UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

wizAI
Solutions with AI

Face and Gaze Controlled Onscreen Presentations (FAGCOP)

Masterarbeit

zur Erlangung des Grades

Master of Science

im Studiengang Web Science

vorgelegt von

Nabil Bukhari

Betreuer: Prof. Dr. Ulrich Furbach, Arbeitsgruppe Künstliche Intelligenz, Fachbereich Informatik, Universität Koblenz-Landau

Erstgutachter: Dipl.-Inform. Markus Maron, wizAI Solutions GmbH

Koblenz, im März 2020

Abstract

On-screen interactive presentations have got immense popularity in the domain of attentive interfaces recently. These attentive screens adapt their behavior according to the user's visual attention. This thesis aims to introduce an application that would enable these attentive interfaces to change their behavior not just according to the gaze data but also facial features and expressions. The modern era requires new ways of communications and publications for advertisement. These ads need to be more specific according to people's interests, age, and gender. When advertising, it's important to get a reaction from the user but not every user is interested in providing feedback. In such a context more, advance techniques are required that would collect user's feedback effortlessly. The main problem this thesis intends to resolve is, to apply advanced techniques of gaze and face recognition to collect data about user's reactions towards different ads being played on interactive screens. We aim to create an application that enables attentive screens to detect a person's facial features, expressions, and eye gaze. With eye gaze data we can determine the interests and with facial features, age and gender can be specified. All this information will help in optimizing the advertisements.

Statement

I hereby certify that this thesis has been composed by me and is based on my work, that I did not use any further resources than specified – in particular, no references unmentioned in the reference section – and that I did not submit this thesis to another examination before. The paper submission is identical to the submitted electronic version.

I agree to have this thesis published in the library. ja ☒ nein ☐

I agree to have this thesis published on the Web. ja ☒ nein ☐

The thesis text is available under a Creative Commons License (CC BY-SA 4.0). ja ☒ nein ☐

The source code is available under a GNU General Public License (GPLv3). ja ☐ nein ☒

The collected data is available under a Creative Commons License (CC BY-SA 4.0). ja ☒ nein ☐

Koblenz, den 5. März 2020

Contents

List of Figures

List of Tables

List of Algorithms and Code Snippets

Chapter I

APPROACH OF THE INVESTIGATION

1.1 Problem Identification

Advertisement is a means of transferring information about a product or service to interested users. An advertisement can inform and influence people to subscribe to a service or buy a product. Advertisement is present in multiple forms and information can be communicated using television, print media, radio, internet, hoardings, mailers, sponsorships, posters, events, endorsements. A well-advertised product is more probable to be sold and can affect the trends in society.
The fact that 85% [4] of Google's revenue is generated from advertising which confirms its importance in today's world.
The problem lies in the viewership of advertisements. An average person is served with over 1700 advertisements per month but merely half of them are viewed while the other half is neglected. From the viewed advertisement, a very small number are browsed by the users.
There are a lot of challenges faced by the advertising industry which are listed below:

- Difficulty in reaching potential customers
- Elusive Audience
- Lack of interest of users in a certain product
- Inaccurate information

1.2 Formulation of the Problem

The advertisement domain has gradually evolved taking it to a point where we are moving to virtual, augmented reality and targeted advertisement.
The goal of this project is to identify user interest from a set of advertisements using visual technology and machine learning. The plan is to design an information extraction algorithm using dual technologies i.e. Eye Tracker and Webcam. User information in this scenario will be:

- Gather the information viewed on the screen by a user. This information can be sub-classified into text, image, and video extraction.
- Gather identity information of users by face recognition. User consent in this regard is specially taken care of.
- Gather demographic characteristic data of the user. This includes age and gender identity.
- Gather areas of interest of a user. This will define what areas on the screen are more likely to be seen by the user.

1.3 Project Goals

This project aims to learn the user's interests, identity, demography and other traits using head position estimation and gaze fixations in a controlled environment. The software is an ensemble of high tech algorithms to extract and analyze multidimensional information of a person using digital signage solutions. The module included are face detection using haarcascade, face recognition using convolutional neural network and deep learning using state of the art machine vision and image processing techniques, gender recognition, eye tracking using myGaze eye-tracker which will perform content extraction classified into text, image and video information viewed by the user, eye tracking using simple webcams, area of interest extraction using an ensemble of eye tracker, webcam and head position data.
User interests combined with a determined area of interest will be used by the application to display targeted advertising and content recommendation on the On-Screen player. The algorithm uses deep learning to extract meaningful information from a user to screen interaction. The components of this project are shown in Figure 1.1:

Figure 1.1: Components of the Project

A proposed name for the current project will be "FAGCOP" derived by Face and Gaze Controlled Onscreen Presentation. The system should then be able to answer the following questions:

- Who is the person?
- What are the demographics of the user?
- What is the principal focus of user attention on the screen?
- What parameters can add to the enhanced attention of the user on the screen?

1.4 Structure of Work

The thesis aims at the development of software that gathers and stores user interaction information with a presentation running on digital signage. Chapter 1 defines my approach of the investigation defining the identification of a problem, formulation, and project goals. Chapter 2 focuses on the review of a few pieces of research related to this thesis. Chapter 3 provides a deep dive into the theoretical backgrounds of concepts used in the implementation. It explains the core concepts used in face detection, face recognition, head position estimation, and gaze recognition. Chapter 4 deals with the definition of requirements and the concept of the software. It describes how the software is structured and how it works. Chapter 5 then explains the implementation. This chapter deals with the frameworks used and the procedure necessary to implement the software. Chapter 6 discusses the technical details and structure of our created database and the anonymity

parameter. In Chapter 7, the development process is evaluated with the explanation of evaluation parameters and accuracy for each algorithm. Thereby, problems that have arisen within the development process are documented to be able to conclude. Afterward, it is shown to what extent this project can be extended, recommendation and the possibilities that can be offered in the future.

Chapter II

LITERATURE REVIEW

2.1 Gaze Tracking state-of-the-art Research

Though less accurate, a number of published attempts are made in the domain of using webcams for gaze tracking. The methods involve a calibration phase and the accuracy is comparatively smaller than an infrared eye tracker. A linear regression model was initially built in this regard which needed a sparse calibration and was sensitive to head movements [33, 34]. The papers contributed to the human gaze via adaptive linear regression and eye image synthesis[44]. Another research conducted by Sugano et al. [55] used image saliency maps to estimate user gaze for calibration purposes, though it only provides a rough estimation of a user gaze [45]. PACE is a desktop application that performs eye-tracking through user interactions[44]. TurkerGaze, another webcam-based eye tracker is deployed on Amazon Mechanical Turk to predict saliency on images. The paper [35] also created a model to add head orientation to determine a user's gaze position with a median error of 3.22°, which validates our calculated results.

2.2 From Gaze Control to Attentive Interfaces

Hyrskykari et al. states that designing eye gaze applications is challenging [23]. The most common problem that needs to be solved is avoiding the "Midas touch" problem: the application should not react every time the target of the gaze changes, but only during appropriate situations, and at the right moment. Humans are good at detecting when the eye gaze demands action, but for computers, this is a big challenge. The research paper focuses on concrete eye gaze applications, and present a survey based on the domain they are intended for. The domain areas covered are (1) offline, noninteractive applications; (2) devices based on eye detection and eye contact sensing; and (3) gaze-contingent systems. The author gave good reviews of multiple eye gaze applications, but the paper lacked

descriptions of features and proper comparison between these tools, which makes the paper less useful than it could have been.

2.3 Real-Time Face Detection and Recognition with SVM and HOG Features

A successful implementation of a face recognition algorithm developed by eInfochips' engineers for an access management application has been discussed by Thaware [56]. In line with authors and their project, there are two phases in such a system: Face Detection followed by Face Recognition. It starts with detecting face using a Haarcascade algorithm (Chapter 3.4.1) on an image in conjunction with the cropping of the prime section of the face. A geometric face model is formed with the detection of eyes and nose performed using the Haar Cascade Classifier. The author then uses the HOG algorithm (Chapter 3.4.2) to extract features from large numbers of facial images to be used as part of the recognition mechanism. The authors have used a complicated but very useful methodology of refining the face detection data using different algorithms and then using it finally for face recognition.

2.4 Age and Gender Classification using Convolutional Neural Networks

Hassner et al. aimed at presenting the Deep Convolutional Neural classification (CNN) algorithm in combination with simple network architecture and newly released audience benchmark for age and gender classification of unfiltered face images to perform automatic age and gender classification in comparison to other state-of-art methods [32]. The experiments were carried out on a reduced set of parameters to avoid over-fitting due to the limitation of the limited labeled data. The author states that in the past many age and gender classification machine learning algorithms were employed which were heavily dependent on facial dimensions and other age constraints but could not provide highly accurate and satisfying results, especially when dealing with a tremendous amount of non-labeled images and data available through the Internet. The paper attempts to minimize the gap between both approaches, face recognition and age and gender classification using CNN. The author has summarized multiple age and gender classification methods and also provides a clear understanding of CNN and its implementation which makes the paper more practical. As a conclusion, it is suggested that room for further improvement is available using advance elaborate systems and large training data set.

Chapter III

THEORETICAL FRAMEWORK

The Chapter aims at discussing the theoretical concepts used in the course of this project. Section 3.2 is based on a discussion of metrics and measurements used in gaze recognition, a list of companies working in the domain and a few limitations of eye-tracking. Section 3.3 elaborates webcam-based eye tracking using the latest research articles and projects. We then discuss the concepts used in face detection and recognition with an explanation of haarcascading technique, the histogram of oriented gradients based recognition, and artificial and convolutional neural networks. The last Section 3.6 explains the metrics of pose angles and Wollaston illusion.

3.1 Background of Study

In this project, an intelligent, multimodal display system is to be developed. The following scenario is conceivable as a basis: A screen is located in a publicly accessible space, which can be located inside buildings, but also outside. Also conceivable is a screen behind a shop window facade, which separates the user from the screen by a pane. In any case, we assume that the viewer or user of the screen does not indicate that and how he or she can interact with the display. Any possibility of interaction must arise for the user during viewing. There are multiple possibilities for interaction, but a few are mentioned below:

- interaction via face recognition and eye movement
- interaction through gestures
- interaction via smartphone

For each of the three interactions, there are exemplary and prototypical installations in the field of digital signage; wizAI itself has made contributions in the field of gesture recognition in research [9], but also product development. Also in the field of facial

recognition, there has been researching and development work at wizAI in the past [51]. In the following descriptions, we will also go into this preliminary work.

3.2 Eye Tracker based Gaze Recognition

3.2.1 Metrics and Measures

In this section, we define a few example measures and definitions to be used during this thesis.

Fixation :

Time taken by the eye to read information is known as fixation. It refers to gathering information by being in a relatively still position. It is an indicator for the user to screen engagement. The measures include the duration of fixations, number of fixations, etc.

Dwell :

Time spent by the eye in a specific area of interest is known as dwell time. It is used to measure attention among a set of targets. A dwell can have multiple fixations. The measures comprise of dwell time and the number of dwells[28].

Saccade :

Quick movements between fixations to relocate focus from one position to another on the screen is known as a saccade. There is no visual information gathered during a saccade. The measures include saccadic length, amplitude, duration, latency, and velocity, etc.

Some other movements such as pupil dilation, smooth pursuit, blink, scan path, microsaccades, post-saccadic oscillations can also be read for understanding [28].

3.2.2 Stakeholders and Innovations

Eye-tracking has been a domain of interest for companies since the inception of artificial intelligence. Different researches have been conducted and products have been developed in the specified area. Eye-tracking has been successfully used in market research, usability testing, human attention investigation, etc. Furthermore, products are also developed for physically disabled or aged people for ease of use and communication with the latest technology such as computers. A recent project, named MAMEM [20] has been successfully implemented and further research is in progress under the umbrella of Horizon 2020, The EU framework programme for research and innovation. A large number of universities and companies have also added to its exponential growth of eye tracking, some of which are listed below.

- **Pupil Labs:** Located in Berlin, the company is run by a team of experts lead by CEO Moritz Kassner. The company revolves around a product named as Pupil

Core - a platform consisting of a software and an eye-tracking headset. Users can view and record real-time gaze and pupil data for gaze estimation, pupillometry and egocentric vision research.[27]
website: https://pupil-labs.com/

- **Affectiva:** The company was founded by Rana EL Kaliouby and Rosalind Picard at the MIT Labs and is backed by investors including Horizon Ventures. The company is working in automotive AI, media analytics and biometric solutions. A facial expression analysis algorithm is also developed to analyze image pixels and classify facial expressions [25]. A combination of facial expressions is then mapped to emotions. The emotion metrics measurement includes anger, disgust, fear, joy, sadness, contempt, and surprise.
website: https://www.affectiva.com/

- **Gaze Point:** Co-founded by Craig Hennessey, Johnny Tam, and Julie Robillard, the company is providing consumer-priced, high-performance eye trackers to clients. The goal is to envision the world of human-computer interaction by providing fast paced support to large-scale research and independently-led innovators.
website: https://www.gazept.com/

- **Smart Eye:** The company revolves around two different products. An automotive solutions module with a Driver Monitoring System (DMS) and an Interior Sensing algorithm to be used in vehicles. The company also provides research instruments for human behavior analytics and eye-tracking.
website: https://smarteye.se/

- **Eye Tribe [Eye Tracker]:** With a vision to make eye tracking available to everyone at an affordable price, the company was founded in Copenhagen by four PhD's in 2011. They created software that enables eye control on mobile devices. The user can navigate through websites, use applications and play games using the tracker without even using their hands. The company also produces eye tracking devices to allow software developers to create solutions based on their ideas.
website: https://theeyetribe.com/

- **Tobii [Eye Tracker]:** Founded in 2011, the Sweden based company also works with eye-tracking solutions and services. The company is providing hardware and software solutions in multiple domains including psychology and neuroscience, infant and child research, marketing and consumer research, professional performance, user experience, and interaction, sports performance and research, education, and clinical research.[57]
website: https://www.tobii.com/

3.2.3 Limitations

There are a few limitations to the eye-tracking technologies. There is little possibility to deduce cognitive processes from eye movements[28]. A certain user might have many cognitive processes while looking at an onScreen. Therefore sometimes it can be inaccurate to estimate information gathered via eye movements. Put differently, it is not possible to measure why a certain user looked at a specific element and what is the user's feeling during interacting with an onScreen.

3.3 Webcam based Gaze Recognition

One way to record user attention without an eye tracker is to create an algorithm that uses normal webcams to capture eye movements. The model is developed to create a mapping between features of the eyes and its corresponding position on a screen. A real-time gazer functionality with a webcam can be implemented using pure Javascript. The approach is best suited for our project as no additional specialized equipment (eye-tracker) needs to be purchased, while sacrificing the exponential accuracy factor. The created algorithm can have multidimensional usage, from basic information retrieval in an onScreen presentation set up at a university to information retrieval on any webpage searched on the internet, therefore gathering and analyzing user search behavior and applying that information to improve our advertisement patterns. The algorithm can also be enhanced to capture user clicks or touch events in a digital signage application and blend them with the eye movements to extract meaningful information. Huang et al. [22] discussed the importance of users click on a webpage. Studies have shown that a user will first look at a specified target before clicking it. Thus user eye patterns can also have a weight-age of click/touch events in an onScreen presentation.
Buscher et al [2] presented an eye tracking study of advertisements quality in web search. A highly cited study investigated search behavior using web advertisements can also be applied on an onScreen presentation involving digital signage. In chapter 7, we focus on the evaluation of our webcam based gazed algorithm solely, and then a pipeline combination with an eye tracker to calculate accuracy. The studies are conducted using a pre-compiled ground truth with participant students from our university.
Hansen et al. [14] provides an in-depth review of various models related to gaze in their paper. There are primarily two kinds of models for video-oculography, one model is to estimate the gaze point while other detects the pupil centers [18]. A third model to estimate the head pose is also applied in the context of this thesis. Our library therefore consist of two key components as defined in chapter 4, a pupil detector combined with an eye detection library and a gaze estimator which uses regression analysis learnt by the

user interaction.
For pupil detection, shape and appearance based models are distinguished [14].
Furthermore, machine learning models can also be applied for facial features detection such as support vector machines.
Several Open Source project attempts were made to perform gaze recognition based on webcams, modified webcams, and depth cameras. The distinguishing factor was based on features of full face, head or individual eyes. A comparison of such libraries is shown in Table 3.3 :

Software and Author	Input	Tracks	License
deepgaze	not available	head	MIT
eyeLike	Face	eyes	MIT
gazr	Face	head	MIT
iTracker	Face, eyes, mask	gaze	Research-only
OpenGazer	Face	gaze	GPL 2.0
webcam-eyetracker	Infrared eye	eyes	GPL 3.0
Webgazer.js	Face	gaze	GPL 3.0
Gaze	Face	eyes	MIT

Table 3.1: Comparison of open source gaze tracking software [18]

3.4 Face Detection

3.4.1 Haar Cascade Features Matching

Face detection and recognition are the most sought-after technologies in machine learning having broadened applications from a surveillance security system to consumer experience, and advertisement, Face detection can be used for personal identification, perception of emotional expressions, perception of intention and attention, perception of age, gender and ethnical race to name a few. Face detection is a substantial part of face recognition operations and uses computational resources to identify the face on the section of any given image. A variability present across head pose, orientation, rotation, expressions, image resolution, and numerous possible illumination makes it a complex method.
The earlier phase of the 20th century saw impressive progress in the field of object detection with the revolutionary advancements made by Paul Viola and Michael Jones in 2001 [59]. The algorithms created for robust real-time object detection were distinguished by these key contributions:

- Haar Features

- Integral Image
- AdaBoost
- Cascading

3.4.1.1 Haar Features

A set of digital image features having a sequence of rectangular or square shape functions are known as Haar Features. The Haar wavelets or Haar Features were first proposed by Alfred Haar in 1909. The face detection algorithm is largely composed of edge features, line features, and four rectangle features to name a few as shown in the figure below:

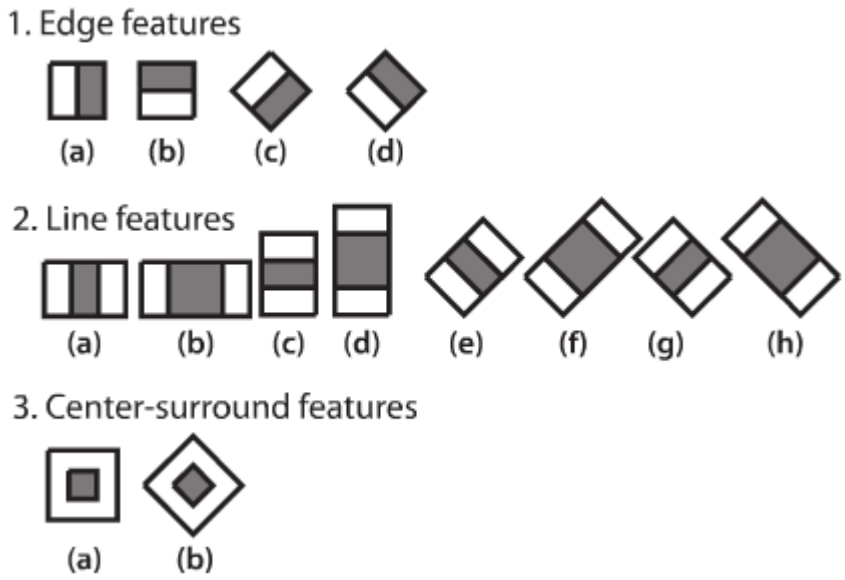


Figure 3.1: Common Haar-features [39]

These Haar features are compared with relevant features in the human face like eyes, eyebrows, nose, and lips and detect if the given image matches the precompiled face features. It considers adjacent rectangular regions at a particular position in a detection window, sums up pixel intensities in each region and calculates the difference between these sums. The approach states that the sum of pixels in the darker region will always be proportionally greater than the sum of pixels in the lighter region.

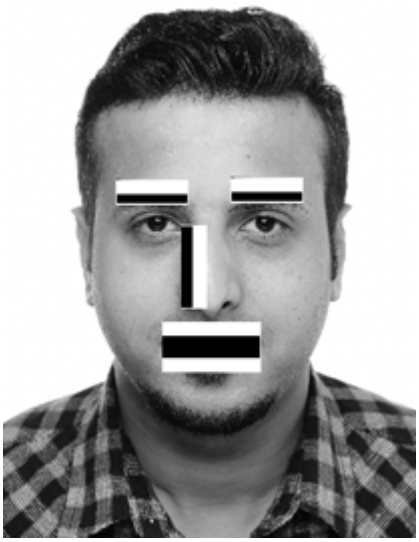


Figure 3.2: Haar-Features implementation

3.4.1.2 Integral Image

The paper [59] contributed to image representation called an integral image that allows for very fast feature representation. The integral image is used in face detection for the rapid computation using the above-mentioned rectangle features. The use of integral images is used for the fast calculation of pixel sums in a section of the picture. The procedure for the fast calculation of pixel sums was first developed by Franklin C. Crow [6]. Thus formulating that an integral image consists of sums of the previous pixel values which are stored within a stretched rectangle. Each point is the sum of all pixels within a rectangle. The integral image at a location (x, y) contains the sum of the pixels above and to the left of (x, y) inclusive.

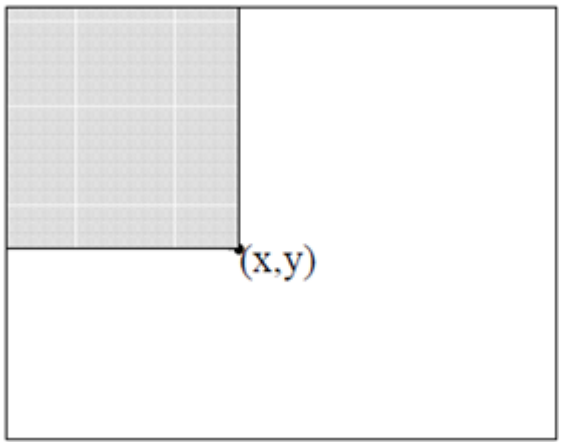


Figure 3.3: The value of the integral image at point (x,y) is the sum of all the pixels above and to the left. [59]

whereas according to Figure 3.3,

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{3.1}$$

- ii (x, y) is the integral image
- i (x, y) is the original image.

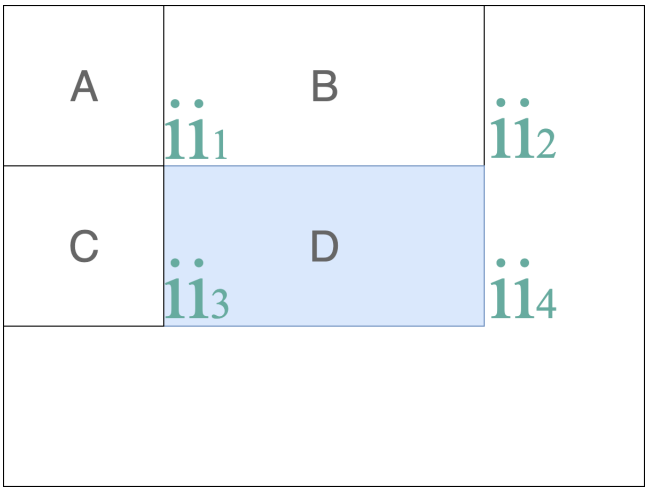


Figure 3.4: Four Array Reference
The sum of the pixels within rectangle D can be computed with four array references.

$$\begin{aligned} \text{ii}_1 &= \text{area(A)} \\ \text{ii}_2 &= \text{area(A + B)} \\ \text{ii}_3 &= \text{area(A + C)} \\ \text{ii}_4 &= \text{area(A + B + C + D)} \end{aligned} \tag{3.2}$$

Let ii_1, ii_2, ii_3, ii_4 be the values of the integral image at the corners of a rectangle. Then the sum of original image values within the rectangle can be computed:

$$\text{area}(D) = ii_4 + ii_1 - (ii_2 + ii_3) \tag{3.3}$$

3.4.1.3 AdaBoost

AdaBoost is used to boost the classification performance of a simple learning algorithm. For our face detection, a variant of AdaBoost is used to select visual features and train out classifier thus formulating a collection of weak classifiers to form exceedingly strong classifiers [8]. Using AdaBoost, Freund and Schapire proved that after some rounds, the

training error of the above formulated strong classifier approaches to exponentially zero without a loss of accuracy.
A pictorial representation of combining weak classifiers $h_i \in \{-1, 1\}$ linearly to build a strong classifier H is given below :

$$H = \text{sign} \sum_{i=1}^{N} w_i h_i \tag{3.4}$$

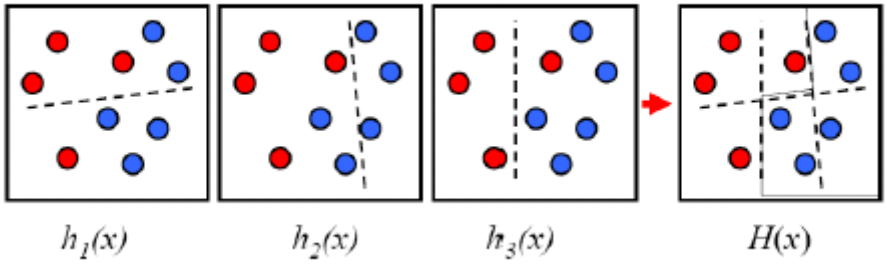


Figure 3.5: Strong classifier creation using AdaBoost.

3.4.1.4 Cascading

The combination of complex classifiers in cascade for face detection with decreased computations is then applied. The image is passed through a set of cascaded classifiers where each upcoming classifier has more precise feature information. The first layer is a single feature-based classifier that detects all positive samples with a 100% detection rate but includes half false positives. The second layer is a multi-feature classifier that focuses on false positives from the first layer and achieves a relatively low false-positive rate. The resultant can then be transferred to upcoming complex feature classifiers to achieve a cumulatively small false positive rate.
Therefore a reduced false positive rate and a decreased detection rate with each subsequent layer are achieved. The resultant target is achieved by adding more features to the corresponding layer and adding layers until the final target of false detections is met.

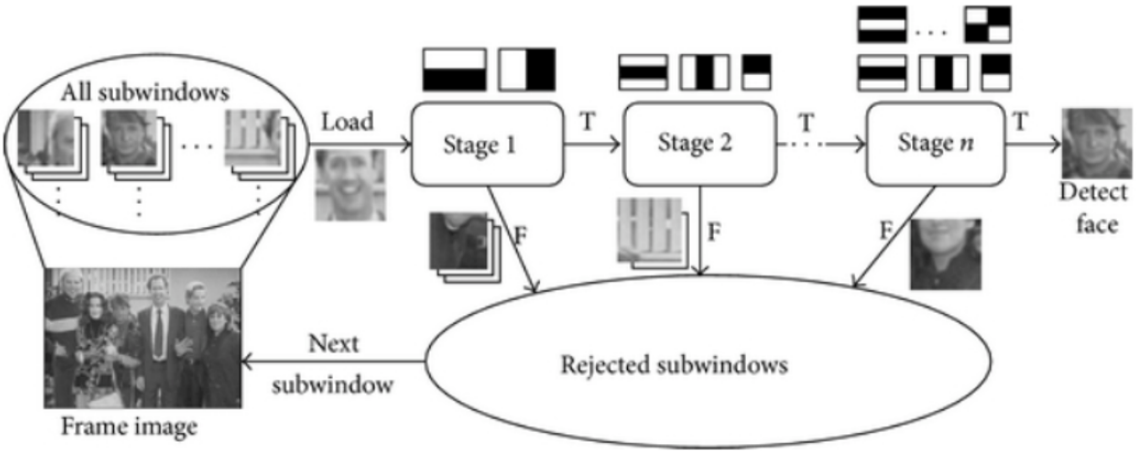


Figure 3.6: Cascade classifier representation.[43]

3.4.2 Histogram of Oriented Gradients

Another important human detection framework was set up by Navneet Dalal and Bill Triggs at the Annual International Conference on Computer Vision in 2005. The idea was to create a robust feature set to allow algorithms to discriminate human forms under conditions corresponding to cluttered backgrounds and difficult illumination [24]. The concept was coined as Histogram of Oriented Gradients (HOG). The essential thought behind the HOG descriptor is that native object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions [7].
In our face detection, we can use the power for HOG features to determine human face detection, use the sliding window classifier to obtain a descriptor set. Then we can adopt the integral image representation and a cascade or acceptance and rejection windows to detect if a human face is present or not. There is a lot of relevant work done in this domain and notable papers on face detection are mentioned accordingly. [47, 15]
While HOG provides a more practical approach if the intention is to collect quantitative data, an Artificial Neural Network is always a better option for quantitative analysis. HOG uses the SVM (Support Vector Machine) algorithm which is a discriminative classifier and finds the optimal categorization for data. [46]

3.5 Face Recognition

3.5.1 Artificial Neural Network

Artificial neural networks (ANN), also called neural networks, are machine learning models inspired by the human brain's biological neural network. The human brain comprises of approximately 86 billion nerve cells named neurons which represent the most basic computational unit that combines to form a complex interconnected network, see Figure 3.7. Neurons form the brain's neural network that passes electrical or electrochemical signals to process information.

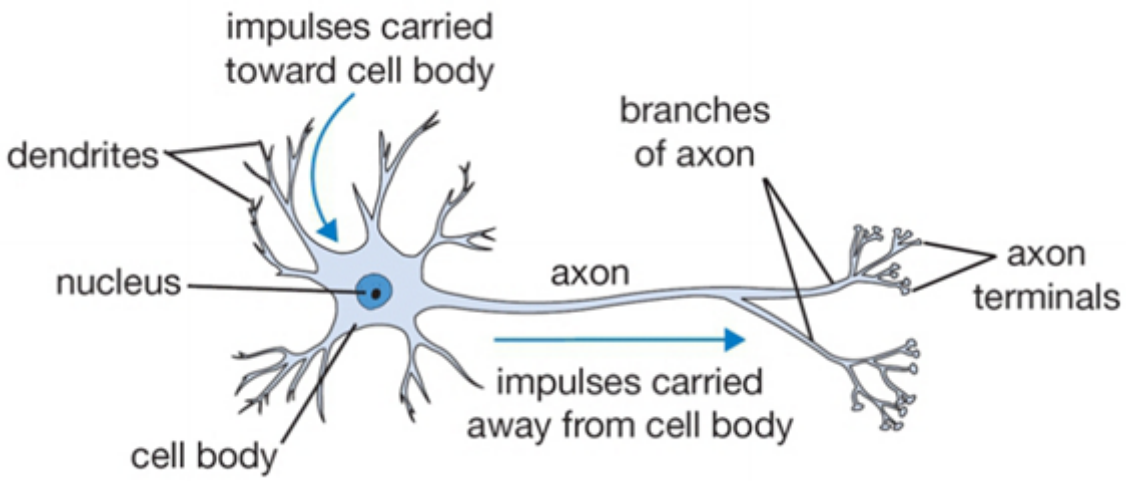


Figure 3.7: Schematic drawing of a biological neuron

Although the state-of-the-art ANNs have never reached the full functional complexity and computational power of their corresponding biological examples. Instead, different approaches are designed to model certain properties in the best ways possible. For instance, spiking neural networks [10] includes a dynamic component in their operating principle as the time between the firing of neurons affects their impact. In contrast, recurrent neural networks [17] are well suited to process correlated input sequences due to their memory capabilities. Convolutional networks described hereafter, are designed to perform well on grid-like structured input.

3.5.1.1 Perceptron

Frank Rosenblatt in 1958, introduced the idea of modeling human brain activities by using a single-layer network: the perceptron [48]. The perceptron can learn its weights to categorize the input information which can be thought of as a very simple learnable classifier. Basic structure of a perceptron has been shown in Figure 3.8

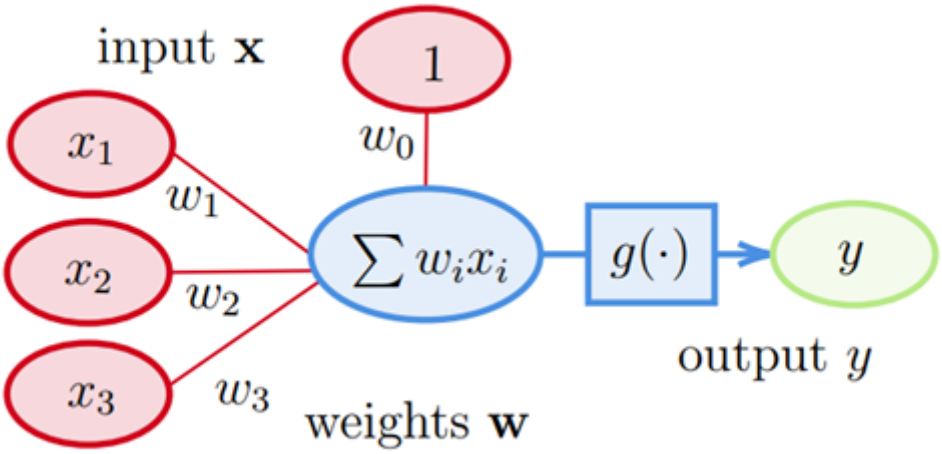


Figure 3.8: Representation of a perceptron.
The elements xi of an input x are individually multiplied by the corresponding weights wi and summed along with the perceptron's bias w0. Subsequently, the weighted sum is processed by an activation function g(•), creating the output y.

As shown in Figure 3.8, every input unit is assigned an individual trainable weight. Moreover, a constant bias is also learned by perceptron. The sum of weighted input and a bias weight is represented as an activation "ax".

$$a_{\mathbf{x}} = \sum_{i=1} w_i x_i + w_0 \tag{3.5}$$

Perceptron's output y is obtained by passing activation ax through a non-linear activation function g.

$$y = g\left(a_x\right) \tag{3.6}$$

3.5.1.2 Multi-layer Perceptron

A single-layered perceptron can be extended to a neural network by stacking multiple layers of neurons together. The output of a neuron is passed as input to another neuron in the subsequent layer. Therefore resulting architecture is referred to as a multi-layer perceptron, often abbreviated as MLP or feed-forward neural network. Neurons can be arranged in any number of layers and every layer can consist of any number of neurons. The model's depth defines the length of the connected chain of layers. The term deep learning refers to a neural network that is deep enough to learn a feature hierarchy within the hidden layers [13]. A simple MLP which maps an input x to an output y is shown in Figure 3.9.

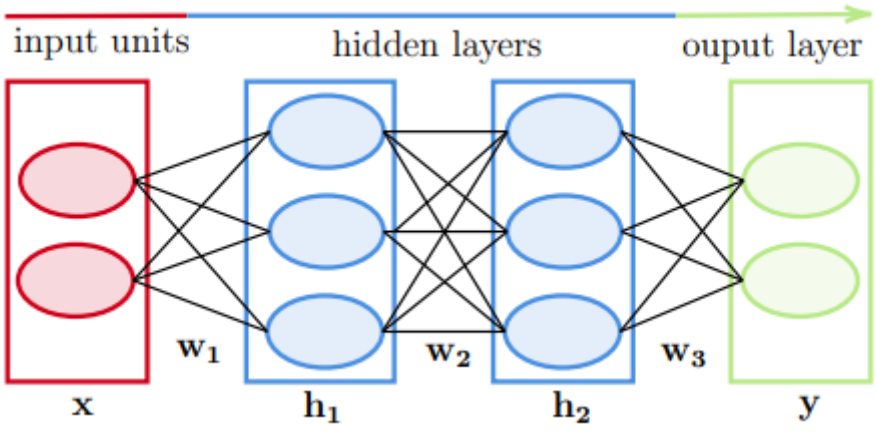


Figure 3.9: Representation of a Multilayer perceptron.
The information x is sent out of the input unit, processed by two hidden layers and finally passed to the output layer, where it is processed once more leading to the output y.

The stacking multiple non-linear units enables the network to model complex systems.

Activation Functions

The weighted sum is a linear combination of the input units. To model any complex function it is necessary to allow for non-linearity. Therefore, an activation function g(•) is used to act on the weighted input sum. Theoretically, the activation function could be of any shape. Three of the most popular activation functions are the hyperbolic tangent function, the sigmoid function, and the rectified linear function, or ReLU.

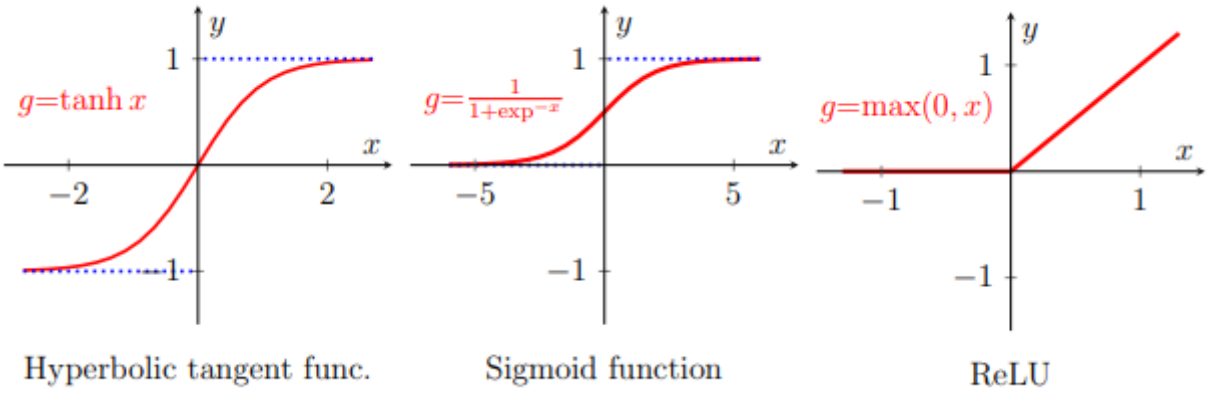


Figure 3.10: Frequently used activation functions.

The most commonly recommended activation function for deep neural networks is the ReLU function, see also [11]. ReLU exhibits necessary characteristics of a nonlinear

activation function, it is still close to being linear. Hence, an optimization with a gradient-based method is comparatively faster and easier. The mathematical definition of ReLU shows that the function consists of two subsections.

$$g(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Since multiplication of linear functions remains linear, a deep neural network without any activation functions would simply learn a linear function. Hence, as soon as activations are included, a multi-layered network can learn a non-linear approximation of a model. Deep Neural networks are considered to be general function estimators that are cable of approximating any kind of complex model.

Learning Optimization

A simple neural network accepts an input x and transforms it to an output ˆy. For this purpose, information is passed through the network which is called forward propagation or alternatively forward pass. Learning is achieved by adopting the weights such that the difference between neural networks prediction ˆy and the corresponding ground truth y is minimized. The cost function, which quantifies the mismatch between prediction ˆy and ground truth y, is called loss function or objective function. The optimizer aims to minimize loss function during the learning process. A common objective function in many machine learning applications is the mean squared error (MSE) between prediction and ground truth:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.8}$$

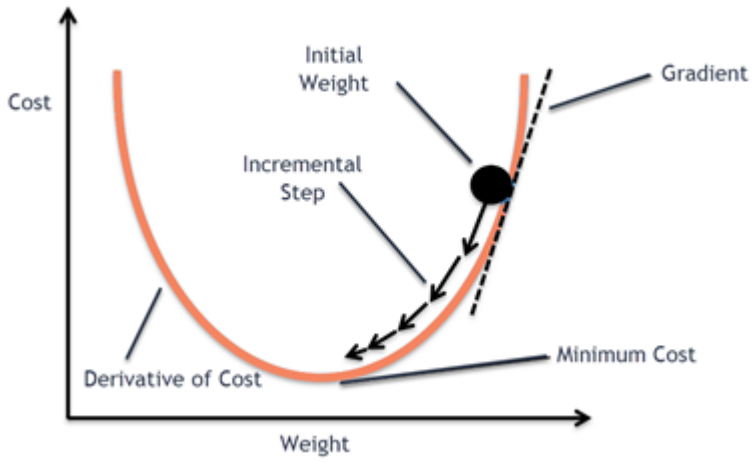


Figure 3.11: Gradient Descent.

Compared to an objective function based on the mean absolute error, it is more sensitive to outliers. Especially in classification settings, a cross entropy loss often yields faster and better results as a model trained with mean squared error gets stuck quicker in local minima [12]. In order to minimize the objective function algorithms such as gradient descent algorithm [13] is used. It is necessary to know how much the loss function depends on every single trainable weight to be able to individually optimize them. The back propagation algorithm [49] is a method to calculate the gradients for all weights in a neural network. It is based on the chain-rule of calculus:

$$\frac{da}{db} = \frac{da}{dc}\frac{dc}{db} \tag{3.9}$$

Given the computational graph of a neural network, the back-propagation algorithm recursively applies the chain-rule to obtain the gradient with respect to a single weight. The classical gradient descent algorithm (GD) calculates the gradient with respect to a whole training set. Considering the non-convex nature of optimization of deep neural networks, this procedure may not yield to the best result. Additionally, it is not feasible to first process all the training data before making the first weight update. Stochastic gradient descent (SGD) [13] is an incremental version of GD whereby only a randomly sampled subset is taken to calculate the gradients. The gradient noise, introduced by taking only a subset of training data into account, helps the algorithm to jump out of local minima. Another important optimization algorithm is Adam, named after adaptive moments [13].Possessing an adaptive learning rate, it is considered to be an extension to SGD. Adam adjusts the learning rate according to the gradients' first and second momentum. In essence, we want to achieve to the minimum possible value by descending and moving towards the minima of the objective/loss function as shown in the Figure 3.11.

3.5.1.3 Convolutional Neural Networks

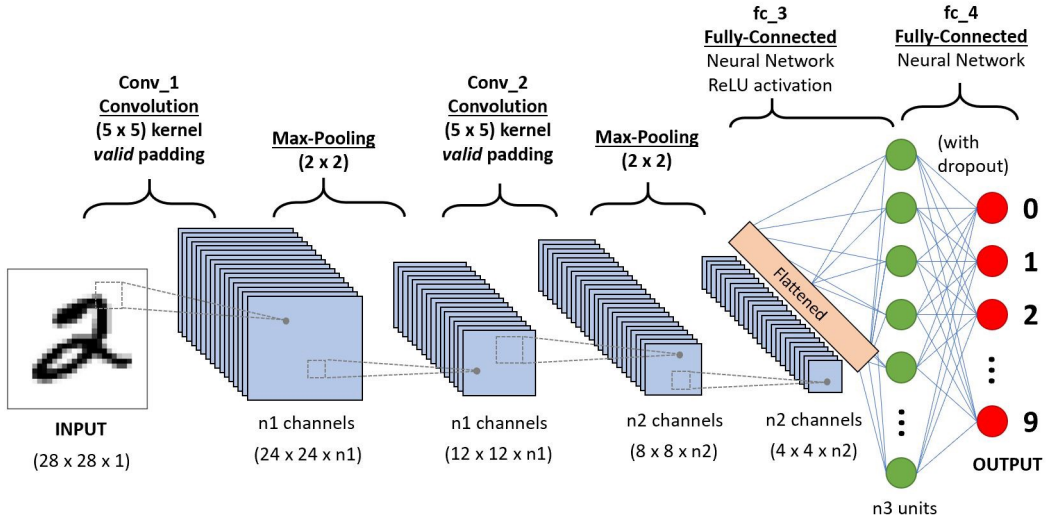


Figure 3.12: A Convolutional Neural Network[50]

can process input in the form of a matrix. This makes it possible to use images represented as a matrix (width × height × color channels) as input. A CNN consists essentially of filters (Convolutional Layer) and aggregation layers (Pooling Layer), which repeats alternately, and at the end of one or more layers of "normal" fully connected neurons (Dense / Fully Connected Layer).

Convolutional neural networks [29], also known as convolutional networks or CNNs, are a variation of MLPs, designed to process grid-like structured data. In a Convolutional Neural Network, the input information x, now structured in an array, is filtered by two convolution layers and passed to the output layer y, increasing the channel-size according to the number of filters per layer.$\mathbf{W_i}$ represents the filters' weights.
In a neural network, each layer consists of a three dimensional array of size h × w × c . While h and w represent the spatial information height and width the third dimension c defines the layer's amount of channels. In common semantic segmentation architectures, the channel dimension increases with the layer's depth while simultaneously the spatial dimensions are decreasing, which is visualized in Figure 3.12.
A CNN consists of different layers which may include: convolution layers, pooling layers or fully connected layers. Where fully connected layers are similar to multi-layer perceptron. The last fully connected layer outputs the network's prediction.

The Convolution Operator

In mathematics, convolution is an operation on two functions, x(t) and y(t), which combines two functions and generates new function. Whereas, one function is mirrored and

subsequently translated, pointwise product of both functions is calculated and accumulated over time or space.

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{3.10}$$

The resulting function can be considered as a cross-correlation or a similarity measure.

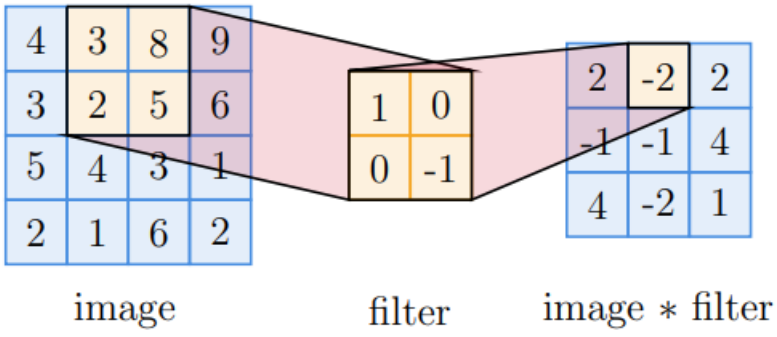


Figure 3.13: Two Dimensional Convolution
Visualization of a 2×2 convolution filter is applied on a two dimensional array.

A convolutional layer applies convolutions with two-dimensional or three-dimensional filters on the input data. Additionally, convolution's output is passed through a non-linear activation function such as ReLU as defined above. Multiple filters can be applied in one layer, all of which produce particular output channels which are then passed as an input to the subsequent layer.

Pooling Layer

A pooling layer aggregates the results of convolutional layers by passing only the strongest signal. For instance, a pooling layer simply samples the highest value of a kernel matrix and discards the rest. The four matrix results created by a 2×2 kernel are reduced to just one number which is the highest among all. The purpose of pooling is to pass on only the most relevant signals to the next layers, to achieve a more abstract representation of the content and to reduce the number of parameters of a network.
The pooling layer is fully defined by its stride, its kernel size and the pooling function. A common pooling method is called MaxPooling show in in Figure 3.14(a). The advantage of concentrating on the maximum values is that focus lies solely on strong activations where a strong activation indicates a close match between the input and the learned filter representation.

Figure 3.14: Representation of Pooling
In each case, the pooling was done with stride $s = 2$ and a kernel size of 2×2. In both cases, the left array represents the pooling layer's input and the right array the pooling layer's output.

An alternative to maxpooling is mean pooling which keeps and processes even low activations would be mean pooling as illustrated in Figure 3.14(b). A mean pooling layer outputs the average value of all filtered pixels. Hence, it retains more information about the neighborhood. Both presented approaches have individual advantages. However, experiments have shown that max pooling is usually more likely to yield better results in combination with CNNs [53].

The Fully Connected / Dense Layer

The Fully Connected Layer or Dense Layer is a MLP or a simple neural network structure in which all neurons are connected to all inputs and all outputs. To be able to feed the matrix output of the convolutional and pooling layers into a dense layer, it must first be rolled out or flattened.

This flattened information is thus fed into one or more fully connected layers and connected to an output layer which has exactly the number of neurons corresponding to the number of different classes to be recognized.

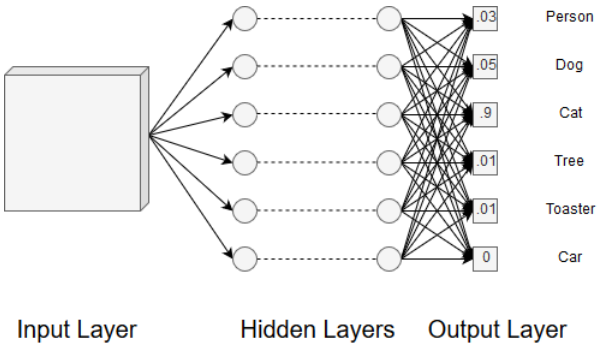


Figure 3.15: A Fully Connected Layer

3.6 Head Position Estimation

The relative orientation and position with respect to a camera depicts the pose of an object. Pose can be changed by either moving the camera with respect to the object, or the object with respect to the camera [36].A head pose estimation can have multidimensional applications. In scope of our project, we plan to gather areas of interest of user on a screen, and validate our eye tracker and camera based gaze algorithm defined in chapter 3.2. A robustness is required to overcome factors including noise, illumination, distortion and occlusion.
A large number of model approaches have been applied to estimate head pose including Appearance Template Methods, Detector Array Methods, Regression Methods, Manifold Embedding Methods, Flexible Models, Geometric Methods, Tracking and Hybrid Methods[54].
A identification of a few facial landmark points can easily estimate pose of the head. For this purpose, the 2D coordinates on the image plane and its corresponding 3D coordinates are required. A 3D rigid object can then have two kinds of motions with respect to a camera known as Translation and Rotation Matrix. Both matrices have 3 DOF in a 3D space. We represent our rotation matrix using Euler angles. Euler angles are used to define the orientation of a body in accordance to a fixed coordinate system. Three Euler angles can be used to specify the three relational DOF as shown in Figure 3.16 and defined later [54].

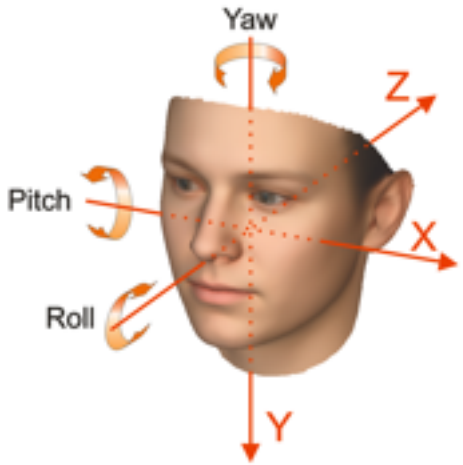


Figure 3.16: Definition of the pose angles [1]

- **Yaw angle :** rotation around the vertical (y) axis. Positive yaw angles represent faces where the person looks to his or her left side.
- **Pitch angle :**rotation around the horizontal side-to-side (x) axis. Positive pitch

angles represent faces looking up.

- **Roll angle :** rotation around the horizontal back-to-front (z) axis. Positive roll angles represent faces tilted toward the right shoulder.

In Figure 3.17, multiple head poses are shown in degree corresponding to the Euler angles defined above.

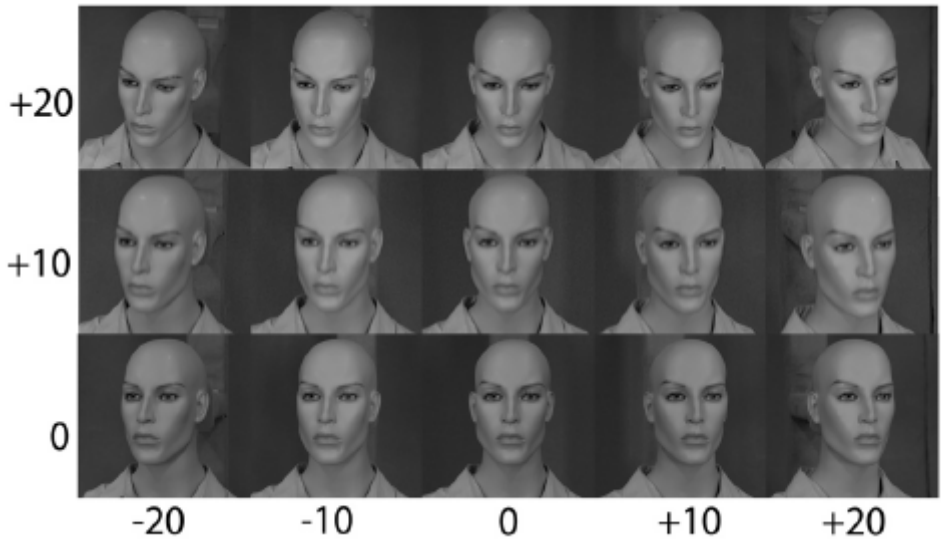


Figure 3.17: Groups of face angles in degree [58]

In the scope of our project, we clearly define that the calculated head pose will only be used for cross validation and can not be combined with gaze direction as the human gaze is not limited to movements of eyes. As shown in Figure 3.18, face and eyes can have multiple deviation angles in the same head pose spectrum.

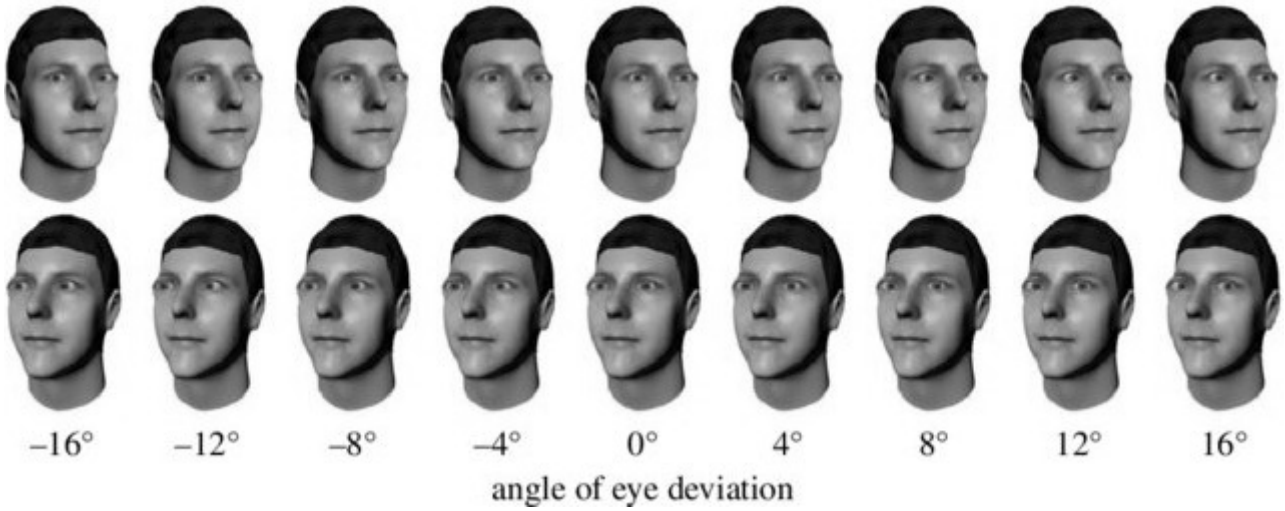


Figure 3.18: Angles of face and eye deviation in degree, Wollaston illusion

Chapter IV

METHODOLOGY

4.1 System Vision

The goal of the thesis revolves around multiple aspects. We perform eye tracking, face recognition, feature extraction, and head pose estimation to create a highly integrated and robust product to be used in the domain of targeted advertisement. The scenario can have multidimensional usage, one of them is defined below.

Faraz arrives in a Koblenz based hotel for a meeting with his professor starting the following morning. He is relaxed and starts revising the slides on his notebook. While studying, he also explores some websites on the web, one of which detects his location based on Faraz's IP address. On a blog, he looks at a German restaurant next to his location which offers musical shows and food at a low price. Faraz clicks the advertisement and reads about the offers. He leaves his room for a walk and finds an interactive display in the hall. The display is fitted with an eye tracker and a webcam, with hotel information, nearby places, offers and bus connections. There is a face detection system working with the interactive display which starts gathering information about Faraz regarding his age, gender, interests, mood, fixations on the screen, etc. The system already has his click information regarding the German restaurant and shows 4 new restaurants in nearby locations, including a Pizza Corner, Bistro, Pakistani restaurant and an Italian Cucina. Faraz wants to try Pakistani food and check for directions, menu prices, and offers. He finds a coupon in the link and heads towards the location.

In this scenario, we analyze that click-based targeted advertisements can serve both, business and the customers. However, using eye-tracking and face recognition, an enhanced product can be developed to display targeted, adaptive advertisements. Therefore the system is created to serve as a prototype for the above-mentioned scenario.

4.2 Requirements

The following topic discusses these requirements for the planned system. Requirements are further distributed into functional and non-functional requirements. The requirements are divided into mandatory, target and optional requirements. Mandatory requirements must be met, target requirements should be met unless there are serious arguments against it, and optional requirements can be met if it is feasible.

4.2.1 Functional Requirements

A system is set up in a controlled environment with the user being aware of the fact that video is recorded. Data collected via this system will be stored in a database and used for model enhancement and evaluation.

4.2.1.1 Authentication

A database of images needs to be created for the model. Therefore the user needs to take pictures with different poses for the system to perform face recognition. The pictures can be taken via webcam or users can directly apply to store pre-compiled images in our database. This functionality is only possible if a person agrees to login and use the face recognition feature. Under the non-authenticated environment, all user's data will be stored as "Unknown" or "Anonymous".

4.2.1.2 Authorization levels

Only admins will be allowed to use data for evaluation and improvements. A user should not have access to a direct database, however, a user can request for his evaluated data after using the system. The data gathered should not be sold to third parties.

4.2.1.3 Business Rules

Code in the backend will have the functionality of face detection, face recognition, gaze tracking using a specialized tracker and common webcams, and head position estimation. An interface is to be provided for user interaction with the system. The user can use all the above functionalities while an administrator can also view evaluation results.
All the functionality defined above should work in real-time with minimal inaccuracy. The functionality should have maximum confidence at a distance of 1 meter or less.
In the case of multiple users standing in front of the onScreen, face recognition must work concurrently on all users. Contrarily, head pose and gaze tracking should work only with a single user in the same scenario mentioned above.

Overall System Requirements

- The algorithm should be robust and computation should function in real time.
- The model should be trained to get max accuracy.
- The algorithm must work at a distance less than or equal to 1 meter.

Face Detection and Recognition

- Faces must be recognized with the Viola-Jones algorithm.
- Face recognition should be functional with an image input, video input or using a live webcam.
- Important characteristics of user must be extracted. We plan to implement demographic information extraction algorithm based on the same input.

Gender Detection

- The gender of a person should be determined based on the images taken during face recognition in real-time.
- Gender must be determined using an SVM.
- SVM model should be trained using different datasets.

Gaze Tracking

- Tracking should work with minimal or no calibration.
- The algorithms should work with a single user at a time.
- Pupil detection and eye feature extraction to be performed.
- Mapping of an eye to screen coordinates should be possible in real time.
- Regions on the screen should be extracted and stored in the database (Areas of Interest)
- Content Extraction to be done based on eye to screen coordinates.

Backend and Graphical User Interface

- A chrome extension needs to be developed for frontend - backend communication.
- The functionality must work even in offline mode.
- The function must support threading and multiprocessing.
- A login functionality to be developed for authenticated users to use the system.
- Real time evaluation by the system should be done and shown on the GUI for admin.

4.2.1.4 Legal or Regulatory Requirements

A user shall be informed about the system before usage. Data collection and storage must be done with consent of the user.

4.2.2 Non Functional Requirements

A list of non-functional requirements is defined below :

4.2.2.1 Development Process

- Python to be used as a backend programming language.
- Javascript to be used as a frontend programming language.
- MySQL database should be used to persist data.
- OpenCV Python to be used for vision operations.
- PyCharm can be used as a development environment.
- Web interface must be delivered by an Apache webserver
- Source code must be documented in a meaningful and comprehensible manner

4.2.2.2 Performance

- There should be a robust request-response mechanism.
- Storage and database size should not affect the performance of the system.
- A better GPU is required for the best performance.
- Multiple CPU's can be used concurrently for fast computations
- Quality of webcam can affect performance metrics.
- Data security is the sole responsibility of the IT administrator employed.

4.3 System Design for Eye Tracking

Eye Tracking refers to the process of measuring gaze relative eye position to a screen. Gaze movements are computed and coordinated on a screen for gathering data. In the scope of this thesis, we use myGaze eye tracker SDK which can measure eye positions,

eye movements, and pupil dilation. These measurements can then be used to predict the meaningful behavior of a user.
For this reason, we have used a myGaze SDK which enables us to use an Application Interface for communication between our application and the Eye Tracking server[37]. Using the device, we were able to retrieve eye tracking data of the users and control the device for our custom application. The SDK is currently supporting a number of programming languages including MATLAB, C, C++, C, Visual Basic, and Python. An API Layer overview functionality is shown in Figure 4.1.

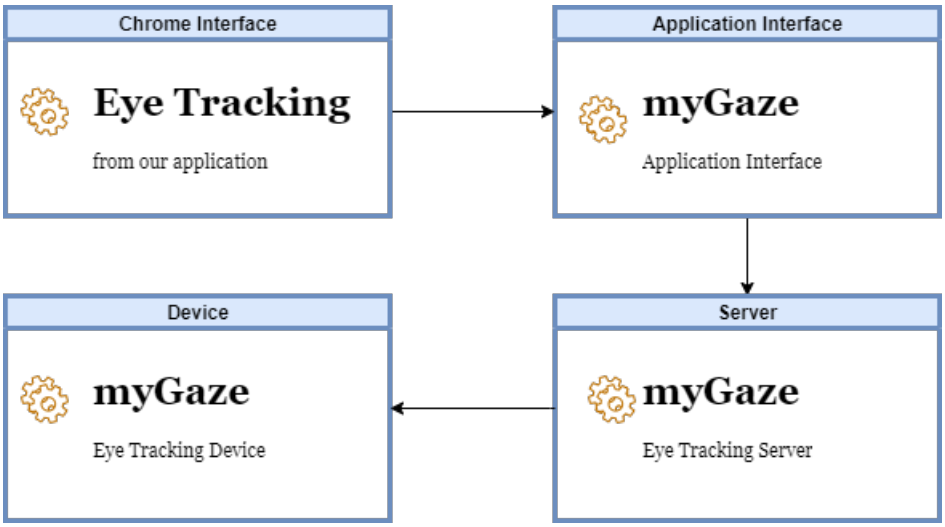


Figure 4.1: API Layers for myGaze Tracker

The above Figure defines the Application Architecture for myGazeSDK used in our project. It also shows the hardware and software components of the eye tracking system[37]. A workflow provided by myGaze SDK is shown in Figure 4.2.

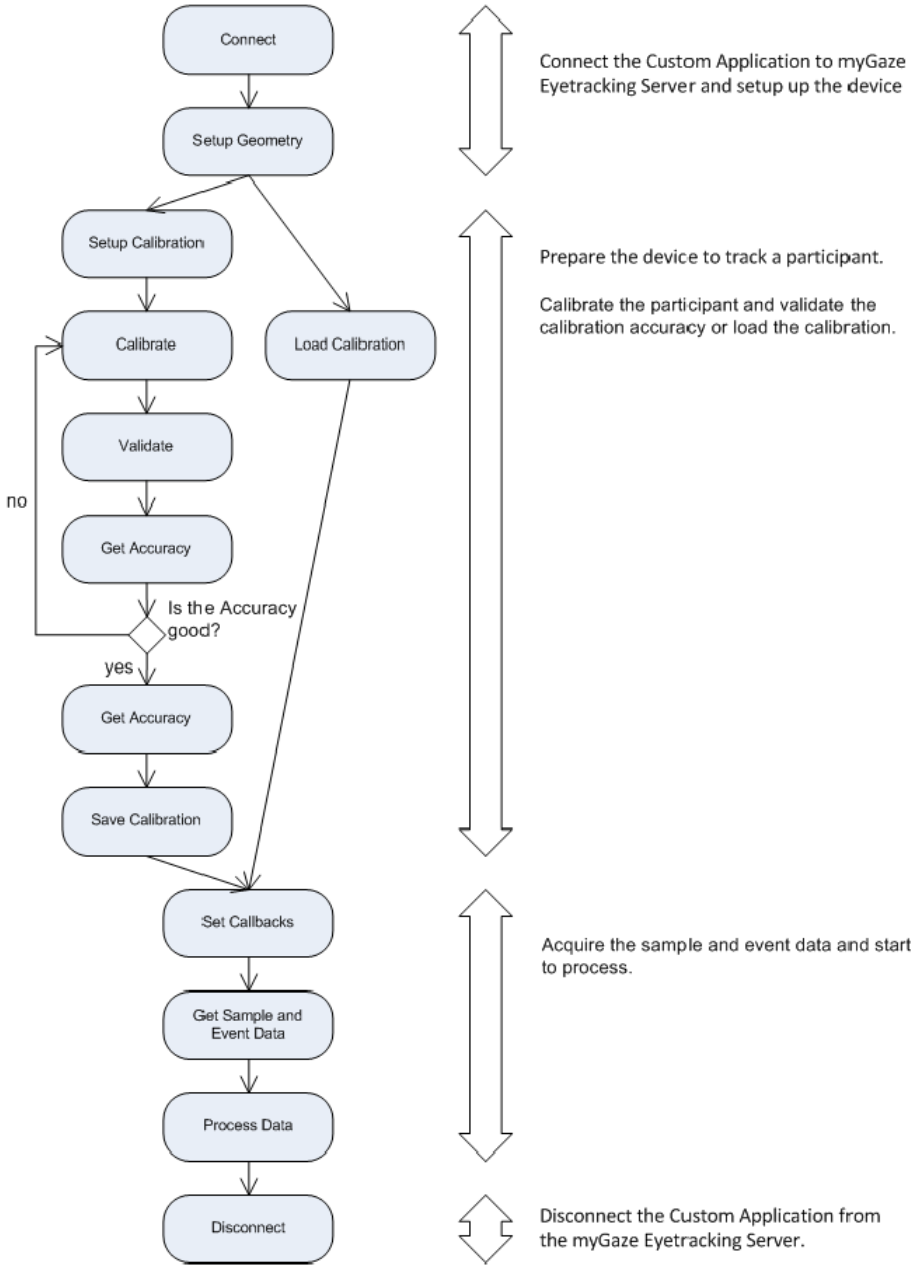


Figure 4.2: Eye Tracker Workflow [37]

All the components shown in Figure 4.1 are defined below:

Gaze Tracking - FAGCOP: This is part of the python project created by us. We use customized methods in our application created in the myGaze SDK and defined in the documentation. With these methods, we can easily extract the features of eye movements, fixations, dwell time etc. 3rd party applications can also be integrated into the same application to create better products.

In the scope of this project, we created a chrome application defined in Chapter 5.3.1, which communicated with the python server to gather data from browser and store in the database for later evaluations.

myGaze Application Interface:

It is a part of myGaze SDK which provides a programmable interface to access the eye tracking device [37].

myGaze Eye Tracking Server:

This is an eye tracking server application and is used to collect data from eye tracking device. It serves as a mid layer between eye tracking device and application interface. All the data collected is send to the application interface.

myGaze Eye Tracking Device:

An eye tracking device that can be connected to a windows operating system as defined in Chapter 5.4.1. The device needs to be connected to a USB 3.0 port, and can launch a standalone server application that runs the background.

4.4 System Design for Webcam Eye Tracking

The model for eye tracking using a webcam is created using JavaScript. The process consists of three main components.

- Pupil Detection
- Calculating Eye Features
- Features to Screen Mapping

The library can use different predictive modeling techniques like linear regression or ridge regression model. The regression model gets pupil positions, eye features and matches it to screen locations while the user is interacting with a screen. Unfortunately, there is no 3D reasoning implemented which makes it a weak predictor. The methodology comprises detecting the eyes and then returning it with enclosed triangles, thereafter performing a more realistic fitting of facial and eye contours.

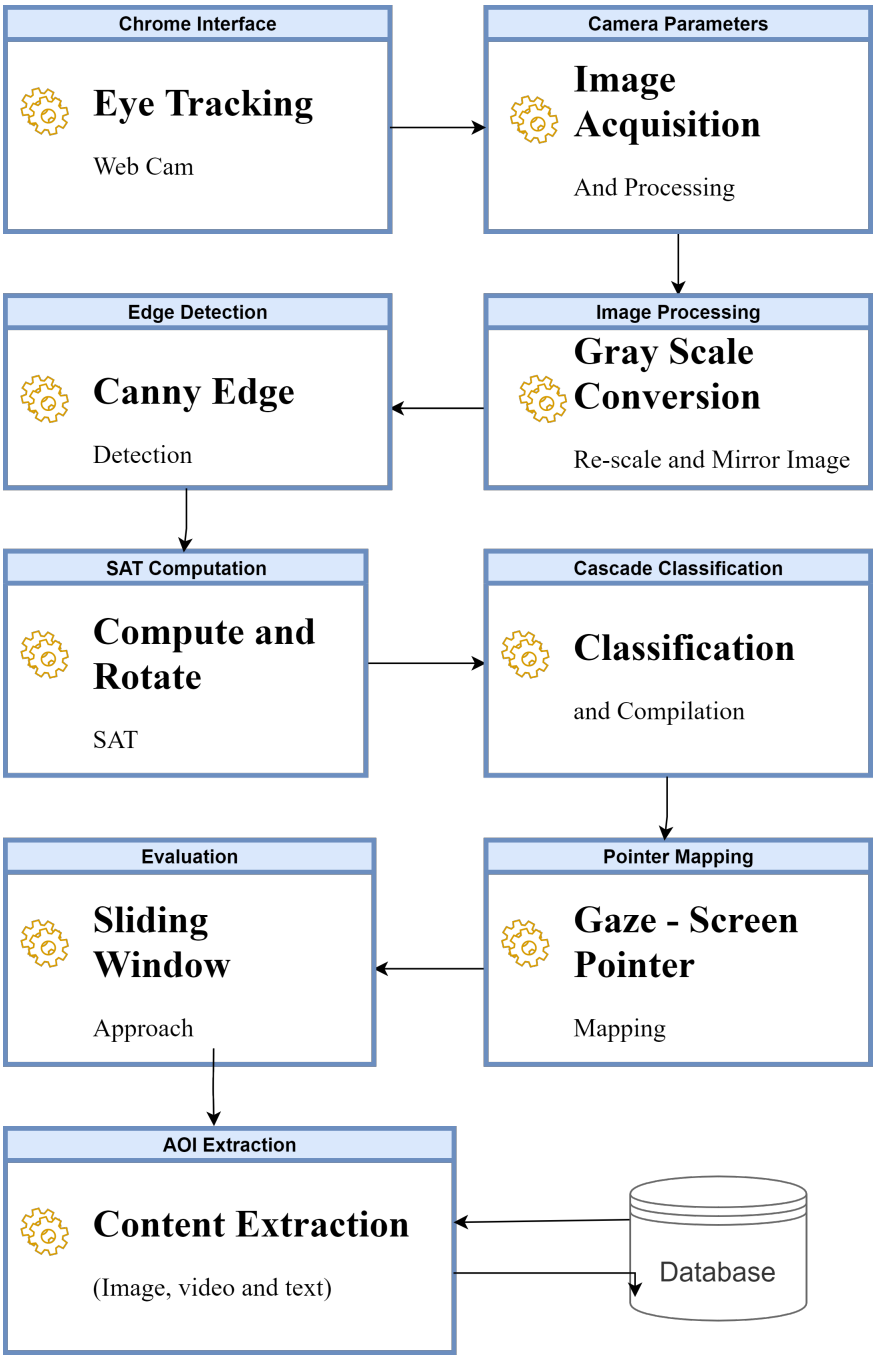


Figure 4.3: Eye Tracking Block Diagram

We explain the following topic using Figure 4.3 and simultaneously cover the components defined above. The chrome extension defined in Chapter 5.3.1 is the starting point of the application. We can start the eye tracker by going to the menu and selecting "Webcam Eye-Tracking". This sends a request to the python server which starts recording the screen. The recorded video frames are then converted to a grayscale image to remove color spacing. The image is converted to remove colors and hence decreases the complexity of

the model. After decreasing the inherent complexity of gray-level images, we perform edge detection using the Canny Edge Detection algorithm. The algorithm required the image to pass through a number of steps defined below [31]. A blur image before edge detection is shown in Figure 4.4.

Figure 4.4: Blurred Image After Acquisition

- **Gaussian Blur :** The Gaussian blur is used to remove the noise from the image before processing it.
- **Determine the Intensity Gradients :** Image processing filters can be used to determine the intensity gradients. Hereby, we use Sobel filters for to detect an edge whenever the intensity of the pixel changes based on the color of the image. Filters are applied in the vertical and horizontal windows to calculate the derivatives. Example filters are shown in Equation 4.1:

$$G_z = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} A, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & +2 & +1 \end{bmatrix} A \tag{4.1}$$

- **Double Threshold Application :** The image produced have thick edges on which we perform a non-maximum suppression. The double thresholding technique is applied to normalize the values of pixels to fall in a range of 0 to 1. Pixels with high values are termed as edges.

- **Edge Tracking by Hysteresis :** After determining the strong and weak edges, we need to filter in actual weak edges and remove the true negatives. Weak edges, connected to strong edges will be counted as edges while others will be removed. We get a final image as shown in Figure 4.5.

Figure 4.5: Final Result from Canny Edge Detection

4.4.1 Pupil Detection

Pupil detection is the core feature for webcam eye tracking. It comes with three main assumptions which can make frail predictions but are enough to get real time results with rational accuracy. The three assumptions are listen below :

- The pupil is located in the center during detection
- The iris is darker than its its neighbouring area.
- The iris is circular.

A search over all offsets with a high contrast to its surrounding region is conducted to detect pupil. The summed area table or integral image technique defined in Chapter 3.4.1 is used to make the search more efficient.

4.4.2 Eye Features

Image Pixels are then mapped to gaze locations using a regression model defined later in this topic. The two eye regions calculated are processed, resulting to a 120D features

which becomes the source for a regression model.

4.4.3 Screen Mapping

By obtaining N number of pupil locations as training examples $\mathbf{x} = (x_1, \ldots, x_N)$ and their corresponding observations of clicks (as true gaze locations), $\mathbf{t} = (D_{x1}, \ldots, D_{xN})$. Using a simple linear regression model, we obtain a function f(v) mapping to Dx, where function f(v) predicts the location of gaze on screen given a pupil feature vector v.

$$\underset{\mathbf{w}}{\text{minimize}} \sum_{x_i \in \mathbf{x}} \|D_{xi} - f(x_i)\|_2^2 \tag{4.2}$$

After the pupil coordinates are mapped to the screen, we can start gathering the content from screen based on pointer location on the screen.

4.4.4 Content Extraction :

Each element on the screen is converted in to an HTML code in the browser. Using JavaScript, we can easily provide location dimensions to each element in the code. This can then be compared with the pointer location to extract content. The content will then be transferred to our python backend, and stored in the database.

4.5 System Design for Face Recognition

Face recognition need faces to be tracked and detected first, as discussed in Chapter 3.4.1. We use the Viola-Jones haar cascade algorithm for face detection, all the faces are first aligned, then representative characteristics are extracted. An embedded file is already stored having spacial face characteristics is fetched and matched with the input face. if matching occurs, the face id is created and data is deposited to the database. The process flow is shown in the Figure below:

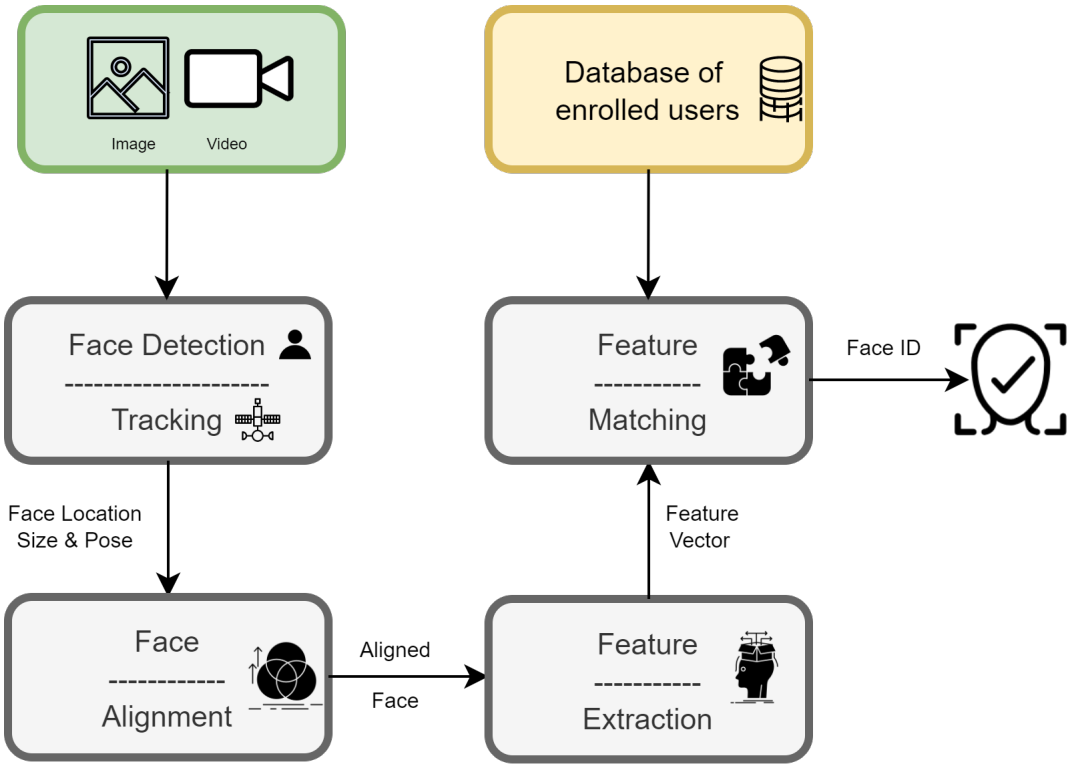


Figure 4.6: Face Recognition Processing Flow

Figure 4.7 shows the block diagram and confirms the complete process defined above. Using the chrome extension defined in Chapter 5.3.1, we can invoke the backend for the face recognition algorithm. An image is acquisition-ed and processed, the face detection and feature extraction are performed using our model and data are stored in the database.

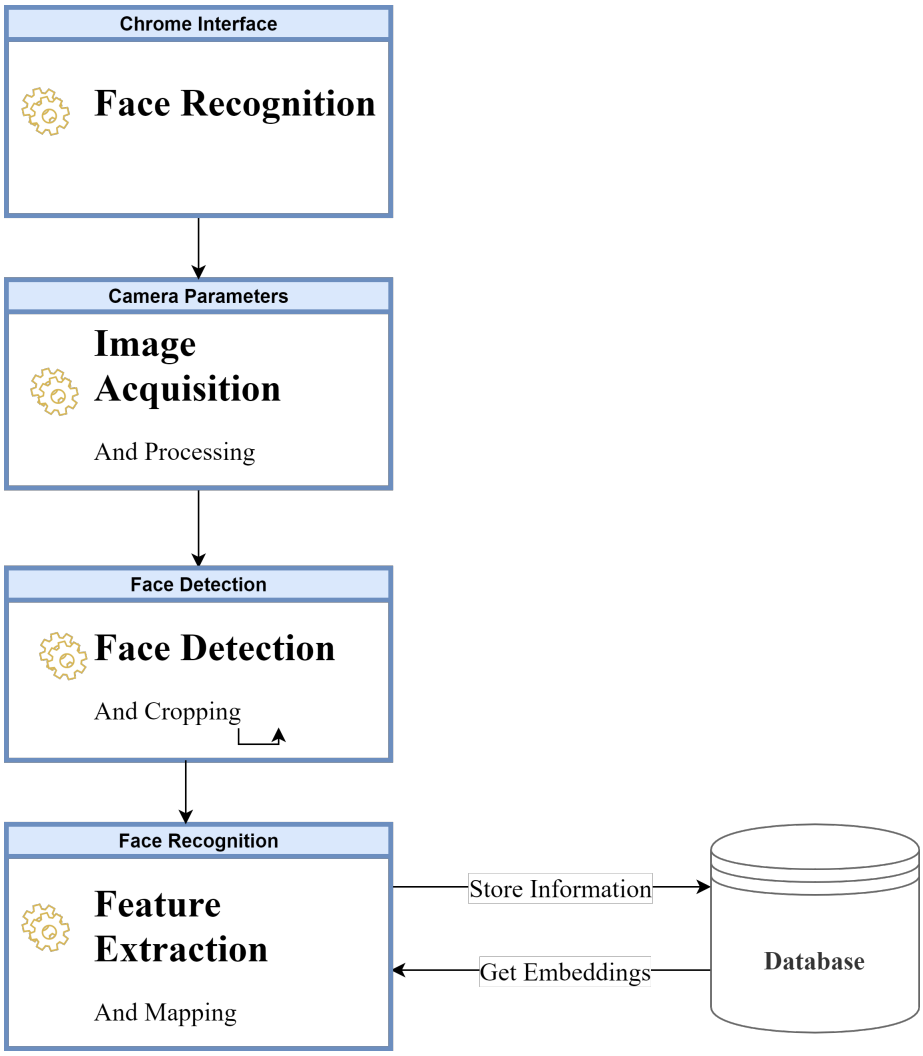


Figure 4.7: Face Recognition System Diagram

The face features mapping works in a cascade, where embedded faces are mapped with the input face. In comparison with multiple functions in a downflow model where a face is matched with face features F(f1, f2....fn) as shown in Figure 4.10. In case of no feature matching, an "Unknown" label is applied to the face otherwise the face label is shown in the camera window. Keeping in view the anonymity factor, we are currently storing no label information for an unknown user, but still, store the features and demographic information.

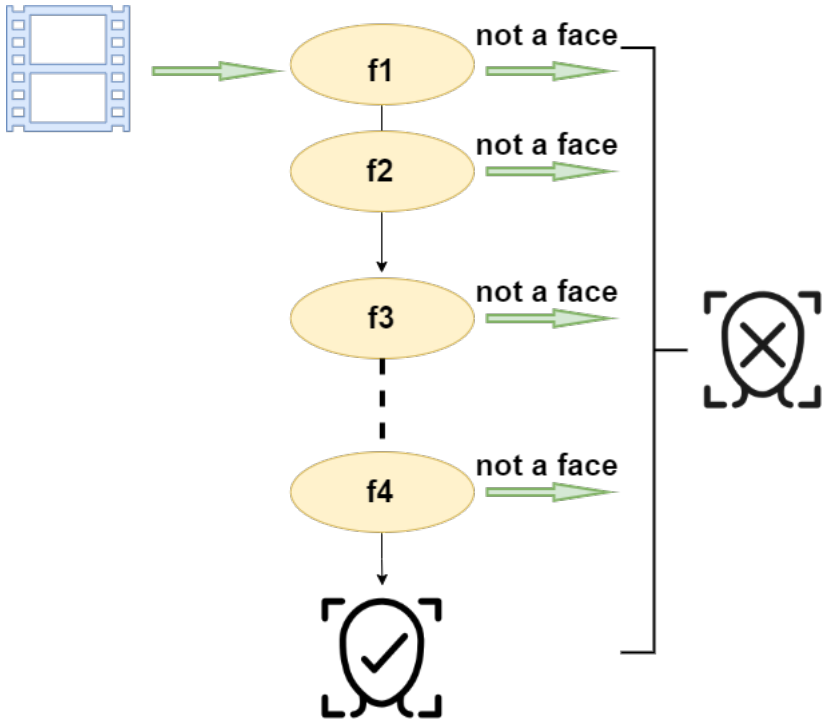


Figure 4.8: Cascaded Face Detection Technique

4.6 System Design for Head Pose Estimation

A head pose estimation works on the basis of multiple functions and parameters. On detection, we consider an image frame as a rigid body with orientation with respect to a fixed coordinate system. We can then define it in a 3-dimensional geometrical space to outline the world to screen coordinates as shown in Figure 4.9.

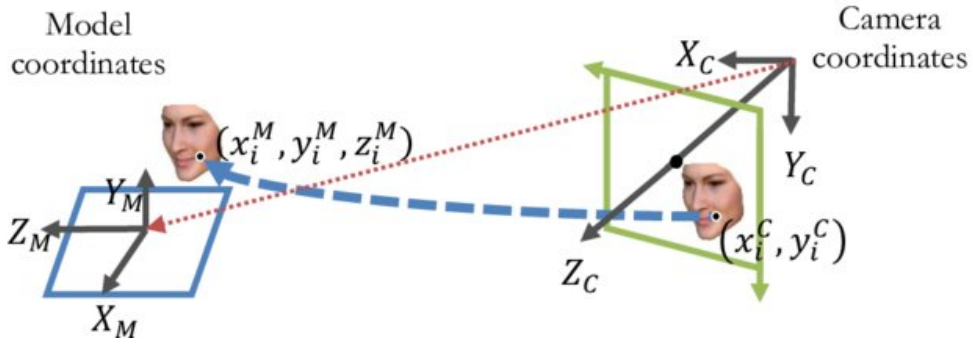


Figure 4.9: World to Screen Coordinates for Pose Estimation

The camera calibration, however, requires us to define a relation between pixels (camera units) and a real-world unit. We, therefore, use the distortion coefficient to correct the position of an output image with respect to an input image [42]. The radical factor uses

the formula defined in Equation 4.3.

$$\begin{aligned} x_{\text{corrected}} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ y_{\text{corrected}} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{aligned} \tag{4.3}$$

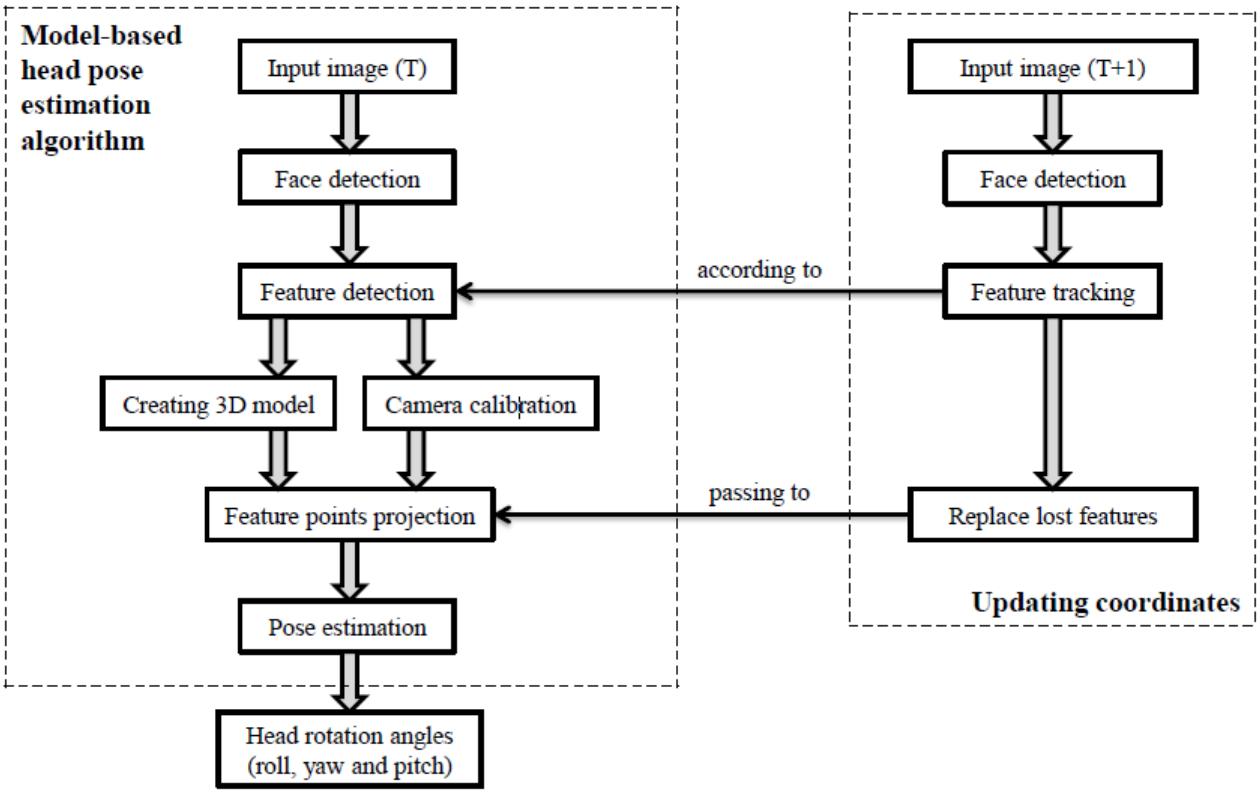


Figure 4.10: Head Pose Estimation Flow Diagram

Figure 4.11 shows the flow diagram for a Head Pose Estimation algorithm. A video is recorded from the camera, converted into the image frame which becomes the input for a face detection algorithm. After calibration using the cam matrix and distortion coefficient application, a feature point projection is created. Using Euler angles, we can then calculate the rotation angles defined in Chapter 3.6.
For a 3D modeling of World Coordinates, we need 3D locations of facial points in the arbitrary reference frame. The following points are the base for calculating the head pose in a 3-dimensional space [42]. The points are defined in the list below :

- Tip of the nose : (0.0, 0.0, 0.0)
- Chin : (0.0, -330.0, -65.0)
- Left corner of the left eye : (-225.0f, 170.0f, -135.0)

- Right corner of the right eye : (225.0, 170.0, -135.0)
- Left corner of the mouth : (-150.0, -150.0, -125.0)
- Right corner of the mouth : (150.0, -150.0, -125.0)

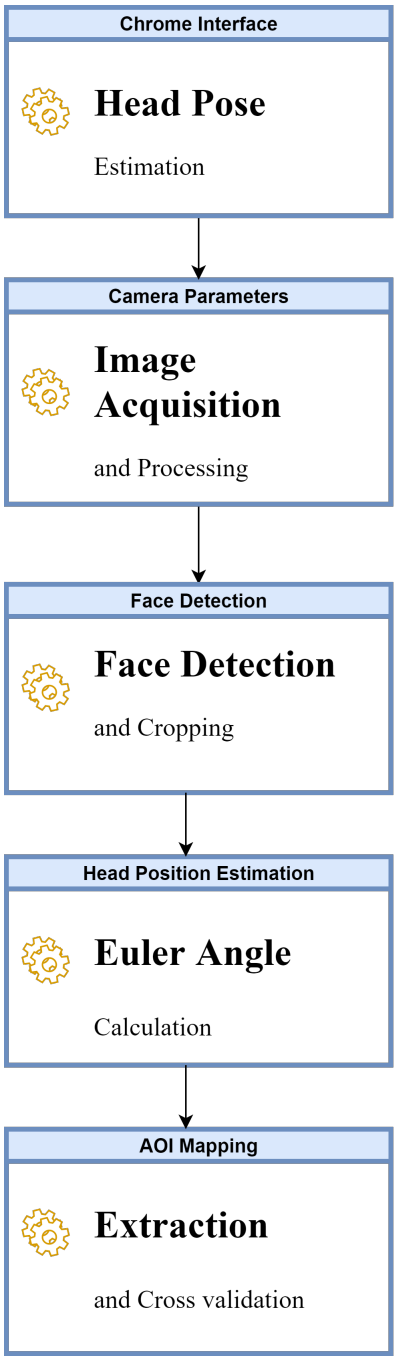


Figure 4.11: Head Pose System Diagram

Two important motion factors need to underlined here to mathematically represent the camera motion. These are the translation and rotational matrices. Hereby, we only use

the rotational matrix to calculate the X, Y and Z axes with three degrees of freedom, and a 3 x 3 matrix.

4.7 Complete Project Architecture

A personalized dataset needs to be built for our face recognition system. Therefore we created a functionality where users can record images from a video in different poses and illuminations. These images contribute to my pickle embedding and are used to compare the extracted features from my face recognition algorithm. A flow diagram to build the dataset is shown in Figure 4.12.

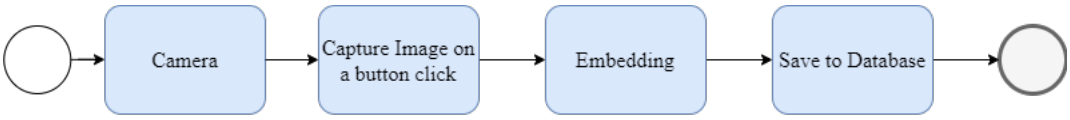


Figure 4.12: Build Dataset

For a complete project architecture to integrate the functionalities defined above, we created a complete project architecture as shown in Figure 4.13. A chrome extension can invoke the login page on our browser. On authentication, an admin can choose a functionality between Eye Tracking using myGaze API, Eye Tracking using Webcam, Face Recognition and Evaluation. The python backend supports multi-threading and maintains the request-response mechanism.

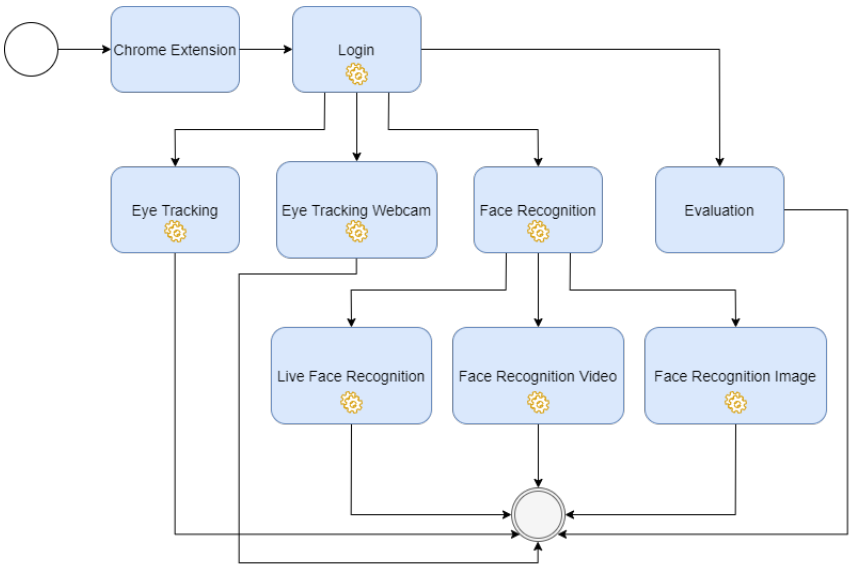


Figure 4.13: Project Flow

Chapter V

IMPLEMENTATION

The chapter defines the complete project implementation with a pictorial view of the components involved in the operations. The project consists of two major components.

- A Chrome Extension (Browser Plugin)
- A Python Project (Business Logic)

A third major component is the storage endpoint for saving information to be used in the evaluation is explained in Chapter VI
In the course of this chapter, we first define the technical requirements of the system. Here we throw light upon the hardware and software environments used during the development of this project. In the second part of the chapter, we define the components listed in the last paragraph. Since these components form the backbone of this project, it can be used to explain the implementation of this project. The chapter is aimed at scripting the algorithms used in my project, define the project structure and classes to validate my methodology defined in Chapter IV

5.1 Technical Requirements

The section describes the technical details of the developmental environment, frameworks and technologies used in this project.

5.1.1 Developmental Environment

Table 5.1 defines the hardware used in creation, training and evaluation of this project, while Table 5.2 defines the software and DBMS used during development.

Feature	Description
Product name	HP Notebook - 15-ac138nia
Microprocessor	Intel Core i5-6200U
Memory	4 GB DDR3L SDRAM (1 x 4 GB)
Video Graphics	AMD Radeon R5 M330 Graphics
Webcam	HP TrueVision HD Webcam

Table 5.1: Hardware used during development

Type	Name
IDE	PyCharm v2019.3.2
Database	MySQL v8.0.18
Source Control	GitHub
Documentation	LaTeX (overleaf)

Table 5.2: Software and IDE used during development

5.1.2 Frameworks and Technologies Used

A list of frameworks and technologies used in the development are defined in Table 5.3

Name	Description
HTML5	Markup language for defining behaviour and properties of a webpage
JavaScript	Client side design and development programming language
Bootstrap	Frontend framework for creating responsiveness of web applications
Python	Backend programming language
Caffe	Deep Learning Framework for AI
myGazeSDK	Eye Tracking Software Development Kit

Table 5.3: Frameworks and Technologies

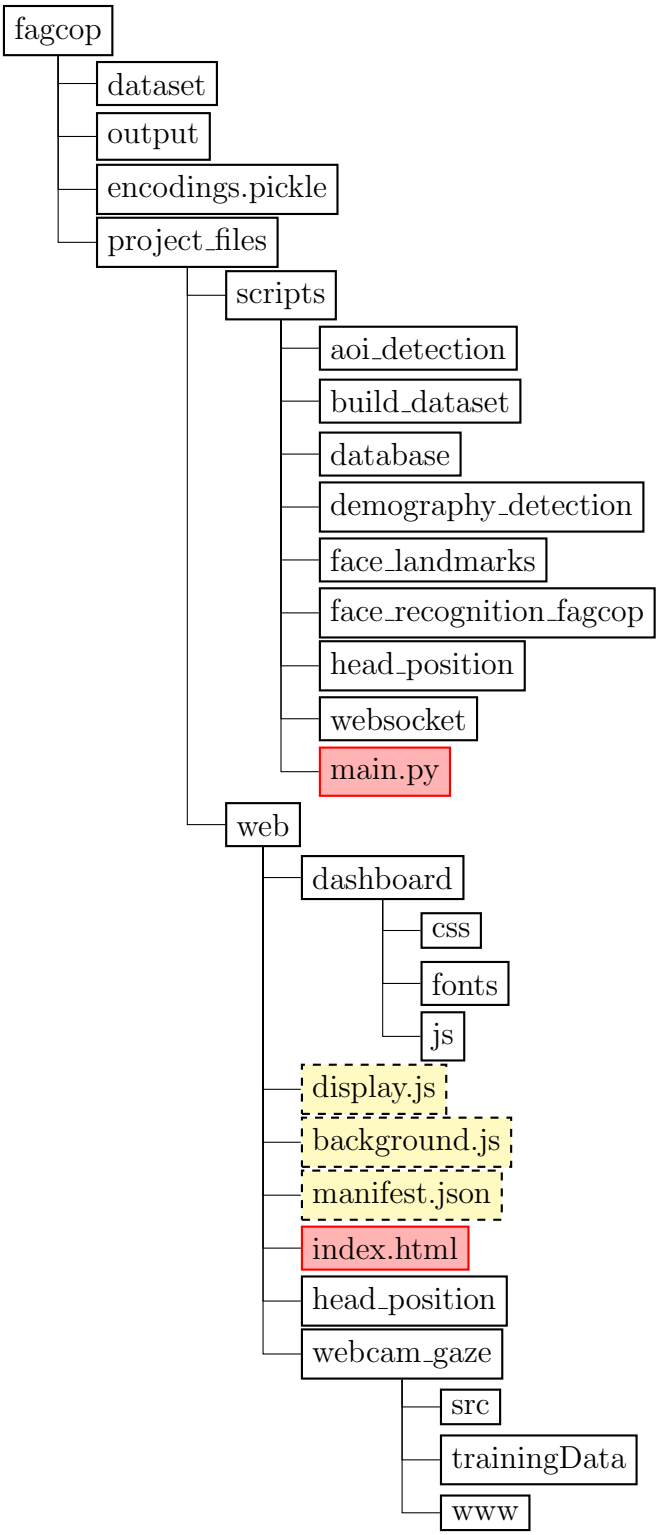
5.1.3 Libraries Used

Pip was used to install libraries used to develop this product. A list of used libraries is shown below :

Name	Description
imutils	image processing functions such as translation, rotation, resizing
matplotlib	plotting package to generate histograms, plots, barcharts
nltk	natural language and computational lingusitics
sklearn	predictive data analysis, machine learning and data mining
scipy	scientific library for mathematics, science, and engineering.
numpy	performs array computing
flask	web application framework for complex applications
opencv-python	wrapper package for OpenCV python bindings.
dlib	data analysis and machine learning application tool
face-recognition	built using dlib, its is used to perform face recognition
requests	used for communication using HTTP requests
pymysql	MySQL driver for python
autobahn	WAMP real time framework for websockets client and server library
twisted	event based framework for asynchronous networking
compress	data compression library
serialization	API for multiple serialization formats
tensorflow	machine learning framework by Google

Table 5.4: List of libraries and their functional description

5.2 Project Structure



5.3 The Browser Plugin

5.3.1 Chrome Extension

We created a chrome extension for communication between our browser window and the python back-end. Extensions are lightweight, easy to create and manage programs that can be created to define custom functionality of any application. Extensions are built using HTML, JavaScript and CSS.
An extension is a combination of narrowly defined components merged together to create a product. An extension file is zipped into a .crx package that we can install in any browser[3].
There are a number of predefined steps to create an extension. The steps involved in creating an extension are :

- Creating a Manifest
- Adding Instructions to Manifest
- Introducing a User Interface
- Layer Logic
- Adding Options.
- Background Scripts

5.3.1.1 Creating the Manifest

An extension consist of a JSON formatted manifest file consisting of basic information. Properties of a manifest are divided into required, recommended, optional, and pick_one. The required properties are manifest version and name of the extension. Icons, description and locale are the recommended properties. Furthermore, we also defined content_security_policy, content scripts and a single background script. This is shown in the code snippet.

```
1 {
2     "manifest_version": 2,
3
4     "name": "FAGCOP Extension",
5     "description": "Face and Gaze for Chrome Presentations",
6     "version": "0.1.0",
7     "author": "Nabil Bukhari",
8     "content_security_policy": "script-src 'self' 'unsafe-eval'; object-
    src 'self'" ,
```

```
    "background": {
    "scripts": [ "background.js" ]
  },
    "browser_action": {},
    "icons": {
      "16": "/dashboard/img/image.png",
      "48": "/dashboard/img/image.png",
      "128": "/dashboard/img/image.png"
    },
    "permissions": [
        "activeTab","tabs","storage"
    ],
  "content_scripts": [
  {
    "matches": ["<all_urls>"],
    "js": ["jquery-3.2.1.min.js"]
  }
  ]
}
```

Algorithms and Code Snippets V.1: Creating the Manifest

After creating the manifest file, we can add the extension into any chrome browser as shown in the Figure 5.1 and 5.2.

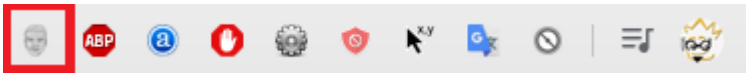


Figure 5.1: Chrome Extension on a browser toolbar

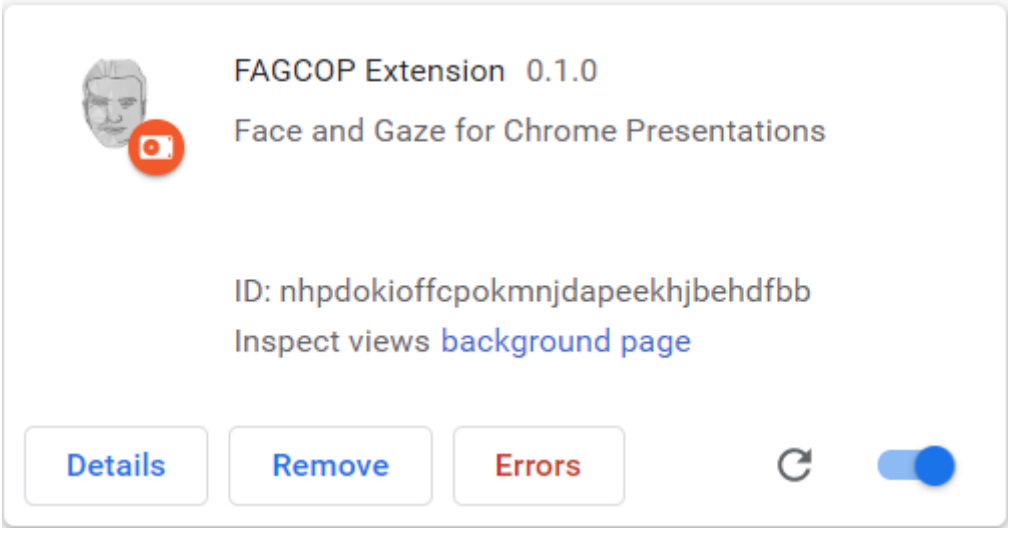


Figure 5.2: Loaded Chrome Extension with encrypted ID

5.3.1.2 Background.js

The background script is directly connected to the chrome extension. The background file is always listening to an event triggered by the extension. For this, we need to register our background script in the extension. There can be one or more background scripts in a single extension to keep the code modular.
In an extension, we can also add listeners to the code. A number of listeners are listed below :

- onInstalled Listener
- omMessage Listener
- onCompleted Listener
- onRemoved Listener
- onConnect Listener
- onActivated Listener
- onFocusChanged Listener

Hereby we define a function which initializes the web socket connection on onConnect Listener. The websocket start the interval for case of the websocket connection. It checks if python is running on a hardcoded 9000 port and then opens the socket. A 'CONNECTION ESTABLISHED' message is sent to the extension by python server as approval. The code for web socket connection is shown below :

```
1  function createWebsocketConnection() {
2      getConnectionInterval = setInterval(() => {
3
4          // Cancels the interval, if a successful connection is
       established
5          if (isConnected) {
6              clearInterval(getConnectionInterval);
7              return;
8          }
9
10         console.log('creating a new web socket connection');
11
12         //Creates a new Websocket connection
13         if (connectionCtrl.pending) {
14             websocket = new WebSocket('ws://127.0.0.1:9000');
15             connectionCtrl.pending = false;
```

```
16          // onopen triggers only one time, if a connection is
    sucessful established
17          websocket.onopen = () => {
18              websocket.send(JSON.stringify({
19                  msg: 'CONNECTION_ESTABLISHED'
20              }));
21              isConnected = true;
22              connectionCtrl.wasConnected = true;
23              console.log('web socket connection successful');
24          };
25          // onmessage triggers every time the ws receives a message
26          websocket.onmessage = (msgObject) => {
27              let message;
28              try {
29                  message = JSON.parse(msgObject.data);
30              } catch (e) {
31                  // Create fake object
32                  message = { msg: null }
33              }
34
35              if (message.msg === 'login-successful') {
36                  if (displayPort) {
37                      username = message.username
38                      displayPort.postMessage(msgObject.data);
39                  }
40              } else {
41                  console.count('Unknown message received...' +
    msgObject);
42              }
43
44              if(message.screen_position_head !== null){
45                  displayPort.postMessage(message);
46              }
47          };
48
49          // onclose is called when the connection is closed from
    client/server
50          websocket.onclose = () => {
51              isConnected = false;
52              username = null;
53
54              if (connectionCtrl.wasConnected) {
55                  createWebsocketConnection();
56                  connectionCtrl.wasConnected = false;
57              }
58              connectionCtrl.pending = true;
```

```
59                  console.log('web socket connection closed successfully')
    ;
60              }
61
62              // onerror only triggers if an error occurs
63              websocket.onerror = (err) => {
64                  isConnected = false;
65                  username = null;
66                  connectionCtrl.pending = true;
67              };
68          }
69      }, 5000);
70  }
```

Algorithms and Code Snippets V.2: Web Socket Connection

5.3.1.3 Connecting the Display.js

We define the onConnect listener in the background file below. A switch case is applied to define multiple cases for messages and payload received from the extension. In the following code, some of the cases defined are :

- login
- still-logged-in
- logout
- face_recognition
- head_pose
- head_pose_extended

```
1  /**
2   * Register listeners to the different plugin ports
3   */
4  chrome.runtime.onConnect.addListener((port) => {
5      console.log('on connect');
6      console.log(port);
7      if (port.name === 'display') {
8          displayPort = port;
9
10         displayPort.onMessage.addListener((msgObject) => {
11
12             let obj = JSON.parse(msgObject);
```

```
13            console.log(msgObject);
14            switch (obj.msg) {
15
16                case 'login':
17                    // Sending a message to the client to perform the
    simple user login
18                    websocket.send(JSON.stringify({
19                        msg: 'USER_LOGIN',
20                        payload: {
21                            username: obj.username,
22                            password: obj.password
23                        }
24                    }));
25                    break;
26
27                case 'still-logged-in':
28                    if (username) {
29                        port.postMessage(JSON.stringify({
30                            msg: 'still-logged-in',
31                            username: username
32                        }));
33                    }
34                    break;
35
36                case 'logout':
37                    websocket.send(JSON.stringify({
38                        msg: 'USER_LOGOUT'
39                    }));
40                    username = null;
41                    break;
42
43                case 'face_recognition':
44                    // Sending a message to the client to run face
    detection
45                    websocket.send(JSON.stringify({
46                        msg: 'FACE_RECOGNITION'
47                    }));
48                    break;
49
50                case 'head_pose':
51                    // Sending a message to the client to run head
    position
52                    websocket.send(JSON.stringify({
53                        msg: 'HEAD_POSITION'
54                    }));
55                    break;
```

```
56
57                  case 'head_pose_extended':
58                      // Sending a message to the client to run head
    position
59                      websocket.send(JSON.stringify({
60                          msg: 'HEAD_POSITION_EXTENDED'
61                      }));
62                      break;
63
64                  default:
65                      console.error('Unknown message ', msgObject.msg);
66
67              }
68
69          });
70      }
71
72      if (port.name === 'content') {
73          contentPort = port;
74
75          contentPort.onMessage.addListener((msgObject) => {
76
77              switch (msgObject.msg) {
78
79                  case 'SCROLL_OFFSET_UPDATE':
80
81                      tabList[msgObject.tabId].scrollOffsetX = msgObject.
    scrollOffsetX;
82                      tabList[msgObject.tabId].scrollOffsetY = msgObject.
    scrollOffsetY;
83
84                      if (websocket && isConnected) {
85                          websocket.send(JSON.stringify(msgObject));
86                      }
87                      break;
88
89                  case 'BROWSER_SIZE_UPDATE':
90
91                      windowList[msgObject.windowId].innerWidth =
    msgObject.innerWidth;
92                      windowList[msgObject.windowId].innerHeight =
    msgObject.innerHeight;
93                      windowList[msgObject.windowId].width = msgObject.
    width;
94                      windowList[msgObject.windowId].height = msgObject.
    height;
```

```
95
96                  if (websocket && isConnected) {
97                      browserSizeUpdateMessage = {
98                          msg: msgObject.msg,
99                          windowId: msgObject.windowId,
100                         tabId: msgObject.tabId,
101                         width: msgObject.innerWidth,
102                         height: msgObject.innerHeight
103                     };
104                     // websocket.send(JSON.stringify(
    browserSizeUpdateMessage));
105                 }
106
107                 updateBrowserPositionOffset(msgObject.windowId, true
    );
108
109
110                 break;
111
112             case 'WEBPAGE_AOIS_UPDATE':
113
114                 if (websocket && isConnected) {
115                     websocket.send(JSON.stringify(msgObject));
116                 }
117                 break;
118
119             case 'WEBPAGE_AOIS_NEW':
120
121                 if (websocket && isConnected) {
122                     websocket.send(JSON.stringify(msgObject));
123                 }
124                 break;
125
126             case 'WEBPAGE_AOIS_REMOVE':
127
128                 if (websocket && isConnected) {
129                     websocket.send(JSON.stringify(msgObject));
130                 }
131                 break;
132
133
134             default:
135                 console.error('Unkown message type: ' + msgObject.
    msg);
136
137         }
```

```
138
139         });
140
141     }
142
143 });
```

Algorithms and Code Snippets V.3: onConnect functionality with various cases

5.3.2 Login and Dashboard - User Interface

On clicking the extension, the user is redirected to a login page as shown in Figure 5.3. The login page consist of text boxes for a username and a password.

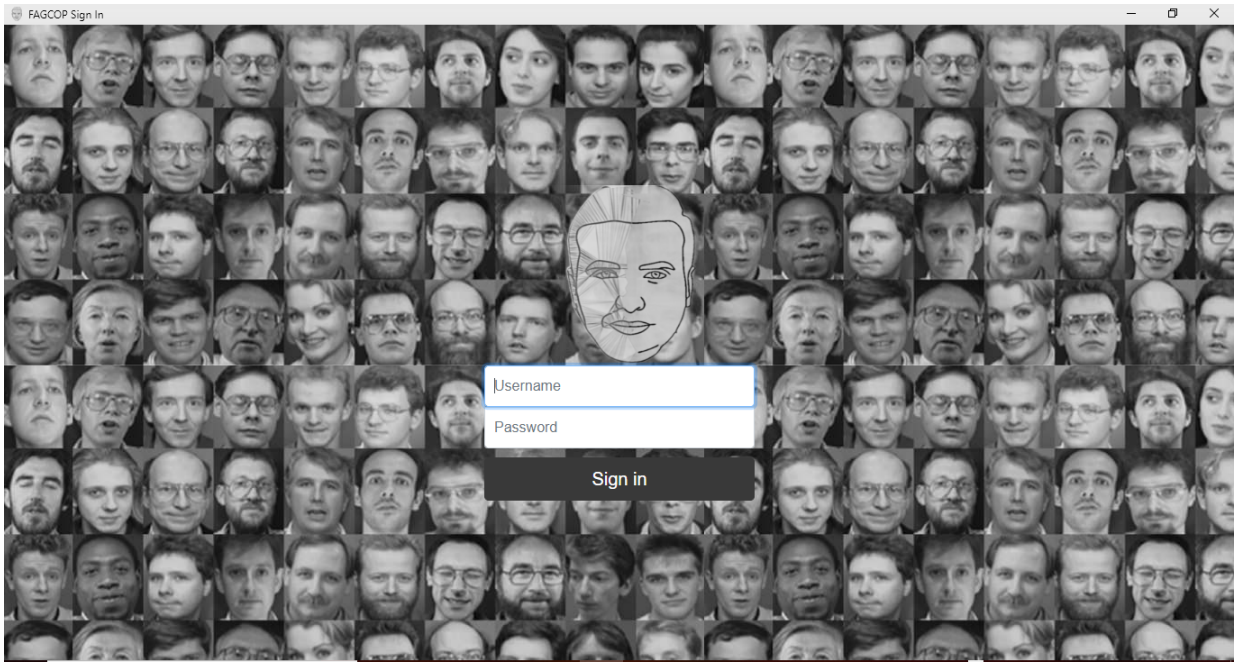


Figure 5.3: Representation of our FAGCOP - Login Page

The onClicked event of our extension run the chrome.windows.create event of JavaScript written in the background file to show the login page. The code for this event is defined below :

```
/**
 * Opens a new window and displays the current selected word
 */
chrome.browserAction.onClicked.addListener(function (tab) {
    popup_window = chrome.windows.create({
        url: chrome.runtime.getURL("index.html"),
        type: "popup",
```

```
8           width: screen.width,
9           height: screen.height
10      }, function (win) { });
11      console.log('browser extension is clicked')
12  });
```

Algorithms and Code Snippets V.4: Listener to Login Initiation

After authentication, the user is redirected to a Dashboard for selection as shown in Figure 5.4. Dashboard consist of analysis, a top navigation bar and a left side bar menu. Different options can be selected as shown in Figure 5.5.

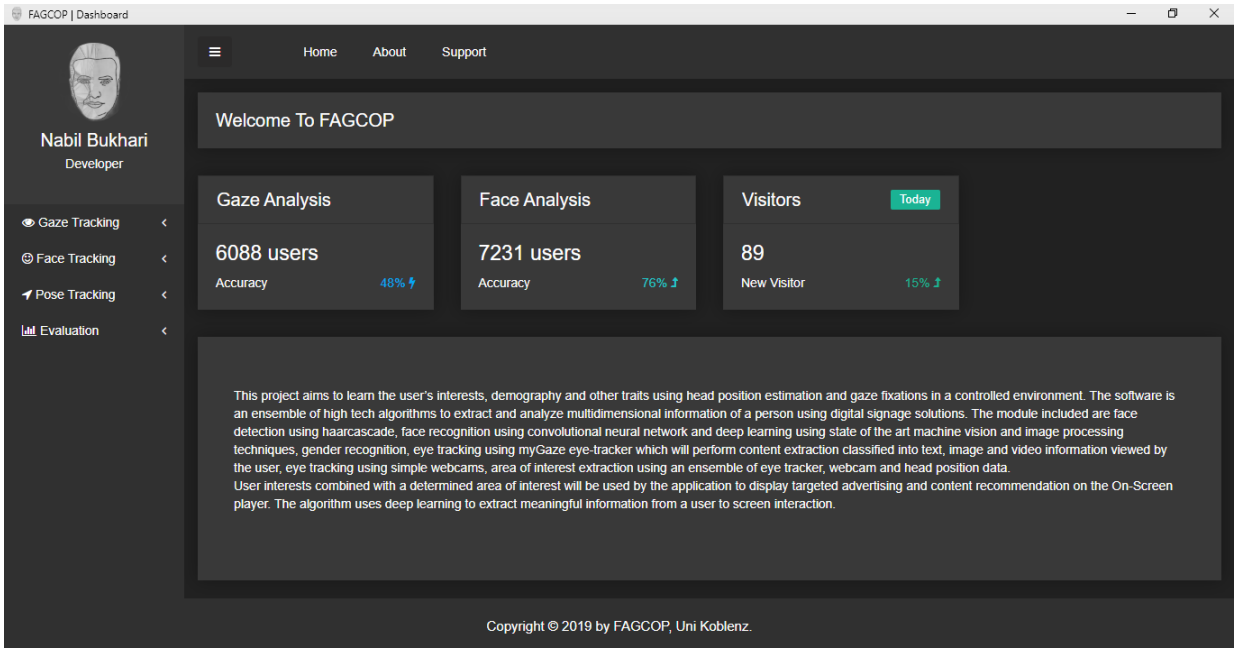


Figure 5.4: Representation of our FAGCOP - Dashboard

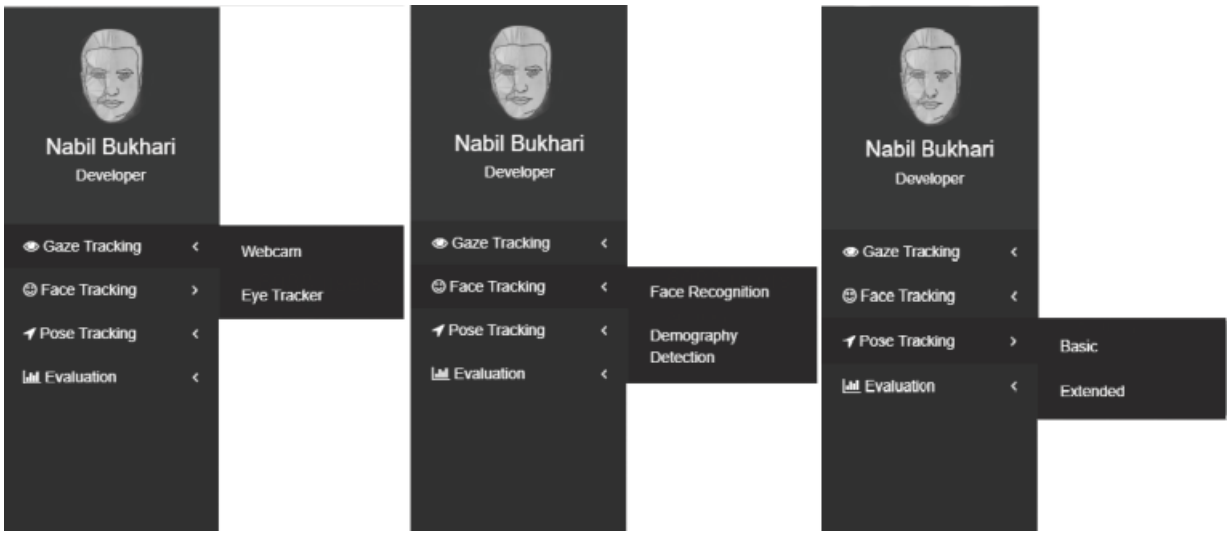


Figure 5.5: Representation of our FAGCOP - Side Menu

5.4 The Business Logic

The execution starts only after starting the web-server on a hard coded port 9000, this can obviously be set in a separate parameters file. The admin needs to run Python main.py file to start the server. The main class is the base class for running every functionality. It initializes all the classes needed for a processing requests. The initialization of server is shown in the code below:

```
(venv) C:\PycharmProjects\FaceRecognitionHOG\project_files\scripts>py
   main.py
Initializing..
Instantiate the DB..
Connection successful..
Starting the Websocket Server..
2020-02-01 20:31:17+0100 [-] Log opened.
2020-02-01 20:31:17+0100 [-] BroadcastServerFactory starting on 9000
2020-02-01 20:31:17+0100 [-] Starting factory <websocket.WebsocketServer
   .BroadcastServerFactory object at 0x0000024644F556D8>
2020-02-01 20:31:17+0100 [-] Site starting on 8080
2020-02-01 20:31:17+0100 [-] Starting factory <twisted.web.server.Site
   object at 0x000002464BFB1D68>
2020-02-01 20:31:17+0100 [-] server running
2020-02-01 20:31:17+0100 [-] Instantiate the Face Recognition Class
2020-02-01 20:31:18+0100 [-] Enter q to quit!
2020-02-01 20:31:20+0100 [-] registered client tcp4:127.0.0.1:38775
2020-02-01 20:31:20+0100 [-] CONNECTION_ESTABLISHED
```

Algorithms and Code Snippets V.5: Starting the Web Server

The server can then be closed by pressing 'q' in the terminal.

5.4.1 Eye Tracking

Eye Tracking performed in this project uses myGazeSDK provided by the university for research. The device needs a few softwares to be installed beforehand which are listed below:

- myGaze Installer Package
- myGaze SDK

but the software comes with a few limitations. The device is not supported for most of the operating systems and development is only allowed using the systems defined in Table 5.5.

Operating System	Notes
Windows 7 32/64 bit	Supported
Windows 8 32/64 bit	Supported
Windows 10 32/64 bit	Supported
Linux	Planned
Mac OS X	Planned
Windows XP/Vista 32/64 bit	Not supported

Table 5.5: Operating System Support for myGaze SDK

myGaze Eye-Tracker needs to be connected to USB 3.0 port since the device requires improved power management, and higher transfer speed. After connection, the admin needs to run the myGaze application to start the tracking process. Though a limitation for the eye tracker is the accuracy analysis of the device since the functionality of the device is factory decided, and can not be improved by our system. After login, the eye tracker starts calibrating the eye coordinates and gets data on fixed time intervals in milli seconds. On redirecting to a page, the aois are allocated on the screen depending on classes defined in the HTML and stored in the database. An area of interest defines the regions in a DOM that the researcher is interested in gathering data about. The guidelines followed for AOI construction are defined below [38]:

- AOIs should only be defined for objects of interest.
- They do not need to fill the entire screen.

- The padding (or margin) around an object should depend on three factors: (i) The importance of capturing every fixation on that object, (ii) The amount of white space surrounding the object, (iii) Expected variance in fixation positions across participants.
- Objects of interest should be positioned in the stimulus such that white space can exist between AOIs.
- The minimal AOI size should be determined by the precision and accuracy of the recorded eye tracking data.
- Arbitrary AOI positioning should be avoided [16].
- For sparse stimuli the AOIs should be as large as possible.
- AOI construction should depend on both the data quality and the type of stimulus used.

In the scope of this project, we check a set of classes defined on each webpage which allocates the area as an AOI. A single webpage can have a single on many AOI depending on the structure of a webpage. The position of each AOI is then retrieved in the form of x and y coordinates relative to the web page's upper left corner and dimensions. Each AOI is then divided into further parts to get insights of parts which attracts the user's attraction. The parts of an AOI for our application are defined below :

- **Header Area :** The top location defining base information about the webpage
- **Text Area :** The text part on the webpage defining any information.
- **Media Area :** Image or video part on the webpage.
- **Bottom Area :** Lower area, containing buttons, reference links, and statistical information.

The above mentioned areas are shown in Figure 5.6.

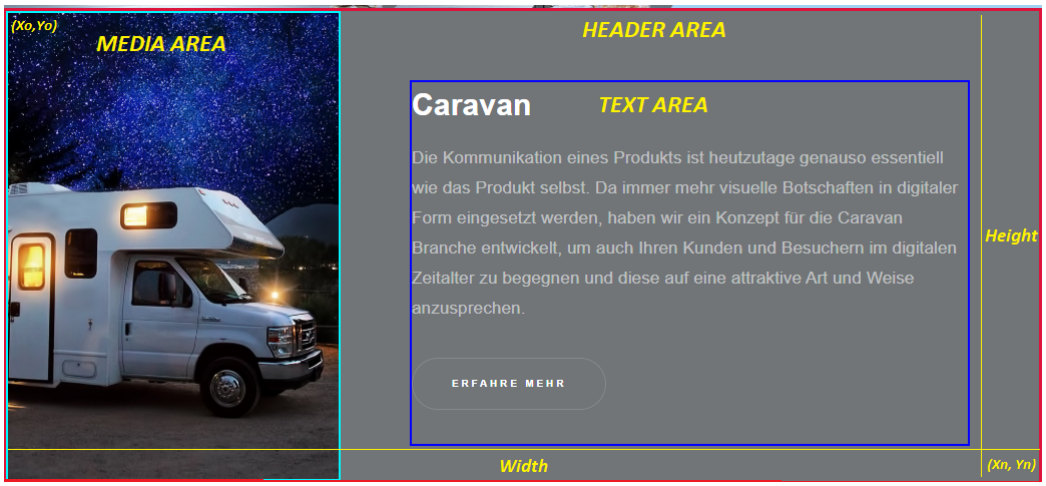


Figure 5.6: Area of Interest Division and Screen Dimensions

Since we have the dimensions of eyes from the eye-tracker and dimensions of AOI from the webpage, we can easily apply a comparison operation to find out the time spent by user on a certain section of the screen. This is termed as the Matching Algorithm. The data collected is divided into following :

- total reading time
- header reading time
- text reading time
- media reading time
- bottom reading time

The algorithm for Matching Algorithm is shown below:

```
If list of aois is not empty
    If fixation is inside window
        Add scrolling_offset to fixation y-coordinate
        Set location to default
    For each aoi in aois do
        If currentAOI is set AND
        aoi is currentAOI AND
        fixation is not inside aoi
            Calculate the total reading time
            Determine the time measurements
            for the locations
            Store time measurement in database
            Set all variable to default
        If fixation is inside aoi
```

```
            If start is not set
                Set start to current time
                Set firstFixation to the current fixation
                Set lastFixation to the current fixation
                Determine the time measurements for the locations
                Store fixation in database
```

Algorithms and Code Snippets V.6: Pseudocode for Matching Algorithm in Eye Tracking

5.4.2 Face Recognition - The FaceNet Architecture

Facenet is a face recognition system developed in 2015 by researchers at Google that achieved then state-of-the-art results on a range of face recognition benchmark datasets. Facenet is a Deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches.
The FaceNet system can be used to extract high-quality features from faces, called face embedding that can then be used to train a face identification system.
Difference between FaceNet and its predecessors was that Facenet did not learn to distinguish faces directly but learned a mapping from face Images to a compact Euclidian space.
Our method is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity faces of the same person have small distances and faces of distinct people have large distances.

Figure 5.7: FaceNet overall architecture [52]

Face recognition/verification simply involves thresholding the distance between the two embeddings; recognition becomes a k-NN classification problem; and clustering can be

achieved using off-the-shelf techniques such as k-means or agglomerative clustering.

5.4.2.1 Triplet Loss

The network is trained using triplet loss, which optimizes the embeddings. Training corresponds to optimizing embedding such that Similar faces lie together in subspace.

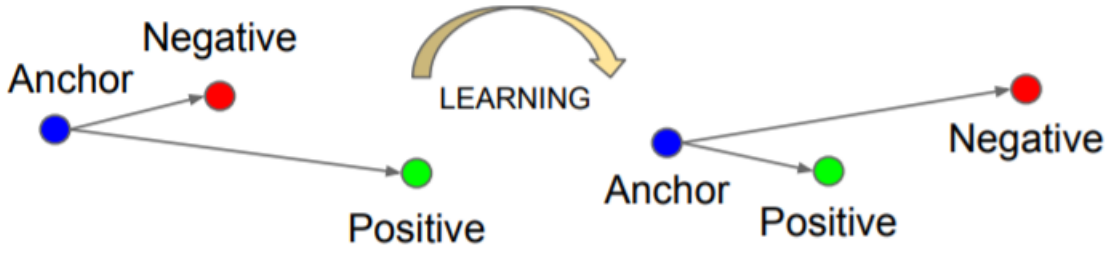


Figure 5.8: Triplet loss training [26]

Here we want to ensure that an image xai(anchor) of a specific person is closer to all other images xpi(positive)of the same person than it is to any image xni(negative) of any other person. This is visualized in Figure 5.8.
Mathematically

$$\|f\left(x_i^a\right) - f\left(x_i^p\right)\|_2^2 + \alpha < \|f\left(x_i^a\right) - f\left(x_i^n\right)\|_2^2 \tag{5.1}$$

$$\forall \left(f\left(x_i^a\right), f\left(x_i^p\right), f\left(x_i^n\right)\right) \in \mathcal{T} \tag{5.2}$$

where is a margin that is enforced between positive and negative pairs.

5.4.2.2 Algorithmic Implementation:

The implementation of face recognition algorithm has 3 core steps of implementation.

- Build Dataset / Gather Pre-build Dataset (build_face_dataset.py)
- Encode Faces (encode_faces.py)
- Recognize Faces

The face recognition algorithm can be divided in to three main procedures, however they work under the same functionality. The procedures are defined below:

- Recognize Face from Images

- Recognize Face from Saved Videos
- Recognize Face from Live Webcam

The algorithm to explain face recognition implementation is explain below:

```
1
2  Load Database Encodings
3  Start Capturing Video Frames
4  While video is being captured, run loop over frames
5      Grab the next frame
6      if not grabbed
7          end of the stream ; break
8      else
9          Convert input frame from BGR to RGB
10         Resize to width of 750px
11         Detect the (x, y)-coordinates of the bounding boxes
12         Calculate Encodings for each face
13             For each Face Encoding in Encodings
14                 Compare Face Encoding to Database Encodings
15                 if match happens
16                     Find the indexes of all matched faces
17                     Count the total number of times each face was
       matched
18                     Determine the recognized face with confidence level
19                     Draw the predicted face name on the image
```

Algorithms and Code Snippets V.7: Pseudocode for Face Recognition

Figure 5.9 shows a real time implementation of the face recognition algorithm with 4 true positives and 1 false positive value. The person "nabil_bukhari" is shown as "mutti_rehman" due to same facial features in both users. Figure 5.10 shows the correct facial recognition by the system even in case of an image shown on phone.

5.4.3 Head Position Estimation

The theoretical background and methodology for head position is already explained in Chapter 3.6 and 4.6 respectively. Here we take you through the code and implementation of our headpose algorithm. We start with initializing the constant parameters required for camera calibration such as *cam_matrix, distortion_coefficients,and shape predictor* for my case. After getting the frames from my video source, a frontal face detector is applied to detect face. A rectangle is then applied on face using dlib's detector. We calculate the translation and rotation vectors using *cv2.solvePnP* method [42], euler angles can then be calculated using *cv2.decomposeProjectionMatrix*[40]. Once Euler angle has been calculated, simple mathematical functions can be applied to calculate the head pose angle of user with respect to screen.

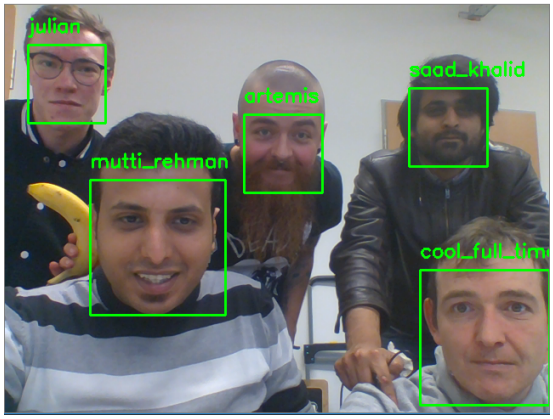


Figure 5.9: Face recognition of multiple persons in a single frame

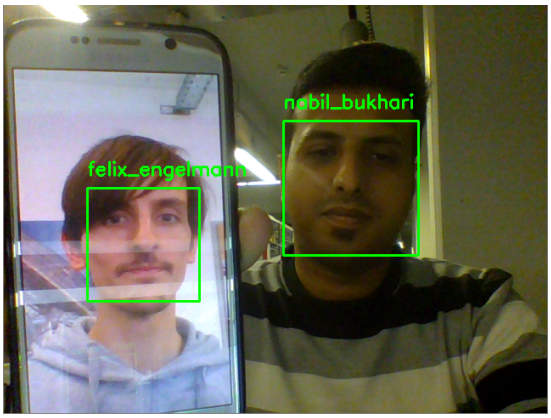


Figure 5.10: Face recognition using Images

```
# Constants
Get Shape Predictor for 68 Face Landmark File
Get Camera Matrix Constants
Get Distortion Coefficient Constants

# Code
Start Capturing Video Frames
While video is being captured, run loop over frames
    Apply face rectangles using dlib's detector

    if face is found
        Predict face using dlib's predictor
        Calculate rotation vector using solvePnP
        Calculate Euler Angles
        x-dim = euler_angle[0, 0]
        y-dim = euler_angle[1, 0]

        if x_dim >= 0 and y_dim >= 0
            head pose is on lower right

        if x_dim >= 0 and y_dim <= 0
            head pose is on lower left

        if x_dim <= 0 and y_dim >= 0
            head pose is on upper right

        if x_dim <= 0 and y_dim <= 0
            head pose is on upper left

        else
```

```
31                Not looking on screen
```

Algorithms and Code Snippets V.8: Pseudocode for Head Pose Estimation

An implementation of headpose functionality is shown in Figure 5.11 and 5.12. X, Y and Z dimensions are clearly visible on the screen in top left corner and a real time head position calculation can be seen.Figure 5.13 shows the screen divided in 4 parts. We divided the screen using div properties of the CSS and then updated it with JavaScript using the parameters sent from the back-end. The backend initially extracts head pose information and dimensions, evaluate where the user is looking on the screen and updates that box on the screen.

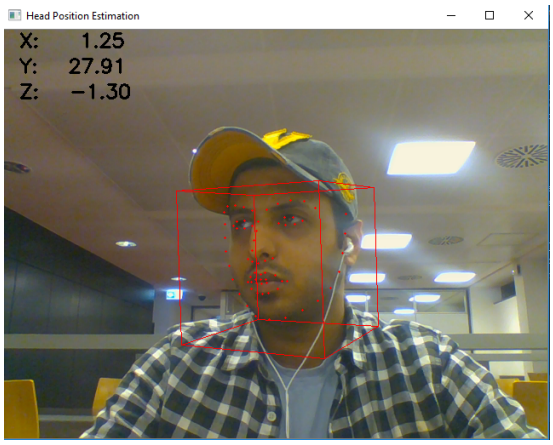


Figure 5.11: Head Position Implementation Side Pose

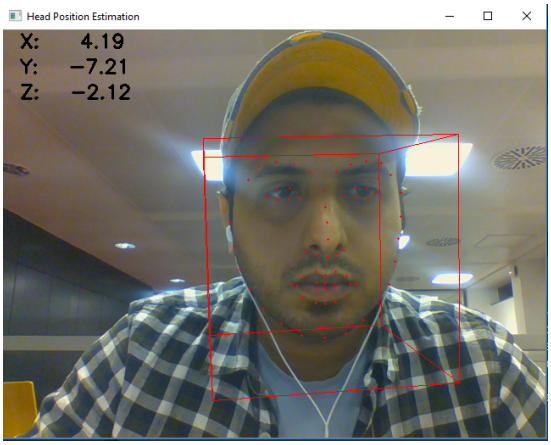


Figure 5.12: Head Position Implementation Frontal View

As shown in the Figure 5.13, the screen is initially divided in to four parts but in future can have an enhanced implementation as well with more screen divisions and a complex mathematical algorithm. White part on the screen defines current head orientation. The screen divisions are as follows :

- Lower Right
- Lower Left
- Upper Right
- Upper Left

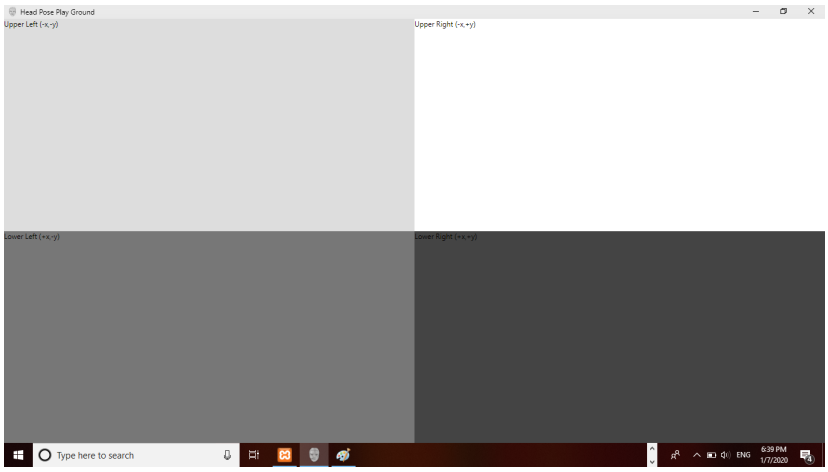


Figure 5.13: Head Pose to Screen Coordinates Calculation

5.4.4 Demography Extraction

The demography detection was comparatively easy but required the model to be trained intensively for good results. We used the functionality of deep neural network [41]. The algorithm focus on the work done by Gil Levi and Tal Hassner in the domain of Age and Gender Classification using Neural Networks [32]. We used the pretrained models created by Caffe authored by Yangqing Jia at Berkeley AI Research [24]. The algorithm starts by loading the caffe model for Gender and Age classification. Blobs of images are created for detection using deep neural network. If a face is detected, then for each face we calculate the confidence of both age and gender of the person. If the confidence value is greater than the threshold confidence, the gender of face is detected. The algorithm for the defined model is shown below :

```
1
2  Get Caffe Model for Gender
3  Get Array of Genders
4  Load OpenCV DNN for Gender and Face Model
5
6  Start Capturing Video Frames
7  While video is being captured, run loop over frames
8      Get Blobs from Image
9      For each detection in the dnn
10         Calculate confidence
11         if confidence > confidence_threshold
12             Add Face to Detected_Face_Array
13             For face in Detected_Face_Array
14                 Predict Gender using GenderNet Caffe Model
```

Algorithms and Code Snippets V.9: Pseudocode for Demography Extraction

The implementation of the above mentioned algorithm is shown below. Figure 5.14 shows the gender of a person detected correctly using the webcam, while Figure 5.15 shows demography detection using picture taken from a phone.

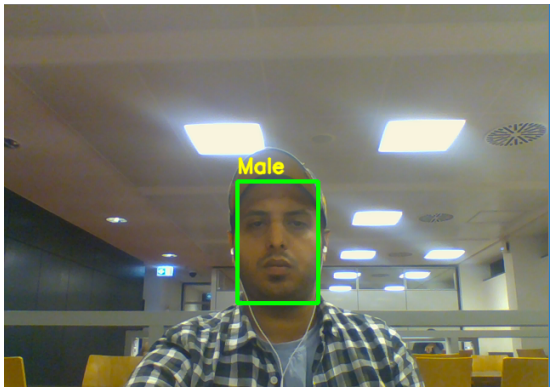


Figure 5.14: Gender Recognition using webcam

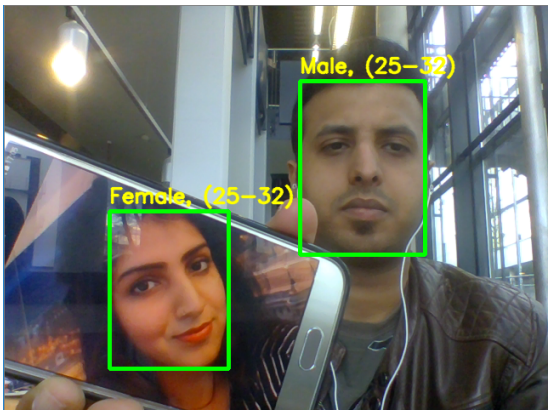


Figure 5.15: Demography Detection from Picture taken from a camera

Chapter VI

DATABASE AND DATASET ANALYSIS

6.1 Technical Information

We used pymysql, a python mysql client as our database connecting tool. The installation was done using a pip command. The API encourages the python based coding style to perform CRUD operations on database. It supports ease of use, and consistency in the applications for modular code.An example code for connection constructor is show below:

```
self.connection = pymysql.connect(
      host='localhost',
      user='root',
      password='',
      db='fagcop',
      charset='utf8mb4',
      cursorclass=pymysql.cursors.DictCursor
    )
```

Algorithms and Code Snippets VI.1: Database Connection

The technical information regarding generated database is enlisted below:
Name of Database: fagcop
Current Number of Tables: 8

6.2 Structure of Tables

We created our own database for storing the data generated from our algorithms. The data stored can then be used for evaluation and results generation. For each algorithm, we store records based on each captured video frame. In this section, we define the database

structure, tables and the values stored in each table. A list of created tables is enlisted below :

- age_gender
- aois
- face_recognition
- fixations
- gaze_data
- head_pose
- time_measurement
- user

6.2.0.1 Table User:

The table stores information regarding authenticated users. My software program can only be initiated after login by an administrator. The table consist fields for username, an encrypted salt and password, and a flag to enabling the user to use the system. The structure is shown in Table 6.1

id	username	email	salt	password	enabled	last_login
1	fagcop	user@fagcop.de	****	****	1	2020-02-04
2	nabil	nabil@fagcop.de	****	****	1	2020-01-14
3	test	test@fagcop.com	****	****	1	2020-02-14

Table 6.1: Table for User Authentication Information

6.2.0.2 Table Face Recognition:

The table for face recognition consist of fields for name, a flag for matched users, a counter for the number of images matched for a certain user's face with respect to images stored in the data-set and a time stamp on which user was using the face recognition functionality. The name field can either have the stored name for user, or an Ünknown"value for users who doesn't belong to the data-set. If an unknown user is detected, its is_matched value and match_count would obviously be 0. For a matched user, the match_count can be used in evaluating the system and calculating confidence value. The structure is shown in Table 6.2

id	name	is_matched	match_count	time_stamp
91	nabil_bukhari	1	11	2020-02-05 15:32:16
92	nabil_bukhari	1	10	2020-02-05 15:32:16
93	nabil_bukhari	1	10	2020-02-05 15:32:16
554	Unknown	0	0	2020-02-05 19:12:19
555	Unknown	0	0	2020-02-05 19:12:19

Table 6.2: Table for Face Recognition

6.2.0.3 Table Demography Information:

The table is used to store demographic information of users with a confidence value. The column for age and gender can have the one of the values from age and gender list as shown in the code below :

```
ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)',
    '(48-53)', '(60-100)']
genderList = ['Male', 'Female']
```

Algorithms and Code Snippets VI.2: Array of Age and Gender values

The detected age and gender is sent to database on each frame with a pre-calculated confidence as shown in Table 6.3.

id	age	age_confidence	gender	gender_confidence	time$_{s}tamp$
3281	(25-32)	0.957348	Male	0.999977	2020-02-05 17:06:47
3282	(25-32)	0.952621	Male	0.999977	2020-02-05 17:06:47
3283	(25-32)	0.910183	Male	0.999994	2020-02-05 17:06:47
3884	(8-12)	0.993034	Female	0.999996	2020-02-05 17:05:49
3885	(8-12)	0.910183	Female	0.999996	2020-02-05 17:05:49
4231	(0-2)	0.475332	Male	0.997565	2020-02-05 18:01:49
4232	(0-2)	0.415263	Male	0.997565	2020-02-05 18:01:49

Table 6.3: Table for Demography Information

6.2.0.4 Table Head Position Estimation:

The table consist of field for dimensions of calculated head pose and its corresponding value for screen positioning. Dimensions are divided in to x, y and z dimension, while screen value can have 4 different values as defined in Chapter 5.4.3. The structure is shown in Table 6.4

id	dim_x	dim_y	dim_z	screen_position	time_stamp
4571	-3.11711	-2.41219	4.52733	Upper Left	2020-02-05 16:59:18
4572	-4.11408	0.117512	4.24887	Upper Right	2020-02-05 16:59:18
4573	-2.58426	-3.17151	3.79655	Upper Left	2020-02-05 16:59:19
4574	-3.4834	-0.146707	4.50887	Upper Left	2020-02-05 16:59:19
4575	-4.18061	-1.31256	4.26317	Upper Left	2020-02-05 16:59:19
5214	18.7316	2.46494	1.87466	Lower Right	2020-02-05 17:01:03
5215	18.863	-0.632596	0.633023	Lower Left	2020-02-05 17:01:03

Table 6.4: Table for Head Pose Estimation

6.2.0.5 Table Gaze Data:

On authentication, a message is sent to back-end to check for a connected eye tracker. If an eye tracker is connected to the display, a start_tracking() process is initiated, which starts sending eye gaze coordinates to the database. On a web-page, a scrolling offset is defined as the scroll performed in Y dimension on web page. Since eye tracker give information in a fixed pixel space, we can add the scrolling offset to determine the exact AOI location on the webpage.The structure is shown in Table 6.5.

id	gazeX	gazeY	scrolling_offset	time_stamp	user
21	746.3793	247.6968	0	2020-02-07 10:41:16.21	1
22	597.6548	161.7598	0	2020-02-07 10:41:16.52	1
23	551.5461	482.5317	248	2020-02-07 10:41:16.83	1
24	674.2416	398.7261	248	2020-02-07 10:41:17.03	1
25	674.2416	282.4463	248	2020-02-07 10:41:17.19	1

Table 6.5: Table for Gaze Data

6.2.0.6 Table Area of Interest:

The table stores information regarding all area of interests present in a web-page. Each AOI is given a specified unique id which can be used as a foreign key in other tables. A json array regarding information of the AOI is stored in field of header, text_area, media_area, and bottom_area. This json array defines the position of AOI with respect to the webpage. We also store width, height and type of the AOI, which are used in processing and evaluation. The structure is shown in Table 6.6.

6.2.0.7 Table Fixation:

Chapter 3.2.1 already defines the idea behind fixations. In Table 6.7, we store the data for each fixation recorded by the eye tracker on screen. The matching algorithm sets AOI

id	aoi_uuid	header	text	media	bottom	user	width	height	type
1	dsa32410	json	json	json	json	1	325	1037	text
2	fac940b4	json	json	json	json	1	711	980	text

Table 6.6: Table for Area's of Interest

and location for each fixation value. Also the scrolling offset and user's id is stored for analysis.

id	aoi_id	location	posX	posY	scroll_offset	user	time_stamp
921	NULL	NULL	912.146	119.410	0	1	10:41:23
1022	fac940b4	header	690.793	562.577	599	1	10:42:23
1023	fac940b4	header	690.793	562.577	599	1	10:42:24
1024	fac940b4	text	790.793	664.945	599	1	10:42:24
1025	fac940b4	media	881.453	520.816	599	1	10:42:24

Table 6.7: Table for Fixations

6.2.0.8 Table Time Measurements:

The table defines total time spent by a user on each AOI. The total time can then be divided in to time spent on the header, text, media and bottom classes of the AOI.

id	aoi_uuid	totalReadingTime	header_rt	text_rt	media_rt	bottom_rt
211	fac940b4	485	255	230	0	0
212	fac940b4	1540	0	420	1120	0

Table 6.8: Table for Time Measurement's

6.3 Confidentiality and Anonymization

Confidentiality of data is a focal point for a lot of countries in today's world. A user is never eager to share its data with companies or research institutes. We observe that a user is always hesitant in a video recorded arena therefore we also tried to keep the confidentiality and anonymity of data in the system. Keeping that in mind, we conclude that using the algorithms in an online application or digital signage poses privacy risk to the customers. However, there can be substantial benefits as well. It can allow companies to improve their usability, understand human behavior and provide targeted content to users based on their interest and demography.

Hong et al. [19] stated that the privacy risks including webcam will be acceptable by users only if they see clear benefits that outweigh the risks. Users will be acceptable to swap their eye gaze behavior in exchange for benefits from companies.

Chapter VII

EXPERIMENTAL SETUP AND RESULTS

The chapter focuses on the experimental setup, evaluation and the discussion of results for the algorithms implemented in Chapter IV and V.

7.1 Environment

The expriment was conducted on a HP Notebook - 15-ac138nia PC with Windows 10 and Google Chrome in a maximized window. The display size for this laptop is 39.6 cm, 15.6“diagonal HD BrightView WLED-backlit with a resolution of 1366 x 768. Eye movements were recorded using HP TrueVision front facing HD Webcam [21].
The evaluation was performed under controlled environments to check the accuracy of different algorithms, however we also cited evaluation results of precompiled functions used in our algorithms.

7.2 Classification Model Evaluation for Face Recognition

We used face recognition python library which is highly inspired by FaceNet: A Unified Embedding for Face Recognition and clustering, which projects the face in low dimensional sub-space and learns embedding directly, where faces of same class lie together.Face similarity can then be quantified by distance between the projected faces. Hence, training corresponds to learning the embeddings where the face reside.
Training and evaluation of face recognition algorithm was done on Labelled Faces in the Wild and YouTube Faces in the Wild datasets.The Performance as reported by the orig-

inal authors on the widely used face datasets is as follows:
Labeled Faces in the Wild dataset, facenet achieved record accuracy of 99.63% and on YouTube Faces DB it achieves 95.12%.A classification accuracy of 98.87% when using the fixed center crop and record breaking 99.63% standard error when using the extra face alignment.

Above mentioned results are summarized in the table below:

Datasets	Classification Accuracy
LFW	98.87%
YouTube Faces	95.12%
LFW + face alignment	99.63%

Table 7.1: FaceNet Evaluation

We also performed evaluation for the face recognition algorithm under controlled condition with 9 participants. Each participant was invited for an interval of 10-15 minutes where we performed the experiment in two phases. In the first phase, We used the build dataset functionality to capture images of a participants with multiple poses. An embedding of files were performed for classification of images by the model. In the second phase, the participants had to sit in-front of the camera for some minutes to check the face recognition accuracy. All the data was stored into the database and evaluation was performed.
We chose the Classification Model Evaluation for this algorithm and calculated the confusion matrix from predicted data. The metrics for this evaluation are calculation of precision, recall, F1 score, and accuracy[5].

Participant A :

A total number of 164 records were stored in the database for Participant A. However 139 records were True Positive, 23 records were classified as Participant B, while 2 records were classified as Participant D. The confusion matrix is shown in Figure 7.1.

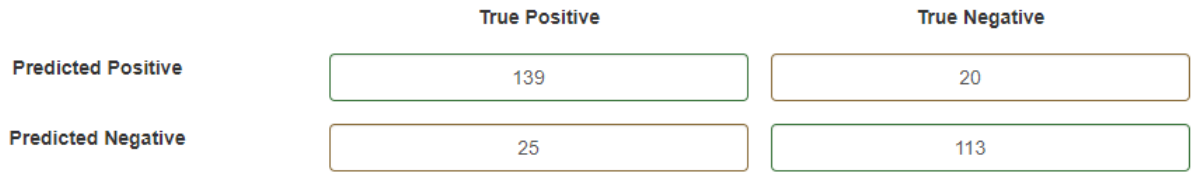


Figure 7.1: Confusion Matrix for Participant A

On the basis of participant's confusion matrix, a number of metrics were calculated as shown in the Table below. The resultant accuracy for Participant A is recorded at 84.85% with an F1 score of 0.8607.

Measure	Value	Derivations
Sensitivity	0.8476	TPR = TP / (TP + FN)
Specificity	0.8496	SPC = TN / (FP + TN)
Precision	0.8742	PPV = TP / (TP + FP)
Negative Predictive Value	0.8188	NPV = TN / (TN + FN)
False Positive Rate	0.1504	FPR = FP / (FP + TN)
False Discovery Rate	0.1258	FDR = FP / (FP + TP)
False Negative Rate	0.1524	FNR = FN / (FN + TP)
Accuracy	0.8485	ACC = (TP + TN) / (P + N)
F1 Score	0.8607	F1 = 2TP / (2TP + FP + FN)

Table 7.2: Classification Calculations for Participant A

Participant B :

A total number of 144 records were stored in the database for Participant B. 138 records were True Positive while 5 records were classified as Participant A, and 1 record was classified as Participant D. The confusion matrix is shown in Figure 7.2.

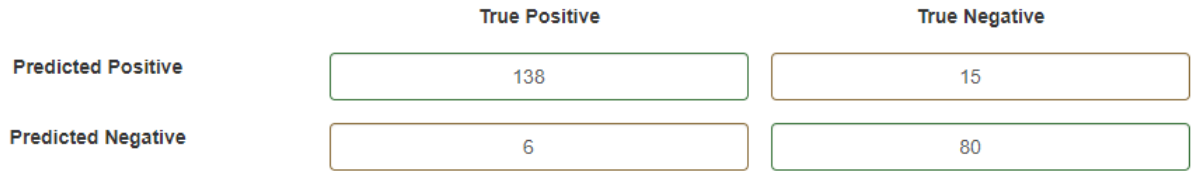


Figure 7.2: Confusion Matrix for Participant B

The calculated metrics for Participant B are shown in the Table below. The resultant accuracy for Participant B is recorded at 91.21% with an F1 score of 0.9293.

Measure	Value	Derivations
Sensitivity	0.9583	TPR = TP / (TP + FN)
Specificity	0.8421	SPC = TN / (FP + TN)
Precision	0.9020	PPV = TP / (TP + FP)
Negative Predictive Value	0.9302	NPV = TN / (TN + FN)
False Positive Rate	0.1579	FPR = FP / (FP + TN)
False Discovery Rate	0.0980	FDR = FP / (FP + TP)
False Negative Rate	0.0417	FNR = FN / (FN + TP)
Accuracy	0.9121	ACC = (TP + TN) / (P + N)
F1 Score	0.9293	F1 = 2TP / (2TP + FP + FN)

Table 7.3: Classification Calculations for Participant B

Participant C :

A total number of 153 records were stored in the database for Participant C. 135 records were True Positive while 18 records were classified as Participant B. There was no classification for Participant A and D. The confusion matrix is shown in Figure 7.3.

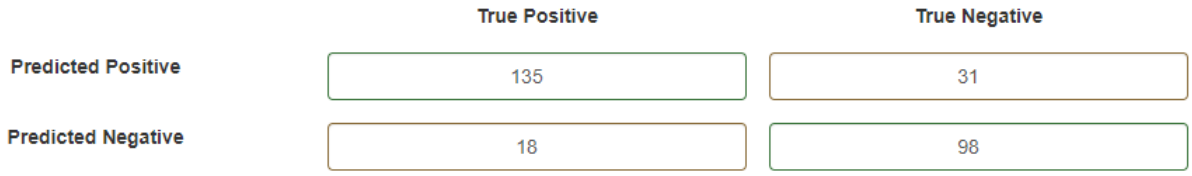


Figure 7.3: Confusion Matrix for Participant C

The calculated metrics for Participant C are shown in the Table below. The resultant accuracy for Participant C is recorded at 82.62% with an F1 score of 0.8464.

Measure	Value	Derivations
Sensitivity	0.8824	TPR = TP / (TP + FN)
Specificity	0.7597	SPC = TN / (FP + TN)
Precision	0.8133	PPV = TP / (TP + FP)
Negative Predictive Value	0.8448	NPV = TN / (TN + FN)
False Positive Rate	0.2403	FPR = FP / (FP + TN)
False Discovery Rate	0.1867	FDR = FP / (FP + TP)
False Negative Rate	0.1176	FNR = FN / (FN + TP)
Accuracy	0.8262	ACC = (TP + TN) / (P + N)
F1 Score	0.8464	F1 = 2TP / (2TP + FP + FN)

Table 7.4: Classification Calculations for Participant C

Participant D :

A total number of 156 records were stored in the database for Participant D. 141 records were True Positive while 2 records were classified as Participant A, and 13 records were classified as Participant B. The confusion matrix is shown in Figure 7.4.

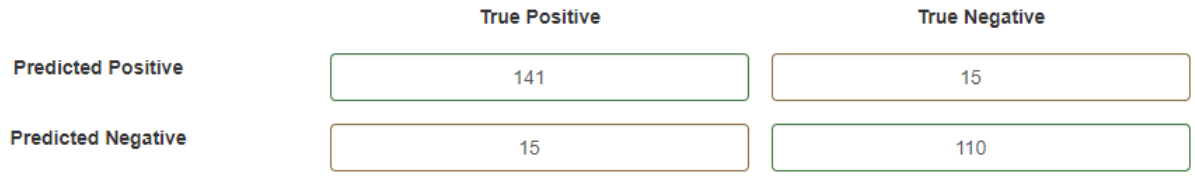


Figure 7.4: Confusion Matrix for Participant D

The calculated metrics for Participant D are shown in the Table below. The resultant accuracy for Participant D is recorded at 89.32% with an F1 score of 0.9038.

Measure	Value	Derivations
Sensitivity	0.9038	TPR = TP / (TP + FN)
Specificity	0.8800	SPC = TN / (FP + TN)
Precision	0.9038	PPV = TP / (TP + FP)
Negative Predictive Value	0.8800	NPV = TN / (TN + FN)
False Positive Rate	0.1200	FPR = FP / (FP + TN)
False Discovery Rate	0.0962	FDR = FP / (FP + TP)
False Negative Rate	0.0962	FNR = FN / (FN + TP)
Accuracy	0.8932	ACC = (TP + TN) / (P + N)
F1 Score	0.9038	F1 = 2TP / (2TP + FP + FN)

Table 7.5: Classification Calculations for Participant D

7.2.1 Mean Confusion Matrix Calculation

A mean confusion matrix is then calculated using above observations. The total record set for all participants was 617 with 553 records classified correctly. The confusion matrix is shown in Figure 7.5.

	Participant A	Participant B	Participant C	Participant D
Participant A	139	5	0	2
Participant B	23	138	18	13
Participant C	0	0	135	0
Participant D	2	1	0	141

Figure 7.5: Mean Confusion Matrix

A mean classification table is created using the generated records of participants. According to calculations shown in the table below, a mean accuracy is recorded at 89.63%. However an increase in accuracy will be observed by a potential increment of records in the images dataset.

Participant	n(truth)	n(classified)	Accuracy	Precision	F1 Score
Participant A	164	146	84.85%	0.8742	0.8607
Participant B	144	192	91.21%	0.9020	0.9293
Participant C	153	135	82.62%	0.8133	0.8464
Participant D	156	144	89.32%	0.9038	0.9038

Table 7.6: Mean Classification Calculations

Face Recognition Experiments at wizAI Solutions GmbH :
Experiments were conducted in wizAI solutions GmbH for different participants with age ranging from 25 to 40 years old. We took 13 images from each participant to build our dataset and encoded their faces. The experiments were conducted under good illumination conditions. The system showed 100% correct face recognition for 3 participants with strong facial features due to which it was easier for the algorithm to recognize them. For the other 2 participants, it was noticed that the system gave 97% accuracy. The only 3% decline was recorded only at the first or last few frames of the video where persons were moving in or out of the video respectively.

7.3 Classification Accuracy Calculation for Demography

The evaluation of demographic extraction can also be termed as a classification metric. The experimental setup for demography extraction was performed with 5 participants of different ethnical and demographic backgrounds. The ground truth for all participants is listed below :

- **Participant A :** A Male aged 27.
- **Participant B :** A Female aged 25.
- **Participant C :** A Male aged 26.
- **Participant D :** A Male aged 28.
- **Participant E :** A Male aged 29.
- **Participant F :** A Female aged 28.
- **Participant G :** A Male aged 26.
- **Participant H :** A Female aged 27.
- **Participant I :** A Male aged 32.

A total of 2186 records were stored for 5 different participants. The gender recognition algorithm worked quite well in different illumination conditions and showed an overall accuracy of 95.83% as shown in the Table below:

Participant	Accuracy	Number of Samples
Participant A	99%	443
Participant B	99%	221
Participant C	98%	193
Participant D	100%	188
Participant E	98%	229
Participant F	100%	124
Participant G	82.19%	292
Participant H	98.3%	236
Participant I	88%	260
Average	95.83%	2186

Table 7.7: Gender Recognition Evaluation

We performed age classification on the same dataset defined above. The algorithm showed minimal, understandable accuracy. From a total set of 2186 records, 1673 records were classified correctly while 513 records were false classifications resulting in an average accuracy of 76.53%. The system correctly classified most of the participants and gave and accuracy between 81% to 88% for most of the participants. However under different illumination conditions, it produced an accuracy of 73.57% for one participant and a low of 43.15% for another participant. The reason for this can be the type of face of user, usage of contact glasses, nose and ear piercings and facial beard. The participant aged 27 while the system was constantly detecting an age of 38-43. This concludes that the system can perform differently under different illumination scenarios.

Experiment	Accuracy	Number of Samples	True Classification	False Classification
Participant A	85.10%	443	377	66
Participant B	88.23%	221	195	26
Participant C	73.57%	193	142	51
Participant D	64.36%	188	121	67
Participant E	82.09%	229	188	41
Participant F	84.67%	124	105	19
Participant G	43.15%	292	126	166
Participant H	88.13%	236	208	28
Participant I	81.15%	260	211	49
Average	76.53%	2186	1673	513

Table 7.8: Age Classification Evaluation

7.4 Classification Accuracy Calculation for Head Pose to Screen Mapping

The evaluation of head pose extraction can be termed as a regression or a classification metric. In order to analyze the algorithm using a regression metric, we can calculate the Mean Absolute Error from the collected data, however a simpler approach for accuracy calculation can also be applied. The experimental setup consisted of many participants of diverse age and height. We collected a total of 2993 calculations for 3 dimensions captured from the webcam, and a screen position mapping with respect to the dimensions. The collected data was then matched against a ground truth which showed that a the system generated 2619 true positive results. The dataset calculations with respect to screen positions are shown in the Table below:

Head Position	Total Data Points	True Classification	False Classification	Accuracy
Upper Left	853	821	32	96.24%
Upper Right	771	725	46	94.03%
Lower Left	603	422	181	69.98%
Lower Right	766	651	115	84.98%
Average	2993	2619	374	87.50%

Table 7.9: Head Pose to Screen Mapping Evaluation

We observed high precision values for upper part of the screen as compared to the lower part. The upper part of the screen gives an accuracy of 95.135% while a low accuracy of 77.48% was observed on the lower part. The reason for this inconsistency can be the distance between camera and the participant, or the height of the participant, as participants with different heights can have different camera angles with respect to the screen. The distance inconsistency can be overcome by using stereo cameras, however angle inconsistency remains an open set problem. It was observed that weights of the Lower Left part on the screen can be adjusted to produce better results. Still we observe a good mean accuracy of 87.50% for the overall head pose system.

Chapter VIII

CONCLUSION

The thesis aimed at using artificial intelligence algorithms in an onScreen presentation. During the thesis, we explained the core concepts related to face recognition, demography extraction, eye-tracking using webcams and infrared eye-tracker. The face detection is based on haarcascading technique coined in the Viola-Jones research paper [59]. Face recognition was performed using convolutional neural networks, facenet and precompiled functions of OpenCV for python. However, we also researched the Histogram of Oriented gradients technique coined by Dalal-Triggs [7]. Though HOG is the base method used in object detection algorithms, it can also be used in face recognition at the cost of accuracy. The face recognition algorithm, FaceNet achieved high accuracy of 98.87% on Labeled Faces in the Wild dataset, and accuracy of 95.12% on Youtube faces dataset. My algorithm with 9 different participants recorded a mean accuracy of 89.63% with a small set of trained images, however with a better camera, and controlled illumination, the accuracy can be increased. It was also observed that people with strong facial features are easily and accurately recognized. Demography extraction is also performed using deep neural networks. Pretrained models, Caffe by Yangqing Jia is used in recognition[24], while the concept for dnn is based on a paper by Gil Levi and Tal Hassner [32]. An algorithm is trained and evaluated on 9 participants and accuracy of 95.83% was recorded for gender recognition and accuracy of 73.57% was recorded for age recognition.
The eye-tracking is based on myGaze tracker and a webcam. A screen is divided into multiple AOI's based on classes used in the DOM. An AOI has multiple sections, including header area, text area, image area, and a text area. We then compare the coordinates from AOI with the gaze coordinates. The compared AOI's are then stored into the database with specific fields of time spent on each part of the AOI. This gives us an insight of time spent on each AOI by the user which can be used to produce targeted advertisements.
It is observed that head pose estimation is not suited best for AOI extraction but can be used to support the accuracy of the above mentioned algorithms. The environment for head pose estimation was based on Euler angle calculations and the screen was divided

equally into four parts. A consistency in pose to screen mapping can be achieved by adjusting the weights in algorithm with respect to screen size. The accuracy of the algorithm is also dependent on the distance and angle factors of a user to screen. Mean accuracy of 87.50% is achieved by my algorithm. A combination of the above-mentioned algorithms can be combined to create a product, with a pipeline of face recognition to get the identity data of the user, demographic information to get age and gender of the user and eye-tracking to get AOI, and time spent on each AOI by the user. The ensemble can then be set up on an onScreen player to create targeted advertisements.

Bibliography

[1] M. Ariz, José J. Bengoechea, Arantxa Villanueva, Rafael Cabeza, "A novel 2D/3D database with automatic face annotation for head tracking and pose estimation", *Computer Vision and Image Understanding, Volume 148, Pages 201-210, ISSN 1077-3142*, 2016.

[2] G. Buscher, S. Dumais, and E. Cutrell, "The good, the bad, and the random: an eye-tracking study of ad quality in web search", *Proceedings of SIGIR, pages 42-49* , 2010.

[3] Chrome Developers, "What are extensions?, [Online]", *Available at: https://developer.chrome.com/extensions* [Accessed 01-02-2020].

[4] J. Clement, "Google: revenue distribution 2001-2018, Statista, 09-08-2019. [Online]", *Available at: https://www.statista.com/statistics/266471/distribution-of-googles-revenues-by-source/* [Accessed 01-01-2020].

[5] "Confusion Matrix - Online Calculator, [Online]", *Available at: http://onlineconfusionmatrix.com/* [Accessed 02-02-2020].

[6] F. C. Crow, "Summed-Area Tables for Texture Mapping 18(3)", *Computer Sciences Laboratory*, Palo Alto, 1984.

[7] N. Dalal, B. Triggs, "Histograms of Oriented Gradients for Human Detection", *International Conference on Computer Vision, pp. 886-893, Vol 1*, 2005.

[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of computer and system sciences, pp. 119-139*, 1997.

[9] U. Furbach and M. Maron, "NUI-Based Floor Navigation — A Case Study", Koblenz, 2013.

[10] W. Gerstner and W. M. Kistler, "In Spiking Neuron Models : Single Neurons, Populations, Plasticity", *Cambridge University Press*, 2002.

[11] X. Glorot, A. Bordes and Y. Bengio, "Deep sparse rectifier neural networks", *AISTATS '11: Proceedings of the 14th International Conference*, 2011.

[12] P. Golik, P. Doetsch and H. Ney, "Cross-Entropy vs. Squared Error Training: a Theoretical and Experimental Comparison", *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, Aachen, 2013.

[13] I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning", *MIT Press*, 2016.

[14] D. W. Hansen, Ji Q, "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, doi:10.1109/TPAMI.2009.30*, 2010.

[15] D. Harihara Santosh, P.G.Krishna Mohan, "Improved Face Recognition Rate Using HOG Features and SVM Classifier", *IOSR Journal of Electronics and Communication Engineering, Volume 11, Issue 4*, 2016.

[16] Roy S. Hessels, Chantal Kemner, Carlijn van den Boomen, Ignace T. C. Hooge, "The area-of-interest problem in eyetracking research: A noise-robust solution for face and sparse stimuli", *Behav Res 48, 1694–1712 (2016).Available at: https://doi.org/10.3758/s13428-015-0676-y*, 2015.

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation, vol. 9, no. 8, pp. 1735-1780*, 1997.

[18] Sebastian Höffner, "Gaze Tracking Using Common Webcams", *Institute of Cognitive Science Biologically oriented Computer Vision*, 2018.

[19] J. Hong, J. Ng, S. Lederer, and J. Landay, "Privacy risk models for designing privacy-sensitive ubiquitous computing systems", *Proceedings of DIS, pages 91-100*, 2004.

[20] Horizon 2020, "Multimedia Authoring Management using your Eyes Mind. [Online]" *Available at: http://www.mamem.eu/* [Accessed 08-09-2019].

[21] HP Team, "HP Customer Support - Knowledge Base. [Online]", *Available at: https://support.hp.com/hk-en/document/c04922813* [Accessed 18-01-2020].

[22] J. Huang, R. W. White, and G. Buscher, "User see, user point: Gaze and cursor alignment in web search", *Proceedings of CHI 12, pages 1341-1350*, 2012.

[23] A. Hyrskykari, P. Majaranta, R. Räihä, "From Gaze Control to Attentive Interfaces, [Online]", *Available at:*

https://www.researchgate.net/publication/228374956_From_gaze_control_to_attentive_interfaces, 2005.

[24] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor, "Caffe: Convolutional Architecture for Fast Feature Embedding", *Available: https://caffe.berkeleyvision.org/*, *Available: arXiv preprint arXiv:1408.5093*, 2014.

[25] Rana el Kaliouby, Rosalind Picard, "Emotions AI Overview [Online]" *Available at: https://www.affectiva.com/emotion-ai-overview/* [Accessed 18-01-2020].

[26] D. Karunakaran, "One shot learning explained using FaceNet, 27-09-2018. [Online]", *Available at: https://medium.com/intro-to-artificial-intelligence/one-shot-learning-explained-using-facenet-dff5ad52bd38* [Accessed 05-08-2019].

[27] Moritz Kassner, Will Patera, "Pupil Core [Online]" *Available at: https://pupil-labs.com/products/core/* [Accessed 18-01-2020].

[28] E. M Kok H. Jarodzka, "Before your very eyes: the value and limitations of eyetracking in medical education", *Available at: DOI: https://doi.org/10.1111/medu.13066* 2016.

[29] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradientbased learning applied to document recognition", *Proceedings of the IEEE*, 1998.

[30] C. Y. Lee, P. Gallagher and Z. Tu, "Generalizing Pooling Functions in Convolutional Neural Networks : Mixed, Gated, and Tree", *AISTATS*, 2015.

[31] Justin Liang, "Canny Edge Detection [Online]", *Available at: http://justin-liang.com/tutorials/canny/*. [Accessed 18-01-2020]

[32] Gil Levi and Tal Hassner, "Age and Gender Classification Using Convolutional Neural Networks", *Available at: IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG), at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Boston, 2015.

[33] Feng Lu, Yusuke Sugano, Takahiro Okabe, Yoichi Sato, "Inferring human gaze from appearance via adaptive linear regression", *In Proceedings of ICCV, pages 153-160*, 2011.

[34] Feng Lu, Yusuke Sugano, Toshiya Okabe, Yuuki Sato, "Head pose-free appearance-based gaze sensing via eye image synthesis", *In Proceedings of ICPR, pages 1008-1011*, 2012.

[35] W. H. Lui, "Real-time Webcam Gaze-tracking", *CS 231A Final Project*, 2014.

[36] Satya Mallick, "Head Pose Estimation using OpenCV and Dlib, 09-2016 [Online]", *Available: https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/*, [Accessed 18-01-2020].

[37] MyGaze Team, "myGaze Software Development Kit", *version 4.3*, 2015.

[38] Abed Naseri Douraki, Alexander Ott, Alisa Karolina Becker, Denis Schmidt, Fiorela Ciroku, Julian Stadager, Nabil Bukhari, Stela Nebiaj, "Eye tracking Research Lab Report", *Available at: [on personal request from nabil.afaraz@gmail.com] Universitat Koblenz Landau*, 2018.

[39] Open CV Developers, "Cascade Classifier, 10-01-2020. [Online]", *Available at: https://docs.opencv.org/3.4/db/d28/tutorial*$_c$*ascade*$_c$*lassifier.html* [*Accessed*$15-11-2019$].

[40] Open CV Developers, "Camera Calibration and 3D Reconstruction, [Online]", *Available at: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruc* [Accessed 02-02-2020].

[41] OpenCV Developers, "Deep Neural Network module, [Online]", *Available at: https://docs.opencv.org/3.4/d6/d0f/group__dnn.htmlga3b34fe7a29494a6a4295c169a7d32422* [Accessed 02-02-2020].

[42] Open CV Developers, "Camera calibration With OpenCV, [Online]", *Available at: https://docs.opencv.org/2.4/doc/tutorials/calib3d/camer_calibration/camera_calibration.html* [Accessed 10-10-2019].

[43] P. Pandey, "Face Detection with Python using OpenCV, 20-12-2018. [Online]", *Available at: https://www.datacamp.com/community/tutorials/face-detection-python-opencv* [Accessed 01-01-2020].

[44] A. Papoutsaki, N. Daskalova, P. Sangkloy, J. Huang, J. Laskey, J. Hays, "WebGazer: ScalableWebcam Eye Tracking Using User Interactions", *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 2016.

[45] A. Papoutsaki, J. Laskey, J. Huang, "SearchGazer: Webcam Eye Tracking for Remote Studies of Web Search", *CHIIR '17: Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval, Pages 17-26, Available at: https://doi.org/10.1145/3020165.3020170*, 2017.

[46] Savan Patel, "Support Vector Machine (SVM) Theory 03-05-2017. [Online]" *Available at: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72* [Accessed 08-09-2019].

[47] Rekha N, Dr.M.Z.Kurian, "Face Detection in Real Time Based on HOG", *International Journal of Advanced Research in Computer Engineering Technology, Vol 3, Issue 4*, 2014.

[48] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", *In Psychological Review, vol. 65, no. 6, pp. 386-406*, 1958.

[49] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors", *Nature, vol. 323*, 1986.

[50] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, Medium, 15-12-2018. [Online]", *Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53* [Accessed 15-11-2019].

[51] A. Schens, "Entwicklung einer Gesichtserkennenden Software zur Erfassung und Analyse von Personenströmen", *Universitat Koblenz Landau*, Koblenz, 2014.

[52] F. Schroff,D. Kalenichenko,J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering", *CVPR, page 815-823. IEEE Computer Society*, 2015.

[53] D. Scherer, A. Muller and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition", *20th International Conference on Artificial Neural Networks (ICANN)*, Thessaloniki, 2010.

[54] Matthias Straka, "Person Independent Head Pose Estimation by Non-Linear Regression of Histograms of Oriented Gradients", *Institute of Computer Vision and Graphics, Graz University of Technology* , 2009.

[55] Yusuke Sugano, Yasuyuki Matsushita, Yoichi Sato, "Calibration-free gaze sensing using saliency", *In Proceedings of CVPR, pages 2667-2674*, 2010.

[56] R. Thaware, "Real-Time Face Detection and Recognition with SVM and HOG Features, [Online]", *Available at: https://www.eeweb.com/profile/rajeevthaware/articles/real-time-face-detection-and-recognition-with-svm-and-hog-features*, 2018 [Accessed 05-08-2019].

[57] Tobii Team, "Tobii Pro. [Online]" *Available at: https://www.tobiipro.com/* [Accessed 18-01-2020].

[58] Anastasia Tsifouti, Sophie Triantaphillidou, Efthimia Bilissi, Mohamed-Chaker Larabi, "Acceptable bit-rates for human face identification from CCTV Imagery", *Proceedings of SPIE - The International Society for Optical Engineering*, *Available: DOI: 10.1117/12.2004140*, 2013.

[59] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *Conference on Computer Vision And Pattern Recognition*, 2001.

[60] M.H. Yang, D. J. Kriegman and N. Ahuja, "Detecting Faces in Images - A Survey", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2002.

[61] M. D. Zeiler and R. Fergus, "Stochastic Pooling for Regularization of Deep Convolutional Neural Networks", *ArXiv e-prints*, New York, 2013.