, according to the predicted geometry of a human face. Specifically, the eye search region is a quarter of the face bounding box in width, and a sixth in height, and is placed slightly to the left and top of the nose tip. The mouth search area is a third of the bounding box in width and a sixth in height, and is placed just below the nose tip. These values were chosen empirically through testing with a small subset of profile faces taken from the internet.

The eye features are then extracted first by attempting to reduce the search window by running the eye COD on it. For full-profile views, the detector will rarely work, but in near-profile views it may sometimes detect the eye. Then, the same method for corner and pupil detection in frontal images is used.

The mouth corner is extracted by simple image thresholding, and selecting the rightmost pixel in the detected blob.


(a) Right profile.


(b) Left profile

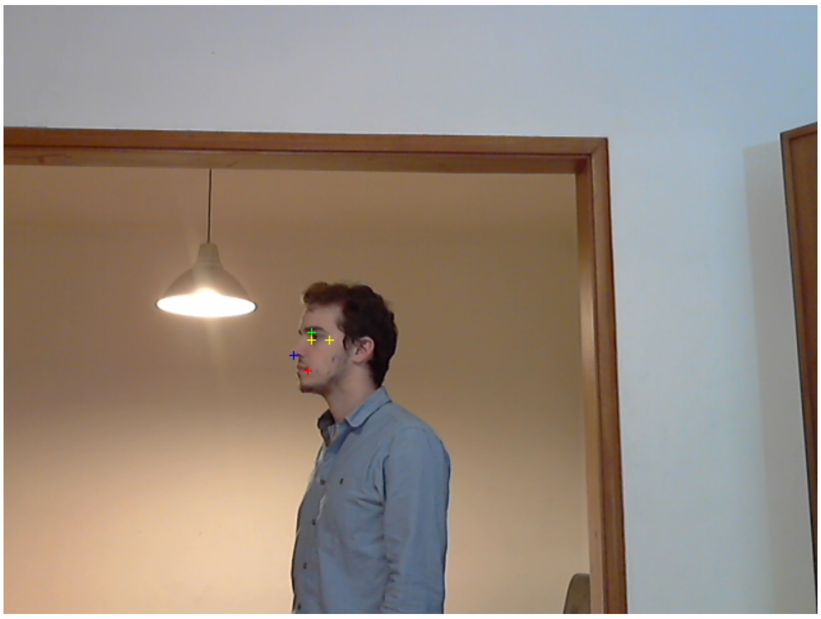


Fig. 3.9 Even at a distance, the system manages to provide a reasonable estimate of the profile feature points.

3.2.3 Feature Tracking

The phase of feature tracking can be further subdivided into three steps: KLT tracking, shape model verification, and applying geometric constraints. The procedure is described in figure 3.10.

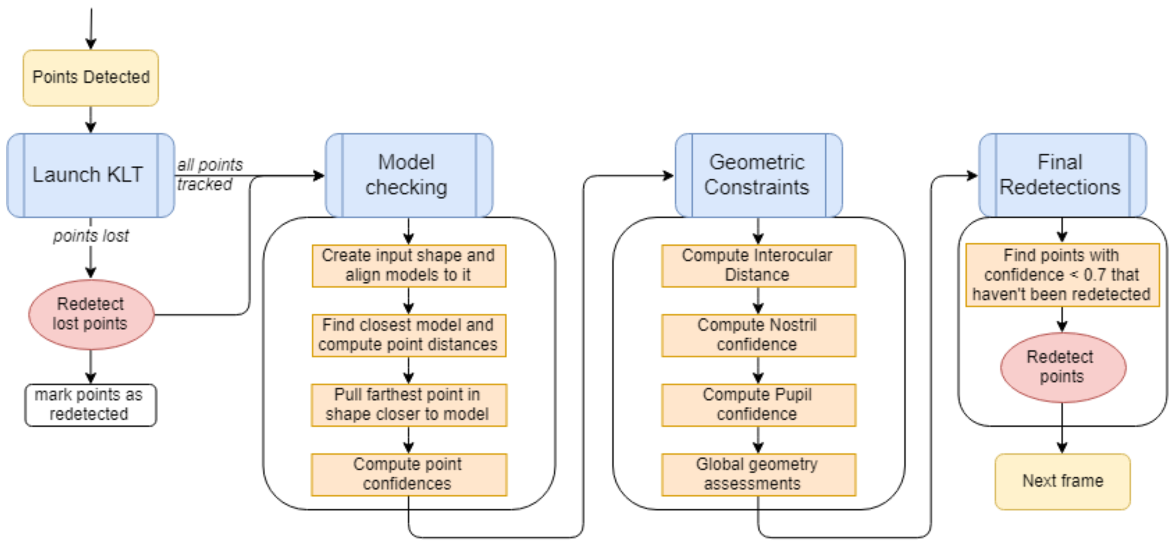

Points Detected
Launch KLT
all points tracked
points lost
Redetect lost points
mark points as redetected
Model checking
Create input shape and align models to it
Find closest model and compute point distances
Pull farthest point in shape closer to model
Compute point confidences
Geometric Constraints
Compute Interocular Distance
Compute Nostril confidence
Compute Pupil confidence
Global geometry assessments
Final Redetections
Find points with confidence < 0.7 that haven't been redetected
Redetect points
Next frame

Fig. 3.10 The tracking procedure used in this work.

For each frame after initialization, the previous points are fed to the KLT algorithm, and it obtains their update on the current frame. The KLT implementation used is the Matlab Computer Vision toolbox *PointTracker*. It comes with a measure of confidence of each point tracked, and also an indication of the points that were lost for some reason, and are not being tracked. This is important, because it only happens when a point goes out of the frame or a sudden difference in the frame causes it to land on a completely different neighbourhood. In this case, as it would make no sense to extract information from a lost point, the ones that KLT couldn't track are immediately re-detected and marked as re-detected for this frame, so no redundant redetections are triggered.

Shape model

The second phase is the verification of the shape against a pre-loaded shape model. This model consists of a set of 7 facial configurations, describing a frontal head pose with varying yaw angle. It was trained using the '300 Faces in the Wild' database [26], by using the following procedure:

- Gather ground truth for all 600 images.
- Normalize all the data and calculate the mean shape. The data is visualized in 3.11.
- Apply Principal Component Analysis to the data, and select the principal component which best describes the yaw variation of the face. From 3.12, it is apparent that the

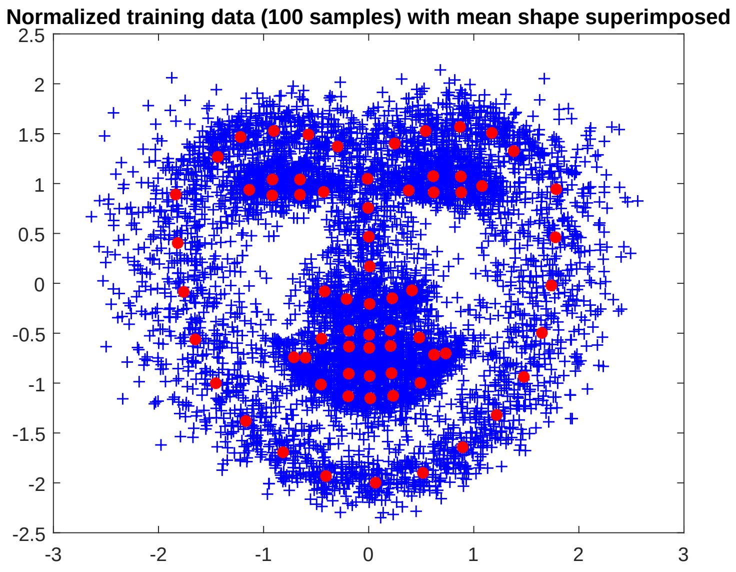


Fig. 3.11 Visualization of 100 normalised shapes from the 300W database, with their mean shape superimposed. The database uses a 68-point markup, from which 7 were extracted to compose the model used in the shape verification phase.

second principal component is the one describing yaw, and it accounts for 30.78% of the variance in the database.

- Generate a set of shapes varying the component coefficient.
- Select the 7 points required, specifically eye corners, nose tip and mouth corners. The resulting shapes are shown in figure 3.13.

All the seven models are aligned to the current tracked shape using Weighted Extended Orthogonal Procrustes alignment [3], as explained in section 2.3.1. The implementation used was my own, as no available Matlab code for WEOP transform was found.

Based on the distance to each model, the current face configuration is found, corresponding to the closest aligned model. Then, the distance between the tracked shape and the best aligned model is calculated, and the farthest point in the tracked shape is pulled closer to the model. This method helps deal with outlying points, holding off a re-detection unless necessary. An example is shown in figure 3.14.

If the current tracked shape is too far from the closest aligned model - the distance threshold was found through trial and error - the system triggers a full re-detection of the face.

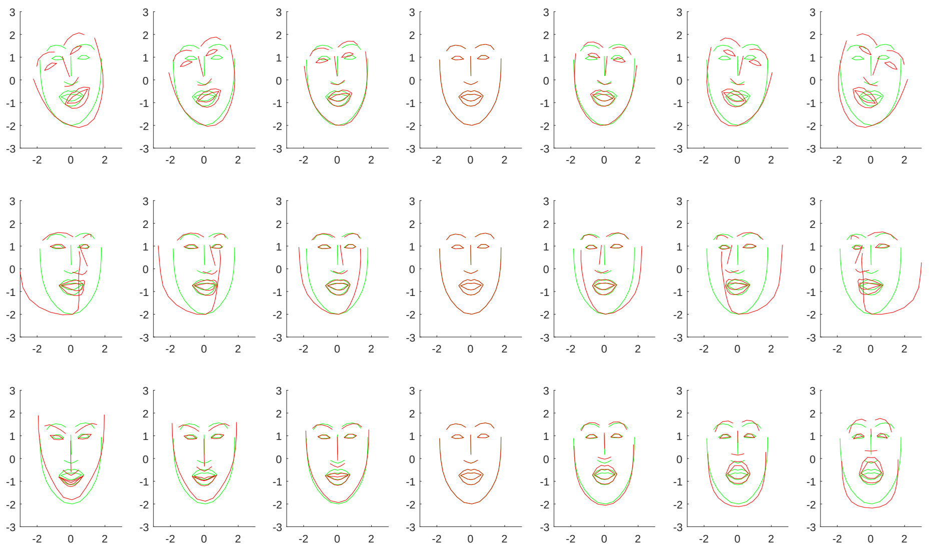


Fig. 3.12 First three principal components and their effect visualized, using the full 68-point mark-up in the 300W database.

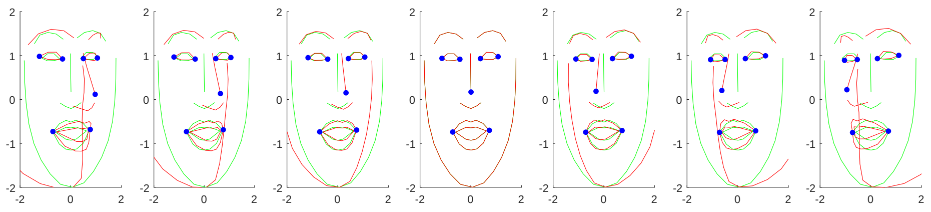


Fig. 3.13 The seven models used to describe the face in different yaw angles.

Feature confidences

If the tracked shape is close enough to the closest model, extra measures are taken to improve the accuracy. A set of confidences is attributed to the points, providing a heuristic from which the system decides when to re-detect a feature. These confidences are a number between 0 and 1, and the threshold from which they trigger re-detection was empirically chosen as 0.60.

Since the seven points belonging to the shape are already evaluated, their confidence is a measure of their distance to the correspondence in the closest model divided by the mean distance. This way, the point with the largest distance from the model will have its confidence reduced the most.

For the case of the irises, since there is no information from the models, a specific confidence function was designed, based on the geometry of an eye.

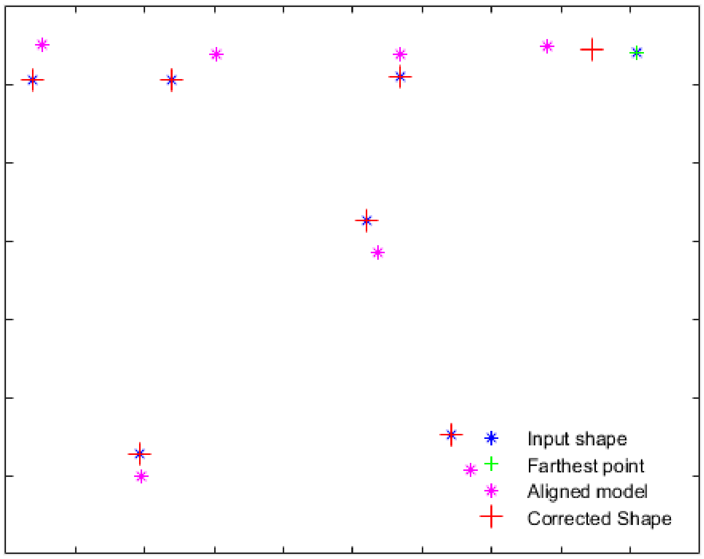


Fig. 3.14 An example of the described shape correction method. The point in the input shape with the farthest distance to its corresponding point in the closest aligned model is pulled closer to the model by a factor of 50%.

Taking into consideration that the pupil is most commonly placed in the centre of its corresponding eye corners, the confidence function designed is equal to 1 when this is the case, decreasing as the pupil moves away from that centre. Therefore, two linear functions taking as input the displacement vector towards the centre of the eye corners. Consider the detected pupil is at coordinates (a,b), and the left and right corners are at (x_1,y_1) and (x_2,y_2), respectively.

$$conf_x = \begin{cases} 0, & \text{if } a \notin [x_1\ x_2] \\ 1 - \frac{0.2}{|x_1 - x_2|}|a - c_x|, & \text{if } a \in [x_1\ x_2] \end{cases}$$

Similarly,

$$conf_y = 1 - \frac{1}{0.15 D_{IO}}|b - c_y|$$

Where c_x and c_y represent the x and y coordinates of the centre point of the two eye corners, and D_{IO} represents the current interocular distance of the face, which is the distance between the outermost eye corners. Both x and y confidences are combined to define the pupil confidence $conf_p$ by their minimum: $conf_p = \min(conf_x, conf_y)$.

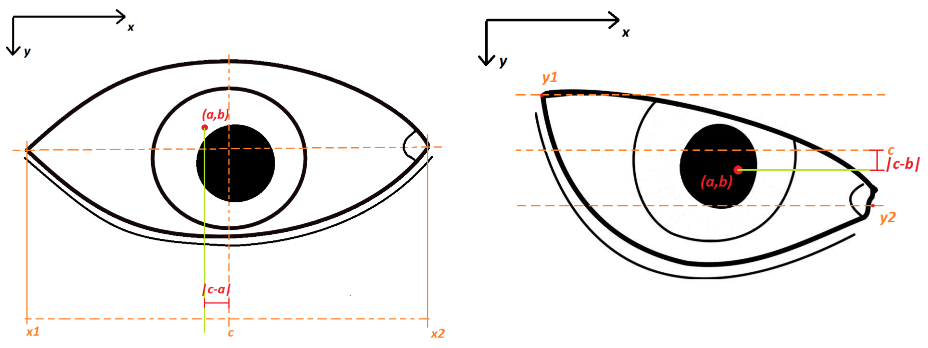


Fig. 3.15 Description of the eye pupil confidence atribution parameters.

3.3 User Interface

A graphical user interface (GUI) was created to interact with the tracking algorithm. It was designed with the help of the GUIDE tool from Matlab, and can be exported as a standalone application. It shows the current frame, superimposing the tracked points, distinguishing them by colour: yellow for the eye corners, green for the pupils, blue for the nose tip and red for the mouth corners. A reset button was added to manually force a full re-detection.

3.4 Results

3.4.1 Testing Databases

To be able to assess the performance of the designed algorithm, it was necessary to test it on annotated video sequences. After a survey on existing facial feature tracking databases, the conclusion is that not a lot exist that fit the purpose of this work. The most common shortcomings are:

- **Short distance**: most databases used in gaze estimation do not focus on robustness to distance, and so, many rely on sequences with high resolution or 'Skype-like' distances.
- **Good continuity**: a lot of databases do not feature good continuity between frames, rendering testing of the tracking procedure impossible, as the system is forced to re-detect on each frame.
- **Variable pose**: databases which contained heavy pose variation, and sustained profile head poses were scarce.
- **Free access**: a lot of surveyed databases were not free to use, and so were discarded.

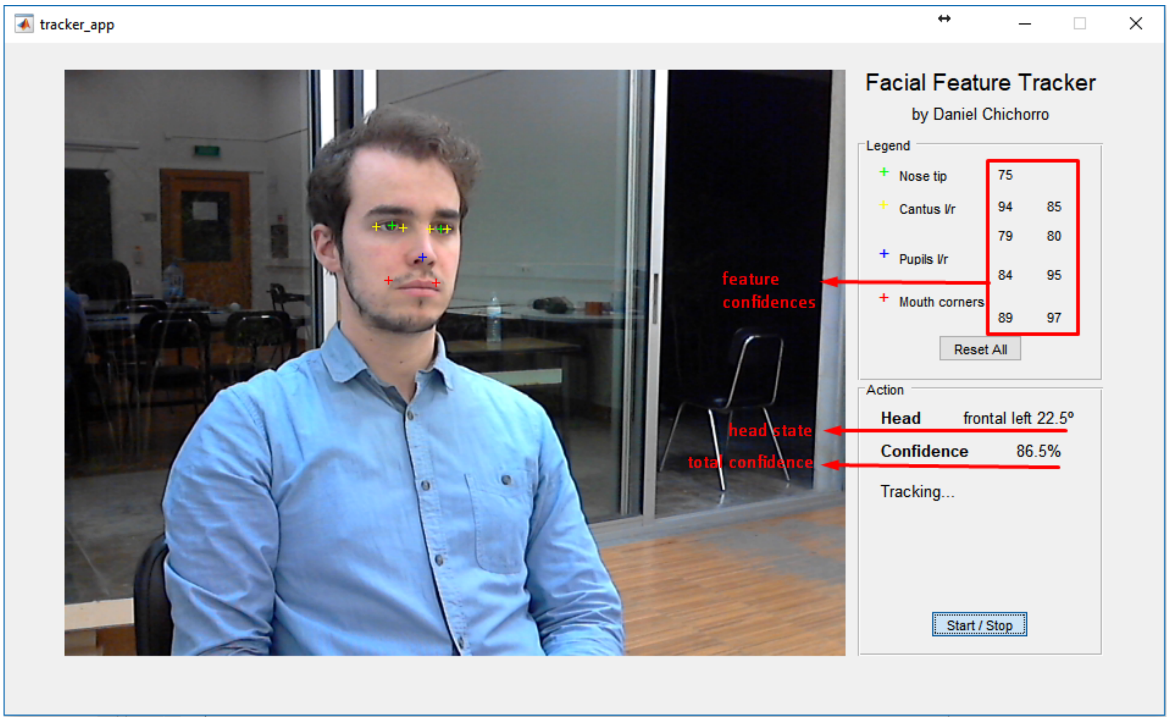


Fig. 3.16 The user interface created. It displays the point confidences, the head pose of the user being tracked, and provides some text info regarding the current phase of the algorithm (tracking/detection).

Two databases were considered for testing: CASIR's own gaze estimation database [CAS], and 300 Videos in the Wild [27]. The former contains image sequences captured with Kinect V1.0, at a considerable distance. However, this database proved very challenging, as the poor image quality and lack of continuity due to low frame-rate led the Haar detectors to underperform, consequently rendering the results quite unsatisfactory. The 300VW database consists of 116 videos annotated using the same 68-point mark-up as [26].

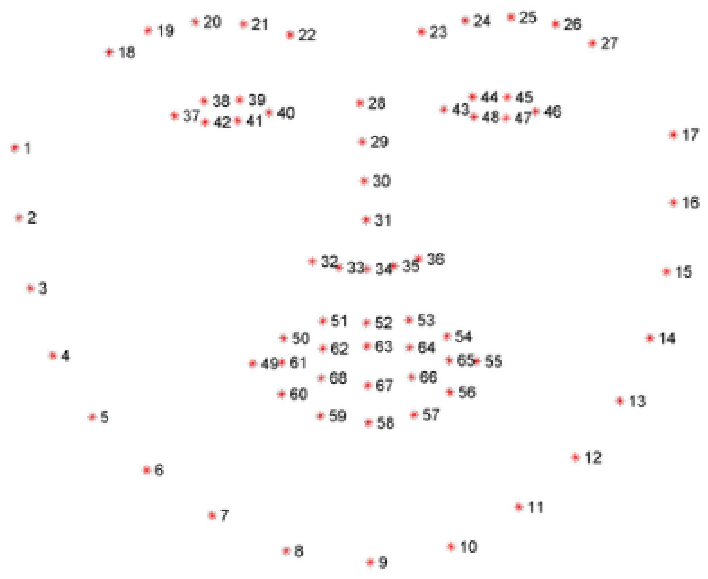


Fig. 3.17 The 68 point mark-up used in the 300VW and 300W databases. It contains the eye corners, nose tip, and mouth corners, among other landmarks and contours.

The 300VW database is divided into three categories, based on the exhibited scenarios [26].

- The first category shows people recorded in well-lit conditions, with arbitrary expressions in various head poses, but without external occlusions.
- The second category adds variety to the illumination, featuring dark rooms or overexposed shots.
- The third category adds more difficulty by introducing occlusion such as hands going over the face, more extreme head pose variation, etc.

Examples of these categories are shown in figure 3.19.

However, this database doesn't tackle strongly the distance problem, presenting a majority of videos with a relatively high resolution. Furthermore, most sequences have the user focused

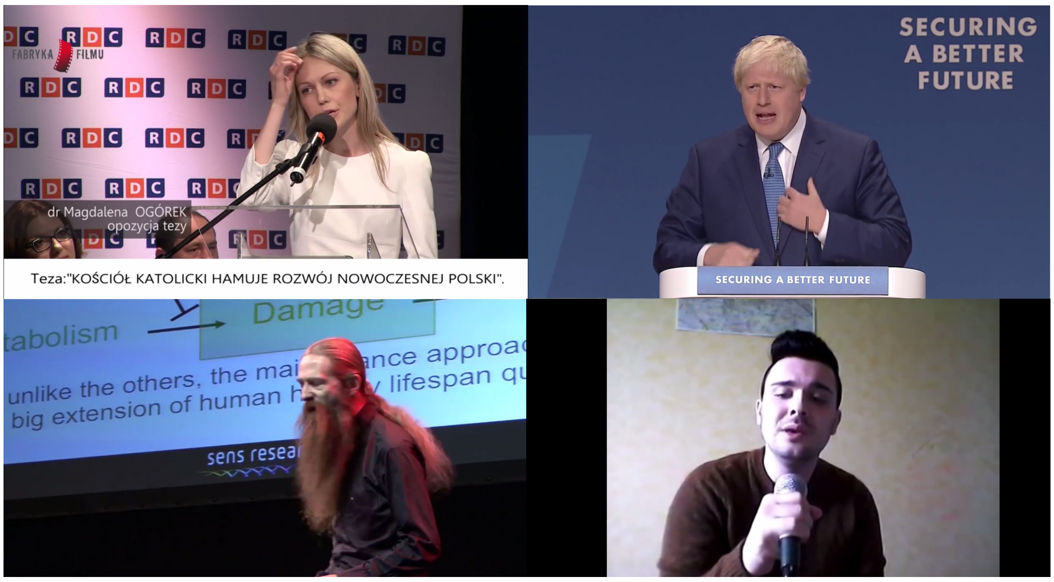

SECURING A BETTER FUTURE
dr Magdalena OGÓREK opozycja tezy
Teza:"KOŚCIÓŁ KATOLICKI HAMUJE ROZWÓJ NOWOCZESNEJ POLSKI".

Fig. 3.18 A few frames of videos in the 300VW dataset.

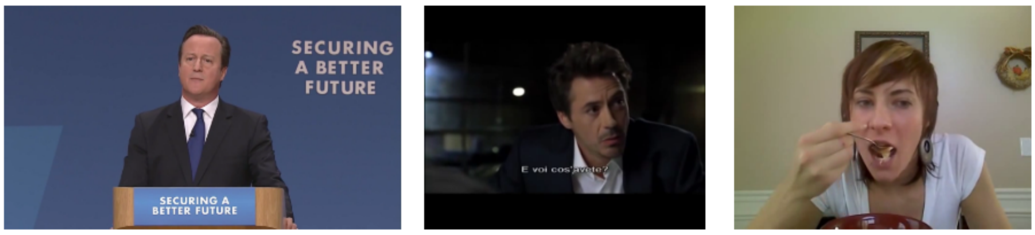


Fig. 3.19 Examples of frames in the three categories in 300VW: category 1 (left), category 2 (middle) and category 3 (right) (from [26]).

on the general direction of the camera, leading to a majority of frontal views. To address these issues, a set of video sequences was created and annotated at lower resolutions, to be able to test the algorithm's robustness. Some examples are in figure 3.20.

3.4.2 Performance measures

Throughout literature the most common measure of success is the normalized displacement or error (see for example [26, 27, 12, 6]). The normalisation can be done with a measure of scale of the face, such as the interocular distance, i.e. the distance between the outermost eye corners [27]. However, in extreme head poses, or in profile views, the interocular distance doesn't provide a good scale reference, and so the results obtained over these frames are made to seem worse than they might be. In the sequences generated and annotated, which feature profile views much more heavily, two scale references are considered: the interocular

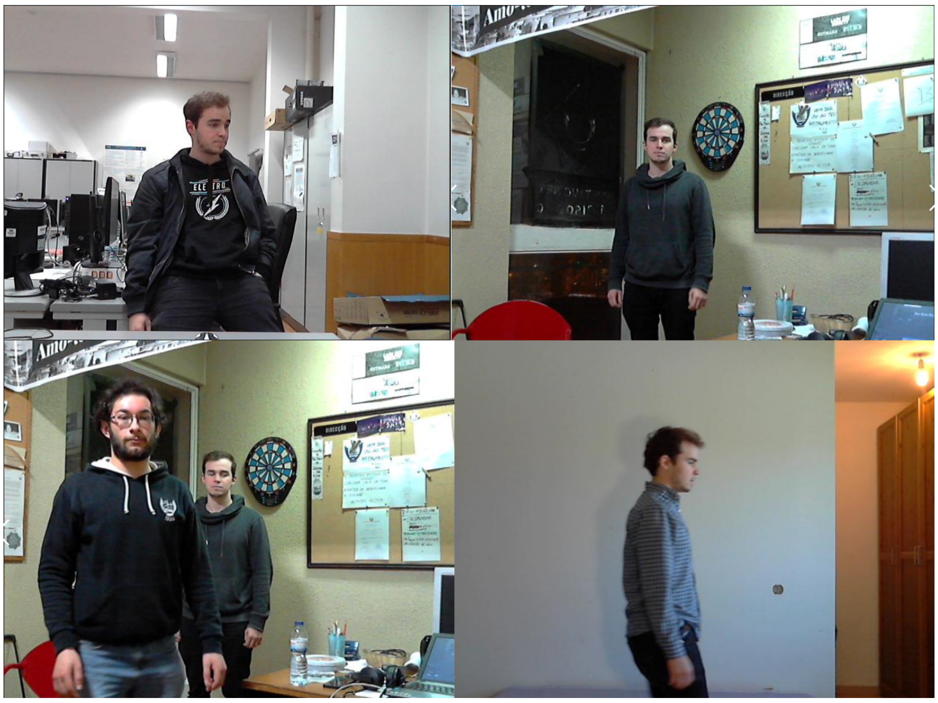


Fig. 3.20 Examples of sequences generated. These exhibit a much poorer resolution, and also account for full profile views of the user, allowing for better testing of the profile feature detector.

distance for near-frontal views, and the eye-to-mouth distance for profile views. As can be seen from figure 1.1, the distance from eye corner to mouth corner in a profile pose is quite comparable to the interocular distance in a frontal pose.

A more absolute measure would be the displacement error in pixels, without normalisation, which is also used (for example, see [24]), but it doesn't provide a good insight considering scale variations. However, it must be considered that as the distance to the detection increases, the normalised error is bound to increase as well, as the ratio becomes larger.

3.4.3 Results and Discussion

Figure 3.21 shows the result of the proposed algorithm in the 300VW database, and for each of the tracking categories, and figure 3.22 superimposes the performance of the proposed method with the results from the 300 Faces in-the-wild competition [26], which was a landmark localization competition using static images. This comparison serves to show that although it is a harder task to track features over long video sequences rather than detecting them on static images, the result of this algorithm still surpasses the baseline established by Sagonas et. al., which consisted of an AAM-based method [26].

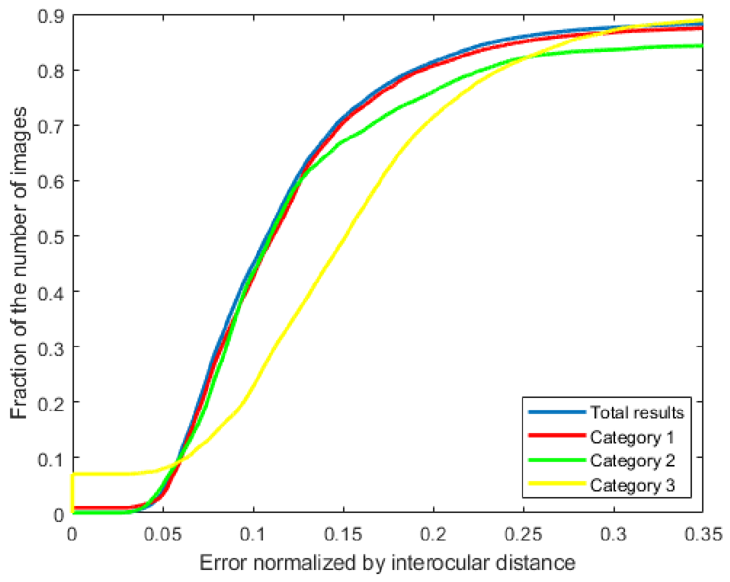


Fig. 3.21 The cumulative error distribution curves for the videos in the 300VW database, and in each of the categories.

Figure 3.23 shows the published results of the 300 Videos in the Wild competition. Unfortunately, a direct comparison to the performance of the current work is not feasible, as the metric isn't the same. The performance measure shown in 3.23 is the Root-Mean-Square normalized error, not the normalized Euclidean distance. However, it is clear the proposed method does not challenge the competition results.

This sub-par performance can be explained by the difference in requirements of the systems.

- In the 300VW competition, the goal is to track 68 or 49 landmarks over every frame (see fig. 3.17). This system only tracks 7 of those landmarks, being made to work at larger distances, where a 68-point markup wouldn't be feasible.

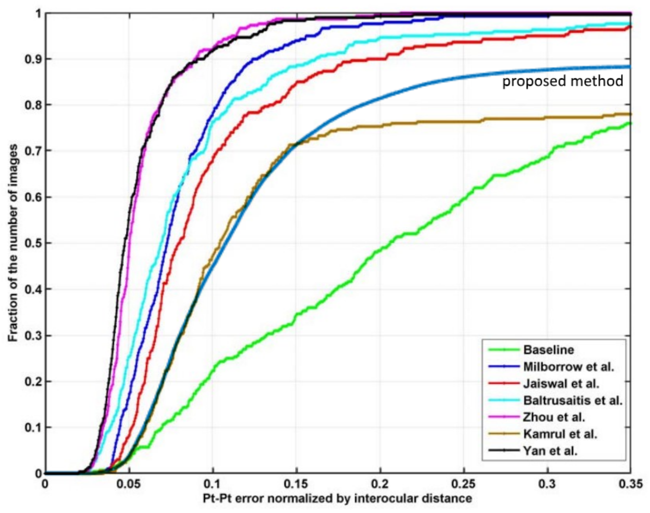


Fig. 3.22 The published results of the 300-W competition (a landmark detection competition, using static images), with the overall result of the proposed method on the 300VW database superimposed. The baseline shown consists of an Active Appearance Model-based method [26].

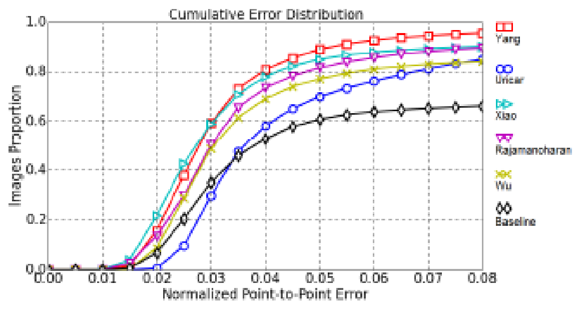
- The required performance of the methods participating in the competition is of at least 0.5 frames/sec, which is a low threshold. The proposed system can track at an average of 10 frame/sec, resulting in almost real-time performance.
- The proposed system incorporates profile view detection, which in some cases confuses the face detection pipeline, leading to false positives that generate a lot of error. This functionality, however, allows for a more complete capture of unconstrained scenarios.

The new sequences generated and annotated to evaluate this method provide a harder challenge. In all of them, the subject is at least 2 metres from the camera, with a resolution of 640x480. This leads to really poor conditions in which to perform face and facial feature detection and tracking. Therefore, the results are expected to be worse than the 300VW database. However, no comparison can be made to other methods, as they weren't tested on the same sequences (see discussion of appropriate datasets in section 3.4.2).

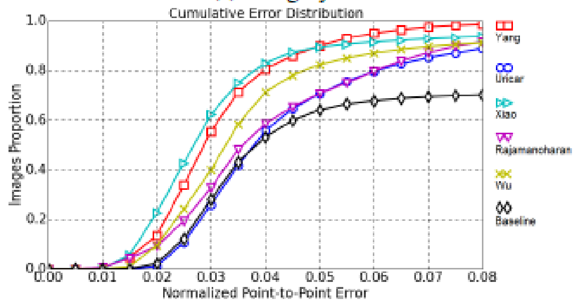
The result over the sequences generated is shown in figure 3.24.

Upon first inspection, the curve represented in figure 3.24 describes a very undesirable scenario. However, there are some key factors to be taken into account.

- The increase in distance means the ratio of pixel displacement to normalized displacement decreases, therefore showing overall higher error in this sequence.



(a) Category one



(b) Category two

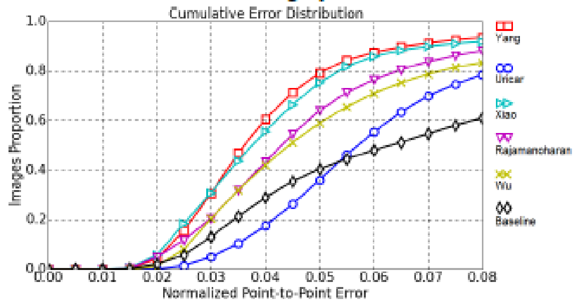


Fig. 3.23 The published results of the 300VW competition. The performance measure is slightly different, consisting of the RMS error, instead of the Euclidean distance used in the proposed work (from [27]).

- The task of face detection is made harder by the distance as well, leading to even more false positives. Coupled with the previous point, the error suffers a lot.

In fact, considering the challenges, 80% of the frames having an error smaller than the interocular distance means the system is providing a good estimate of the face geometry most of the time, even at long distances from the camera. Therefore, the usefulness of this system is still of considerable significance.

From the analysed results, one major flaw of this system is the face detection method used leading to too many false positives, which generate a lot of error (if the displacement is normalised by the interocular distance and the tracking is occurring outside of the expected

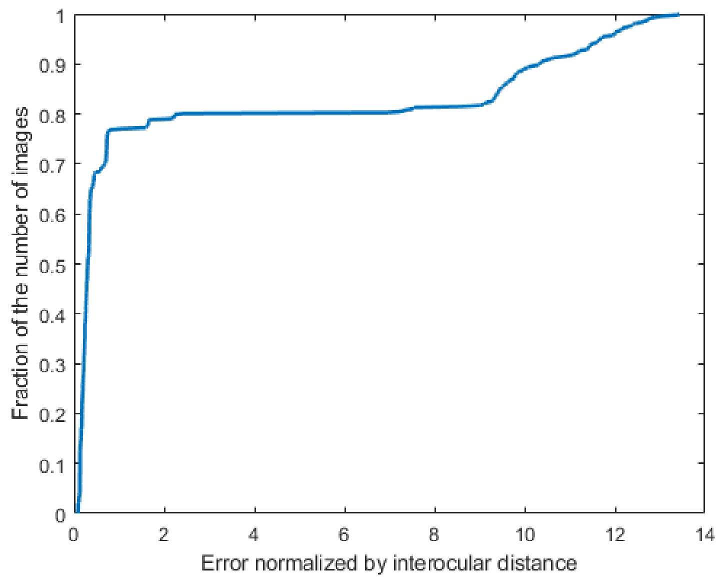


Fig. 3.24 The cumulative error distribution curve on the produced sequences.

face region, the error is very high). However, considering the difficulty of the generated sequences, the results show that the proof of concept is viable even so.

Since the goal of this work is to be integrated into the CASIR-IMPEP architecture, it is relevant to analyse the temporal performance of the designed system. The requirements for integration in CASIR-IMPEP are a minimum of 2 frames per second, or less than half a second of delay.

Deriving from the hybrid detection/tracking approach, the system will have different time performances considering the dynamics of the scene. A full re-detection on a frame takes considerably longer than the task of tracking. Therefore, in sequences with more movement, such as frequent occlusion and high variation in head pose, more partial or full re-detections are triggered, leading to an overall lower temporal performance as a more stable sequence, in which mostly tracking is occurring. Also, profile detection proved to be more time-consuming, essentially due to the Active Contour search, which is computationally intensive. Therefore, in sequences featuring profile views, the performance is expected to be lower. A temporal analysis of the performance of this work is shown in table 3.1.

As can be seen, the time performance of this system is superior to the requirements of the CASIR-IMPEP architecture. In fact, since it was tested in Matlab with the GUI-based application developed (which provides a visualization of the frame being tracked), a lot of improvement can be expected by porting to OpenCV. Also, the CASIR-IMPEP architecture

Scenario	Average fps
Stable sequence	9.43
Mild movement	6.21
Medium movement	5.79
High movement	5.60
Profile stable	4.24
Profile dynamic	2.32

Table 3.1 Temporal analysis of designed solution, in six different scenarios. Mild movement sequences can consist of a stable person talking, medium and high movement sequences incorporate varying head pose, movement along the frame, and occlusions, in increasing dynamic. Two scenarios almost all comprised of profile views were considered.

boasts a better processing machine than the laptop this system was tested on, so further improvement is expected.

Chapter 4

Conclusions and Future Work

The results of the proof-of-concept designed show that the solution meets the core functional requirements. Although the proposed framework compares positively to the related work, it does not stand out. However, this is to be expected, as the functional requirements are not the same - for example, the designed system is steered to work at longer distances (more than 2m), where resolution is lower, and it is also capable of dealing with profile views, something that isn't thoroughly explored in the field. Since these use cases are not sufficiently featured in the only benchmark for facial feature tracking (the 300VW database [27]), a direct comparison can't be established.

The designed system is ready to be integrated into the CASIR-IMPEP architecture, which is the main goal of this work. It performs above the minimum functional and temporal requirements, and is able to receive further optimization.

There are many steps that can be taken to improve the performance of this framework.

- Face detection algorithms: the method used for face detection produces a lot of false positives, especially the profile detectors, particularly in awkward lighting conditions (e.g. one of the sides of the face being much more lit up than the other), which explain much of the poorer performance in the test sequences. In fact, without the profile feature detection, the system was able to stick to the face significantly better, although producing less reliable results when the face presented extreme poses. A better solution for face detection is of the highest priority. Additionally, an optical flow-based method to assess the area in which to attempt to detect a face could prove very robust in eliminating false positive detections. If the tracked points are very stable while there is a lot of movement in the frame, odds are the tracker is lost.

- Applying different feature extraction methods based on resolution. A number of techniques are better suited for high resolution images, which aren't a thoroughly explored use case in this work, and were heavily featured in the 300VW database. The selection of the best method to use based on the size of the face bounding box could greatly improve the overall results.

- Adding the functionality of distinguishing between an open eye and a closed eye. Trying to extract the position of the pupil is unnecessary in this case, as is estimating the gaze.
- Lowering the overall confidence of the tracking based on the resolution of the detected face. Therefore, as the face gets smaller (i.e. its distance from the camera increases), the system would lose confidence on the accuracy of the detection/tracking. This information would be useful for the integration in the probabilistic framework of CASIR.
- Creation of a facial feature tracking database that thoroughly tests the use case of distant subjects, integrating profile views.

Aside the algorithm improvement steps discussed above, future work would also deal with porting the system to OpenCV in ROS, and installing in the CASIR-IMPEP architecture, to enhance the gaze estimation functionality of this system.

References

[CAS] Casir gaze estimation database. http://mrl.isr.uc.pt/experimentaldata/public/gaze-casir/. Accessed: 2018-05-09.

[ski] Skin colour segmentation matlab code. https://www.mathworks.com/matlabcentral/answers/43123-skin-color-detection-problem. Accessed: 2018-05-22.

[3] Akca, D. (2003). Generalized procrustes analysis and its applications in photogrammetry. Technical report, ETH Zurich.

[4] Asadifard, M. and Shanbezadeh, J. (2010). Automatic adaptive center of pupil detection using face detection and cdf analysis. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, page 3.

[5] Batista, J. P. (2007). Locating facial features using an anthropometric face model for determining the gaze of faces in image sequences. In *International Conference Image Analysis and Recognition*, pages 839–853. Springer.

[6] Bengoechea, J. J., Cerrolaza, J. J., Villanueva, A., and Cabeza, R. (2014). Evaluation of accurate eye corner detection methods for gaze estimation. *Journal of Eye Movement Research*, 7(3).

[7] Bourel, F., Chibelushi, C. C., and Low, A. A. (2000). Robust facial feature tracking. In *BMVC*, pages 1–10.

[8] Bradley, D. and Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of graphics tools*, 12(2):13–21.

[9] Chrysos, G. G., Antonakos, E., Snape, P., Asthana, A., and Zafeiriou, S. (2018). A comprehensive performance evaluation of deformable face tracking “in-the-wild”. *International Journal of Computer Vision*, 126(2-4):198–232.

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

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

[12] Cristinacce, D. and Cootes, T. F. (2004). A comparison of shape constrained facial feature detectors. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 375–380. IEEE.

[13] Cristinacce, D. and Cootes, T. F. (2006). Feature detection and tracking with constrained local models. In *Bmvc*, volume 1, page 3.

[14] Graham, G. H., Unruh, J., and Jennings, P. (1991). The impact of nonverbal communication in organizations: A survey of perceptions. *The Journal of Business Communication (1973)*, 28(1):45–62.

[15] Hansen, D. W. and Ji, Q. (2010). In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):478–500.

[16] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer.

[17] Lanillos, P., Ferreira, J. F., and Dias, J. (2015). Multisensory 3d saliency for artificial attention systems. In *3rd Workshop on Recognition and Action for Scene Understanding (REACTS), 16th International Conference of Computer Analysis of Images and Patterns (CAIP)*, pages 1–6.

[18] Lanillos, P., Ferreira, J. F., and Dias, J. (2017). A bayesian hierarchy for robust gaze estimation in human–robot interaction. *International Journal of Approximate Reasoning*, 87:1–22.

[19] Li, Y., Wang, S., Zhao, Y., and Ji, Q. (2013). Simultaneous facial feature tracking and facial expression recognition. *IEEE Transactions on Image Processing*, 22(7):2559–2573.

[20] Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision.

[21] Macesanu, G., Comnac, V., Moldoveanu, F., and Grigorescu, S. M. (2014). A time-delay control approach for a stereo vision based human-machine interaction system. *Journal of Intelligent & Robotic Systems*, 76(2):297–313.

[22] Massé, B., Ba, S., and Horaud, R. (2017). Tracking gaze and visual focus of attention of people involved in social interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[23] Mead, R., Grollman, D. H., Lim, A., Yeung, C., Stout, A., and Knox, W. B. (2018). Hri 2018 workshop: Social robots in the wild. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 399–400. ACM.

[24] Ong, E.-J. and Bowden, R. (2011). Robust facial feature tracking using shape-constrained multiresolution-selected linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1844–1859.

[25] Pieterson, W., Ebbers, W., and Madsen, C. Ø. (2017). New channels, new possibilities: A typology and classification of social robots and their role in multi-channel public service delivery. In *International Conference on Electronic Government*, pages 47–59. Springer.

[26] Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S., and Pantic, M. (2016). 300 faces in-the-wild challenge: Database and results. *Image and Vision Computing*, 47:3–18.

[27] Shen, J., Zafeiriou, S., Chrysos, G. G., Kossaifi, J., Tzimiropoulos, G., and Pantic, M. (2015). The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 50–58.

[28] Shi, J. et al. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE.

[29] Skodras, E. and Fakotakis, N. (2011). An unconstrained method for lip detection in color images. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 1013–1016. IEEE.

[30] Smith, L. I. (2002). A tutorial on principal components analysis. Technical report.

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

[32] Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features.

[33] Tong, Y., Wang, Y., Zhu, Z., and Ji, Q. (2007). Robust facial feature tracking under varying face pose and facial expression. *Pattern Recognition*, 40(11):3195–3208.

[34] Uricár, M., Franc, V., and Hlavac, V. (2015). Facial landmark tracking by tree-based deformable part model based detector. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–17.

[35] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.

[36] Wang, N., Gao, X., Tao, D., and Li, X. (2014). Facial feature point detection: A comprehensive survey. *arXiv preprint arXiv:1410.1037*.

[37] Wang, N., Gao, X., Tao, D., Yang, H., and Li, X. (2018). Facial feature point detection: A comprehensive survey. *Neurocomputing*, 275:50–65.

[38] Yang, J., Stiefelhagen, R., Meier, U., and Waibel, A. (1998). Real-time face and facial feature tracking and applications. In *AVSP'98 international conference on auditory-visual speech processing*.

[39] Zhang, C. and Zhang, Z. (2010). A survey of recent advances in face detection.

[40] Zhou, Z.-H. and Geng, X. (2004). Projection functions for eye detection. *Pattern recognition*, 37(5):1049–1056.