TÉCNICO LISBOA

Gaze detection as a proxy for medical doctor behavior during appointments

Ricardo Andrade Antão

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Inv. Plínio Moreno López
Prof. José Santos-Victor

Examination Committee

Supervisor(s): Inv. Plínio Moreno López, Prof. José Santos-Victor
Member of the Committee: Prof. Rodrigo Ventura

June 2021

Abstract

Non-verbal communication in the physician-patient relationship has been the target of more detailed studies in recent years. Of the various aspects of non-verbal communication, the physician's gaze direction has been one of the most studied, specifically the amount of time the physician spends looking at the patient. However, almost all the studies have used human annotation methods for gaze direction annotation, which are costly and laborious. This work will study the differences in gaze pattern behaviours in two types of consultations: presential and virtual. Initial observations at the Luz Saúde group indicate that the physicians look more at the patient during the virtual consultations. We will develop an appearance-based approach to gaze estimation that will estimate autonomously the physician's gaze patterns, to compare quantitatively the differences between presential and virtual consultations.

Keywords: Gaze estimation, Physician gaze, Automatic labelling, Clinical Interaction

Contents

List of Tables

List of Figures

Nomenclature

CNN	Convolutional Neural Network
MAML	Model-Agnostic Meta-Learning
MLP	Multilayer Perceptron
PoR	Point of Regard
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

The physician-patient relationship is a crucial component of the effectiveness of any health care system. Communication is one of the main components in a good physician-patient relationship where communication during medical interviews plays one of the most prominent roles.

The importance of non-verbal communication has gathered more and more attention from research studies that analyze the relationship between patient satisfaction and physician behaviour. A general practitioner can conduct between 120,000 and 160,000 interviews during a 40-year career [4]. For which good communication is essential for managing chronic diseases like diabetes, hypertension, and others. In addition, patients are more adherent to medical recommendations and healthy behavioural changes when they are more informed and involved in the decision-making. Several studies indicate that the non-verbal cues of the physician are one of the most critical aspects of physician-patient communication [15, 24, 16, 1].

Among the non-verbal behaviour, the gaze is one of the most important cues to analyze in non-verbal communication. Studies have concluded that there is a positive correlation between patient satisfaction and the amount of eye contact between the physician and the patient [1]. The amount of time the physician is gazing at the patient and not at the patient's health records on the screen can improve the patient's perception and cognitive functioning [33]. In virtual consultation, a preliminary observation made by the Luz Saúde group during 2019 noted a rise in the eye contact between the patient and the physician compared to the presential consultations. This observation needs to be quantified and objectified clearly and scientifically.

The works on this topic use manual annotation systems to quantify non-verbal behaviours. This process is costly and laborious, which is not convenient. Recently, some methods for automated annotation systems have been proposed [14, 33]. However, they were done in a very constrained environment which wouldn't translate well to other consultation offices/setups. Nonetheless, they provide the first approaches to automatic classification of physician's gaze during medical appointments.

Communication in the physician-patient relationship can be divided into verbal and non-verbal communication. Verbal communication can be defined as communication behaviour with linguistic content [24]. This can be classified according to the model described by Bird and Cohen-Cole model [5]. Ac-

cording to this model, we can classify verbal interactions into three key functions: data gathering to understand the patient (gathering information), development of a rapport and responding to the patient's emotions (developing therapeutic relationship), and patient education and behavioural management (decision making and management). Non-Verbal communication can be defined as communication behaviour without linguistic content and is typically distinguished by which part of the body is being used to express the behaviour. Face non-verbal behaviours include smiling, gazing, frowning, eyebrow-raising, and facial expressivity. Body non-verbal behaviour is expressed through posture or gestures. Vocal non-verbal behaviour includes loudness, voice pitch, monotony, and speech rate.

This work will focus on non-verbal communication and its impact on the physician-patient relationship. The amount of studies published concerning this topic has been increasing in recent years [15, 24, 16, 1], all of them point to the great importance non-verbal behaviours have in patient satisfaction. Specifically, we will focus on the automated analysis of the gaze direction of the physician during the medical interview, classifying according to whether the physician is looking at the patient or not. It will aim to support the automated analysis of primary care visits, helping to support effective communication in the physician-patient relationship.

Additionally, with the advent of the COVID-19 pandemic, there was a rise in the use of virtual consultations. This provides an entirely different environment for the physician-patient relationship. Therefore we will also aim to compare the gaze from the physician between the presential and virtual consultation medium.

This introductory chapter starts by placing the thesis in the context of this research topic. Followed by a topic overview and the objectives, where the intent of the thesis is clarified. The chapter ends with an outline of the thesis.

1.1 Objectives

The main objective of this thesis is to verify, quantitatively, the proportion of time, during a consultation, the physician spends looking at the screen and the patient. Additionally, we also want to compare how this proportion changes when changing from a presential consultation environment to a virtual consultation environment.

The plan, as previously mentioned, is to use a gaze estimation approach to estimate the gaze of the physician. The main objectives are as follows

1. Physician's gaze estimation - In the context of an actual consultation, use a gaze estimation method to estimate the physician's gaze from a video of the physician.

2. Classify physician's gaze and quantify it - After estimating the gaze direction, we will classify it according to where the physician is looking, either at the patient or at the screen.

3. Comparison between presential and virtual consultations - With the gaze classified and quantified, we will compare the results between the virtual and presential consultation environment.

1.2 Thesis Outline

Chapter 2 begins with an explanation on how neural networks work and how to estimate their parameters. With the concept of neural networks introduced, we explain a particular type of neural network used in state-of-the-art gaze estimation methods, the Convolutional Neural Networks (CNN). Then we introduce the concepts of camera intrinsic and extrinsic parameters needed to perform the gaze estimation task. In the end, we move on to gaze estimation methods. First, we explain the different types of gaze estimation tasks. After this, we introduce the different approaches made to solve these gaze estimation tasks. Then we compare the performance of the different approaches.

Chapter 3 contains a description of the presential and virtual consultations and the different consultation rooms (environments) in which they can occur. Then, we explain the software toolkit used to perform the gaze estimation along with any auxiliary methods. After this, we explain our methodology for data acquisition and processing. Then, the planning for the implementation of the thesis is presented. In the end, some preliminary experiments with the software toolkit are shown.

Chapter 2

Background and State of the art

This section of the report will overview the different techniques previously used in human gaze estimation tasks. It starts by giving a theoretical background on neural networks and camera intrinsic and extrinsic parameters used in state-of-the-art gaze estimation. Then it moves on to explaining the different types of existing gaze estimation methods.

2.1 Neural Networks

A Neural Network, also called Artificial Neural Network, is a computing system loosely based on the structure of animals brains. A Neural Network can be defined as a conjunction of connected nodes, each node is called a *neuron* or *perceptron* and each connection between nodes is called an *edge*, each edge is weighted by a *weight*.

Neuron

The fundamental building block of a Neural Network is called a Neuron. A single Neuron possesses a set of ***m*** inputs, $x_1, x_2, ..., x_m$ and an output $\hat{y}$. Each input x_i is weighted by its corresponding weight w_i. A Neuron converts the set of inputs $x_1, x_2, ..., x_m$ into output $\hat{y}$ by propagating the inputs through the neuron itself. During the propagation, the neuron executes two different operations. The first operation is the linear combination of the inputs weighted by their weights. The second operation consists of taking the result of the linear combination and passing it by a non-linear activation function f. Typically used activation functions include the sigmoid function, the hyperbolic tangent and the ReLu function. The output $\hat{y}$ is the result of these two operations done sequentially and is given as:

$$\hat{y} = f\,(\sum_{i=1}^{m} w_i\, x_i\ + w_0) \tag{2.1}$$

The weight w_0 is known as the *bias* of the neuron or threshold. Essentially, propagating data forward through the network is just applying sequentially linear transformations and activations (non-linear transformation) on that data. A neural network with just one neuron just applies a linear transformation, which

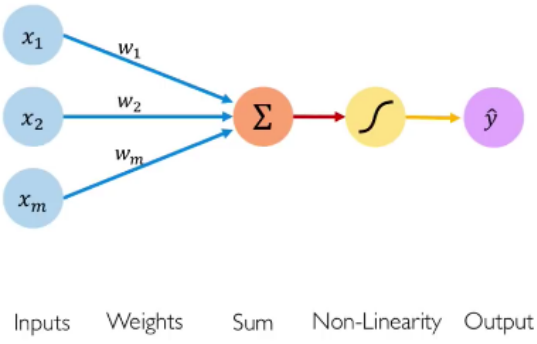


Figure 2.1: Single Neuron Diagram. [1]

is not very powerful. However, connecting multiple neurons in layers enables the network to perform non-linear transformations and, consequently, gain the ability to learn non-linear and linear relationships between the inputs. The process of feeding the input data and propagating it through the network is called forward propagation.

Neurons give the ability to neural networks of performing two types of tasks: regression and classification. In neural networks, a regression task implies that the output of the network is a continuous value. It comes directly as the output of a neuron or set of neurons (depending on how many output variables there are). For example, a regression task can be estimating the gaze direction of a human. On the other hand, a classification task implies that the neural network classifies the input into a set of output classes. An example of a classification task would be the classification of a person on whether she is happy or sad according to its facial image, where happy and sad are both output classes.

Multilayer Perceptron

One of the simplest forms of Neural Networks are Multilayer Perceptrons (MLPs). In MLPs, the neurons are structured by layers where neurons' output from one layer serves as input of neurons in the next layer.

The first layer is called the input layer. Each neuron of the input layer has one input. The inputs of the neurons correspond to the input data for that task and vary from task to task. The middle layers are called the hidden layers. Each hidden layer's neurons have a weighted connection to neurons in the next layer. The last layer is the output layer. The neurons in this layer do not have the activation function. Each neuron in the output layer corresponds to one of the task's outputs.

2.1.1 Supervised Learning

Learning is the adaptation of the neural network to better handle the task at hand. The adaptation is made by changing the connection weights according to sample observations.

[1] https://medium.com/analytics-vidhya/neural-network-part1-inside-a-single-neuron-fee5e44f1e [Date Accessed: 04-05-2021]

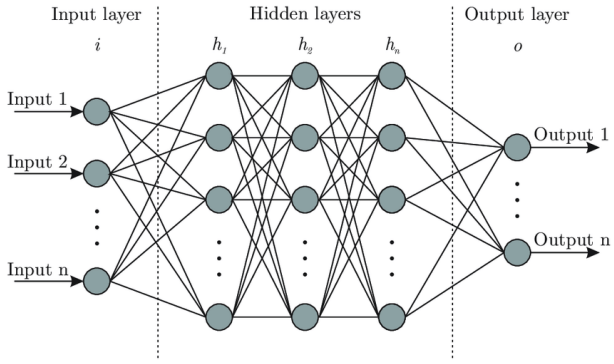


Figure 2.2: Multilayer Perceptron. [2]

There are several learning paradigms that alter the way the network is trained. In other words, the process with which the weights in the network are changed. From the several paradigms: supervised learning, unsupervised learning, reinforcement learning and self-learning, we are going to focus on the supervised learning paradigm.

The supervised learning task is to teach a model how to yield the correct output by using input-output pairs. This set of input-output pairs is the training dataset of the algorithm, where each training example consists of the input labelled with the desired output. This training dataset enables the model to learn over time how to perform a specific task correctly. The model learns by measuring its performance on the training data with a loss/cost function and adjusting itself so that this function's value is minimized. Essentially, the training of the model becomes an optimization problem of minimizing the cost function. This minimization problem is typically solved by applying an iterative method like the Stochastic Gradient Descent method.

In neural networks, the result of the loss function is calculated after the forward propagation of the training data through the network. The update of the weights is done by propagating the result of the loss backwards through the network. More specifically, the gradient of the loss function in respect to the weights of the network is propagated backwards.

Loss function

The loss function measures the performance/accuracy of the model. It can be as simple as the mean squared error (MSE) or something more complex like the binary cross-entropy function. The loss function is parameterized by the parameters of the model being trained. In the case of neural networks, the parameters of the loss function are the weights of the network.

Stochastic Gradient Descent

The Stochastic Gradient Descent (SGD) is an iterative method for minimization of an objective function $Q(w)$ where w are the parameters of the objective function. As the name indicates, the SGD makes use of the gradient of the objective function at each iteration. It takes advantage that the gradient of

[2] `https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051` [Date Accessed: 04-05-2021]

the function at any point always gives the steepest ascent in the function. Therefore the symmetric of the gradient gives the steepest descent. At each iteration t the parameters of the objective function are altered according to the following rule, where η is the learning rate of the algorithm:

$$w_t = w_{t-1} - \eta \nabla Q(w_{t-1}) \tag{2.2}$$

At each iteration i we move the value of the parameters w_i in the direction of the steepest descent, given by the symmetric of the gradient of the function. This way we minimize the value of the objective function in function of the parameters w.

The SGD method is used in Supervised Learning to minimize the loss function in function of the parameters of the model being trained. In neural networks, the objective function $Q(w)$ is the loss function chosen, and the parameters w are the weights of the neural network. The training of a neural network involves applying the SGD to the loss function, obtained after the forward propagation of the training data through the network. Consequently, computation of the gradient of the loss function in relation to each weight of the network is needed. This computation is performed with the Backpropagation algorithm.

Backpropagation

In a neural network, to minimize the loss function, the network alters its weights. The alteration of the weights is done according to the SGD algorithm, calculating the gradient of the loss function in relation to the weights of the network and then updating the weights according to (2.2).

Backpropagation refers specifically to the algorithm used to calculate the gradients of the loss function. Due to the way neural networks operate, the gradient of the loss function in relation to each weight of the network needs to be calculated by propagating the loss backwards through the network with the application of the chain rule. The chain rule is used to calculate the derivative of composite functions, and it is given by:

$$\frac{\partial z}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} \tag{2.3}$$

where $z = f(g(x))$.

Adapting to the case of backpropagation, we first start by defining:

- x - the input of the network.
- y - the desired output of the network.
- C - the cost function
- L - number of layers
- $W^{(l)}$ - the weight matrix of the weights from layer $l-1$ and l .
- $f()$ - The activation function of each node
- $C(y, g(x))$ - The loss function resulting from propagating x forward through the network

The output of the network $g(x)$ can be seen as a composite function of the form:

$$g(x) = W^{(L)} f(W^{(L-1)} \cdots f(W^{(1)} x)) \tag{2.4}$$

In the output layer, the gradient of the loss function in relation to the weights in the final layer can be calculated as:

$$\frac{\partial C(y, g(x))}{\partial W^{(L)}} = \frac{\partial C(y, g(x))}{\partial g(x)} \cdot \frac{\partial g(x)}{\partial W^{(L)}} \tag{2.5}$$

In the input and hidden layers, the gradient will be calculated based on the output of that layer denoted as $z^{(i)}$ where i is the number of the layer:

$$\frac{\partial C(y, g(x))}{\partial W^{(i)}} = \frac{\partial C(y, g(x))}{\partial z^{(i+1)}} \cdot \frac{\partial z^{(i+1)}}{\partial z^{(i)}} \cdot \frac{\partial z^{(i)}}{\partial W^{(i)}} \tag{2.6}$$

By propagating the gradient $\frac{\partial C(y, g(x))}{\partial z^{(i+1)}}$ to the previous layer i, the backpropagation algorithm is able to calculate the gradients in respect to every weight. The update of the weights is done according to the SGD method using (2.2) in the following way:

$$W_t^{(i)} = W_{t-1}^{(i)} - \eta \cdot \frac{\partial C(y, g(x))}{\partial W_{t-1}^{(i)}} \tag{2.7}$$

Comparing with (2.2), the objective function $Q(w)$ corresponds to the loss function $C(y, g(x))$, where $g(x)$ is dependent on the weights of the network like in (2.4). The parameters w of the objective function correspond to the weights of the network $W^{(i)}$.

2.1.2 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a type of Neural Network. CNNs have been gaining popularity in recent years for Computer Vision tasks [2, 29]. CNNs consist of a set of convolutional and pooling layers followed by set of fully connected layers. There are various architectures of CNNs, some examples are LeNet [22], ResNet [17], AlexNet [21], among others. A CNN can be seen as a conjunction of 2 steps: Feature Extraction and Classification. The Feature Extraction step is performed by the convolution and pooling layers. During this step, CNNs take an input image and extract various characteristics (features) from the image. Features can range from something as simple as lines and edges in the picture to more abstract (high-level) features that consist of combinations of simpler features. The Classification step is performed by the Fully Connected Layers, which are, in essence, an MLP.

Convolutional Layer

The Convolutional layer gives the CNN the ability to capture spatial and temporal dependencies in the image through filters and the convolution operation. The filter is a conjunction of kernels, with different kernels being able to extract different features like edges or lines. The convolution operation extracts features from the input image according to the kernel used. Typically the first convolutional layers in the

network extract low-level features such as edges, colour and gradient orientation. Using these features as inputs of other Convolutional layers, the CNN can extract high-level features, combining the previously obtained low-level features. The output of the Convolutional layers is sometimes called a feature map.

Pooling Layer

The Pooling layer is responsible for condensing the convoluted features returned by the Convolutional layer. The main objective of this operation is to reduce the spatial size of the feature maps, leading to the reduction of the computational power required in the processing of the data. Additionally, the pooling operation also extracts dominant features, which are rotational and positional invariant. Pooling works by passing the kernel through the image. There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the kernel. On the other hand, Average Pooling returns the average of the values from the portion of the image covered by the kernel. The Pooling layer is responsible for increasing the efficiency of the CNNs training by reducing the dimensionality of the data and extracting the most dominant features simultaneously.

Fully Connected Layers

The Fully Connected Layers take the feature map returned by the convolutional and pooling layers and output the classification of the input image. This section of the CNN is an MLP that learns how to perform the wanted task with the feature map as its input.

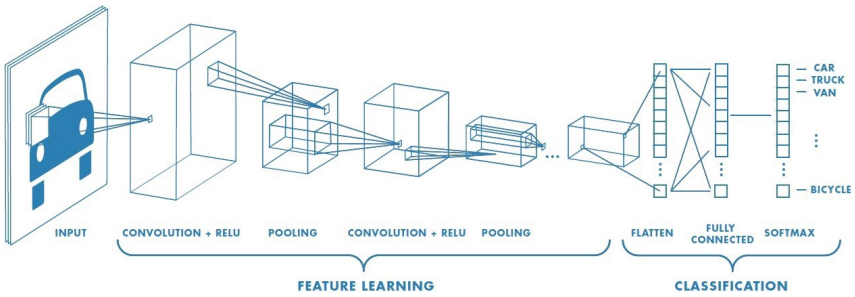


Figure 2.3: Convolutional Neural Networks. [3]

2.2 Camera Parameters

A camera can be described by the pinhole camera model illustrated in Figure 2.4. The pinhole camera model describes the mathematical relationship between the points in the world coordinate system and their projection onto the camera image plane (Pixel Coordinate System). This mathematical relationship is parameterized by the camera's intrinsic and extrinsic parameters.

The intrinsic parameters of the camera describe its internal characteristics such as focal length, skew, distortion and image center. They are necessary to link the pixel coordinates of an image point

[3] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [Date Accessed: 04-05-2021]

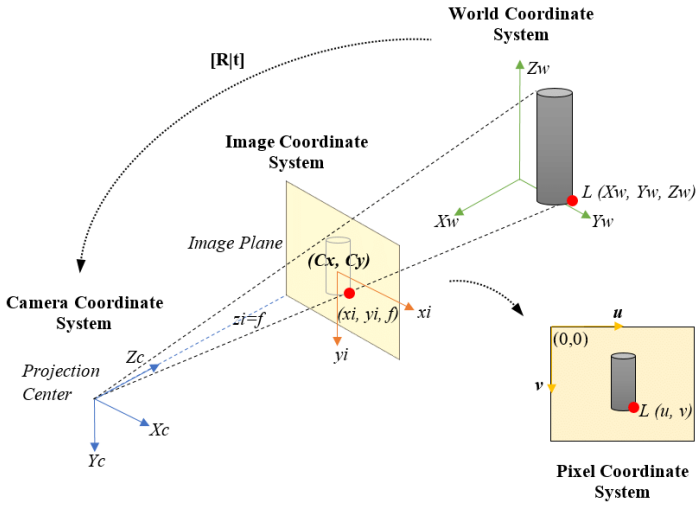


Figure 2.4: Pinhole Camera Model. [4]

with the corresponding coordinates in the camera coordinate system. Essentially, they describe the transformation of points in the Pixel Coordinate System to points in the Camera Coordinate System and vice-versa.

The extrinsic parameters describe the camera's position in relation to the world coordinate system, defining the location and orientation of the camera reference frame in respect to the world reference frame. The extrinsic parameters describe the transformation of points in the world coordinate system to the camera coordinate system and vice-versa.

In gaze estimation, the extrinsic parameters are used to convert 3D gaze predictions (vectors in the Camera Coordinate System) into 2D gaze predictions as *PoR*s (points of regard) on a screen. The screen is described as a plane in the world coordinate system. Using the camera's extrinsic parameters, we can convert the 3D gaze direction in the camera coordinate system to a 3D vector in the world coordinate system. Having the 3D vector and the screen plane both on the world coordinate system, we intersect the vector with the plane to obtain our *PoR* on the screen.

Therefore, we need to know both the intrinsic and extrinsic parameters of the camera. For this, we use two camera calibration routines. To find the intrinsic parameters of the camera, we use the routines offered by the *OpenCV* [6] library. To find the extrinsic parameters of the camera, we use a mirror-based calibration method from [32].

2.3 Gaze Estimation Methods

In the context of a gaze estimation task, gaze should be understood as *gaze direction* or point of regard (*PoR*). *Gaze direction* is represented as a 3D vector in the Camera Coordinate System and the *PoR* is represented as a 2D point in the target Screen Coordinate System. The typical setup (Figure 2.5) for gaze estimation tasks consists of a camera directed at the subject and a gaze target, normally a screen.

[4] `https://www.researchgate.net/figure/Pinhole-Camera-Model-ideal-projection-of-a-3D-object-on-a-2D-image_fig1_326518096` [Date Accessed: 02-06-2021]

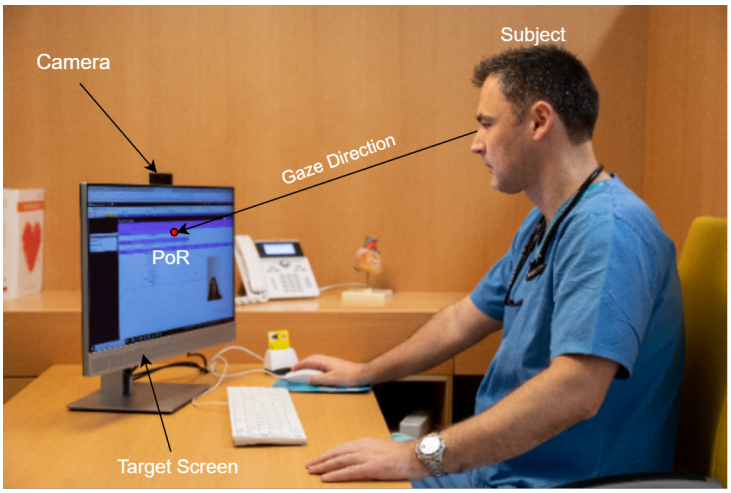


Figure 2.5: Typical Desktop Gaze Estimation Setup.

Therefore, the objective with gaze estimation tasks is to model the relationship between input images of a subject and the gaze direction/*PoR* of that same subject. This divides gaze estimation into two different types depending on the output: 3D gaze estimation (gaze direction) and 2D gaze estimation (*PoR*). Additionally, gaze can be classified according to zones of the screen, this type of methods are classified as gaze zone estimation methods.

2D Gaze Estimation

In 2D gaze estimation, the task is to regress from the input image of the subject to a 2-dimensional on-screen gaze location $\boldsymbol{p}$ as $\boldsymbol{p} = f(\boldsymbol{I})$, where f is the regression function and $\boldsymbol{I}$ is the input image [20, 28, 37, 36, 23]. The point $\boldsymbol{p}$ is typically defined in the screen coordinate system. The screen is defined as a virtual plane in the camera coordinate system. The regression function typically needs the input image and additional information like 3D head pose [19] and face bounding boxes locations [20].

One limitation of 2D gaze estimation comes from the fact that it assumes that the screen and camera have fixed positions in relation to each other. In other words, the target screen plane is fixed in the camera coordinate system. Consequently, 2D gaze estimation tasks do not allow for free camera movement after training the system. Another limitation for 2D gaze estimation comes from another assumption that the camera's intrinsic parameters are always the same. Therefore, a trained regression function cannot be applied to different cameras.

3D Gaze Estimation

In 3D gaze estimation, the task is to regress from the input image $\boldsymbol{I}$ to a 3-dimensional vector $\boldsymbol{g}$, as $\boldsymbol{g} = f(\boldsymbol{I})$, in the camera coordinate system. The function f, optionally, takes the 3D head pose as an additional input apart from the input image [35]. The gaze direction output is a vector typically represented as a unit vector originating from a point of reference in the image, like center of the eyes [28, 31, 23, 11] or center of the face [19, 36, 37]. Additionally, the 3D gaze estimates can be converted into 2D estimates by projecting the gaze vector onto the target plane (screen plane) in the camera

coordinate system.

Gaze Zone Estimation

The fundamental difference between this type of estimation and the previous two is the output of the method. Instead of outputting the gaze direction or *PoR*, the gaze zone estimation methods output the zone at which the subject is looking. In other words, the gaze zone estimation task is a classification task, unlike the 2D and 3D gaze estimation tasks, which are regression tasks. Usually, the zones are regions of interest on the screen [8, 34, 13], but they can also be a binary value of whether the subject is looking at a target or not [33, 14]. The applications mentioned in Chapter 1 to classify the physician's gaze are of this type of method. However, these methods are not robust for "in the wild" gaze estimation due to only being applicable to the environment in which they were trained.

For this type of approach the training is done as a classification task. The function that is learned by the network now is the classification of an input image $\boldsymbol{I}$ into a set of n output classes $\boldsymbol{c} = \{c_1, c_2, ..., c_n\}$ as $\boldsymbol{c} = f(\boldsymbol{I})$. It is due to this type of training that gaze zone estimations can't be applied in environments where they were not trained.

2.3.1 Feature-based Approaches

Feature-based approaches use eye features, such as corneal reflections caused by reflections of external light sources on the cornea, to perform the gaze direction regression. They are the best performing methods. However, to achieve this performance, they require expensive special hardware not accessible to everybody. In addition, they require lengthy calibration sequences, which is not convenient for the user. This type of approach is employed by commercial eye trackers like the entry-level eye tracker Tobii EyeX.

Figure 2.6: Tobii Eye Tracker. [5]

[5] https://www.ign.com/articles/2017/08/10/tobii-eye-tracker-4c-review [Date Accessed: 03-05-2021]

2.3.2 Model-based Approaches

Model-based approaches infer gaze from a 3D model of the head and/or eye. This model is created from the input image by detecting features related to the eye shape like: pupil center, eye corners, iris edges and facial feature pose. These features are used to locate the position and orientation of the head and the eye in the camera coordinate system. This information can then be used to fit the 3D geometric eyeball model.

After estimating the parameters of the 3D model, the calculation of the final gaze direction can be done by modelling the eye as a sphere in the following way according to [7]. Denoting:

- $EyeCenter = (x_{EyeCenter}, y_{EyeCenter}, z_{EyeCenter})$ - The 3D coordinates of the center of the eye, on the eye sphere surface (see Figure 2.7).
- $EyePupilCenter = (x_{EyePupilCenter}, y_{EyePupilCenter}, z_{EyePupilCenter})$ - 3D coordinates of the center of the eye's pupil (see Figure 2.7).
- $EyeballCenter = (x_{EyeballCenter}, y_{EyeballCenter}, z_{EyeballCenter})$ - 3D coordinates of the center of the sphere that models the eye (see Figure 2.7).
- $Radius_{EB}$ - eyeball radius.
- (ω_x, ω_y) - pitch and yaw head pose angles.

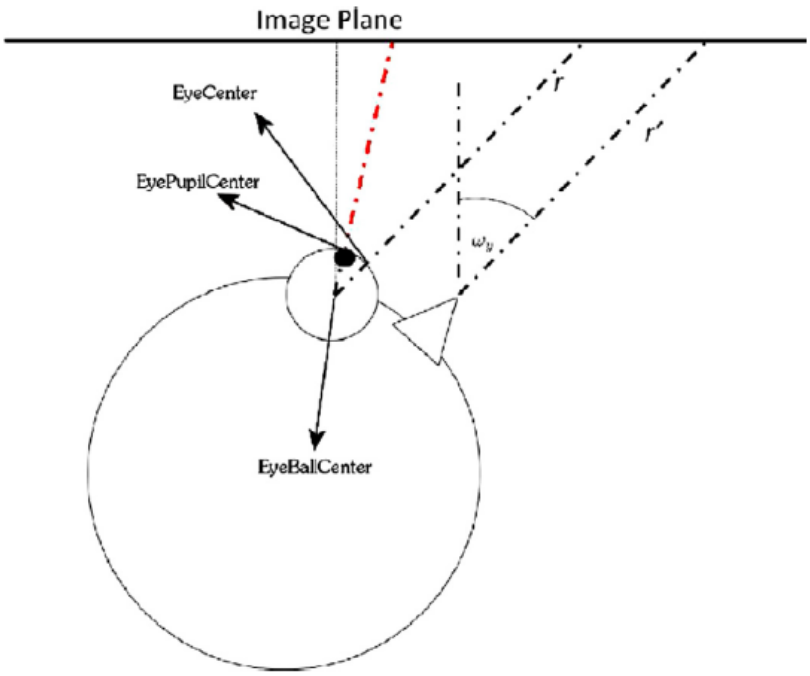


Figure 2.7: 3D Eye Model for [7].

The head pose in the camera coordinate system (the exact position of the head in relation to the camera

described by the (ω_x, ω_y) angles) can be integrated like in [7] to calculate the $EyeballCenter$:

$$EyeballCenter = \begin{cases} x = x_{EyeCenter} \pm |Radius_{EB} \cos\omega_x \sin\omega_y| \\ y = y_{EyeCenter} \pm |Radius_{EB} \cos\omega_x \sin\omega_y| \\ z = Radius_{EB} \cos\omega_x \cos\omega_y + z_{EyeCenter} \end{cases} \tag{2.8}$$

With the $EyeballCenter$, the coordinates on the screen plane can be computed like:

$$\begin{cases} x_{IP} = \frac{z_{EyeballCenter}}{z_{EyeballCenter} - z_{EyePupilCenter}} (x_{EyePupilCenter} - x_{EyeballCenter}) \\ y_{IP} = \frac{z_{EyeballCenter}}{z_{EyeballCenter} - z_{EyePupilCenter}} (y_{EyePupilCenter} - y_{EyeballCenter}) \end{cases} \tag{2.9}$$

Another way to model the eye would be by using a two sphere eyeball model, which is closer to the real anatomy of the eye like in [9]. The line that is traced with (2.9) is now traced with some optional parameters that are denoted as:

- C_0 - Corneal eye center.
- K_0 - Distance between Corneal eye center (C_0) and $EyePupilCenter$
- K - Euclidean distance between the $EyePupilCenter$ and the eye ball center of the biggest sphere denoted as C

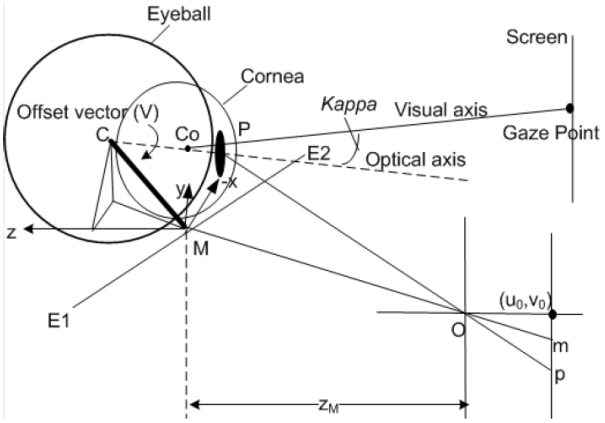


Figure 2.8: 3D Eye Model for [9].

The corneal center is calculated in the following way:

$$C_0 = C + \frac{K_0}{K}(EyePupilCenter - C) \tag{2.10}$$

The visual axis can be traced by drawing a straight line from C_0 to $EyePupilCenter$ as illustrated in Figure 2.8, where P is the $EyePupilCenter$. Additionaly, using the approach in [9], person-dependent calibration can be added to correct the angular difference between the optical axis and the visual axis, leading to better performance.

One drawback of model-based methods is that it is difficult to estimate the 3D parameters precisely. One small error propagates in space leading to highly inaccurate estimations in those cases. Early versions of this methods needed high resolution cameras or infrared light sources to capture the eye accurately. Nowadays, methods use machine learning to perform feature detection enabling the use of lower quality images.

2.3.3 Appearance-based Approaches

Appearance-based approaches learn to map gaze direction directly from input images using machine learning techniques. It can use images with lower resolution and quality than the ones needed for the model-based methods. The need for highly accurate modelling of the eyeball based on eye features has been removed thanks to machine learning techniques. Nowadays, using large-scale datasets and deep learning, gaze estimation accuracy has significantly improved in more challenging real-world situations ([20, 19, 37, 36, 27, 28]). Appearance-based approaches can provide 2D gaze estimation or 3D gaze estimation. Some provide both types of estimation being able to convert from one to the other.

Gaze Estimation Datasets

Most of the appearance-based approaches referenced use deep learning architectures to perform the gaze estimation task [20, 23, 19, 28, 37, 35, 36, 11]. Consequently, different datasets for gaze estimation have been developed. They are represented in Table 2.1. Many of these are geared with desktop or smartphone gaze tracking in mind. They are captured with a static camera setup [12, 30, 28, 35, 36] or with a camera integrated into a mobile device [20]. The static approach leads to higher accuracy but is more limited in terms of variation of illumination and motion blur. The datasets geared towards mobile gaze tracking offer more variety in number and variety of subjects. Some datasets also contain additional information like the 3D head pose [35, 30, 12, 31].

Table 2.1: Datasets

Ref.	Dataset	Type of content	# Subjects	#Samples	Gaze
[35]	MPIIGaze	Face + Eye crops	15	213,659	3D
[19]	Gaze360	Face + Full body	238	172,000	3D
[12]	EYEDIAP	Face + Eye crops + Depth data	16	62,500	3D
[31]	UT Multiview	Face +Eye crops	50	64,000	3D
[30]	Columbia	Face	56	5,800	3D
[11]	RT-GENE	Face + Wearable device img + Depth data	15	122,533	3D
[28]	EVE	Face + Screen Content	54	12,308,334	3D
[20]	GazeCapture	Face	1,450	2,129,980	2D

The Columbia dataset [30] consists of 5880 images from 56 participants. It covers 5 different head poses and 21 gaze directions. Participants were asked to sit using a chin rest, with a grid of dots attached to the wall in front. This lead to limited variation in appearances. It was created with the idea of *gaze locking* in mind, which consists of sensing eye contact directly from an image in a passive, appearance-based manner. It also did not need any active illumination (like infra-red illumination), which was something commonly used at the time and reduced accessibility of gaze tracking methods.

The EYEDIAP dataset [12] is composed of data obtained from 16 people in a total of 94 sessions with a Microsoft Kinect and HD camera synchronized with 5 leds. Participants sat in front of the setup and looked at continuous and discrete targets on a computer screen. The EYEDIAP dataset was proposed as the first benchmark dataset for gaze estimation from remote RGB and RGB-D images. It allowed for different methods to be compared and identify the advantages and disadvantages of each method.

The UT Multiview dataset [31] is composed of video sequences of 50 participants from 8 different views. The 2D and 3D facial landmarks are annotated along with the ground-truth gaze. It was created with the intent of eliminating the requirement for person- and session-dependent training for appearance-based 3D gaze estimation method. It has a wide variety of head poses, gaze directions and subjects.

The MPIIGaze dataset [35] was created for ”in the wild” gaze estimation when most of the existing datasets, at the time, contained low variety in head poses variation and illumination conditions. The EYEDIAP and the UT Multiview datasets were the only ones that had significant head pose variety, yet they lacked variety in illumination conditions since they were captured under laboratory conditions. The MPIIGaze dataset contains 213,659 images from 15 participants collected during natural everyday laptop use. It is significantly variable in relation to appearance and illumination.

The GazeCapture [20] is a large scale dataset made especially for *PoR* estimation in mobile devices, but it can be extended to desktop environments. There was a need for a dataset with a large number of subjects, with high head pose and illumination variety, since the existing datasets did not provide enough subject variety. Consequently, crowdsourcing was used to build the dataset, addressing the lack of variety problem. During the data capture, subjects were encouraged to move their head to increase variety in head poses. The GazeCapture dataset is composed of 1450 people with unconstrained head motion and a wide range of backgrounds and illumination.

The RT-Gene dataset [11] is geared for dealing with high subject to camera distances and high variation in head poses and eye angles. It was the first work to try and address the problem of high subject to camera distances. To build the dataset, RGB-D data of the subject wearing a mobile eye tracker is recorded. To avoid the changes in human appearance caused by the mobile eye tracker, semantic inpainting was used in the regions covered by the eye tracking glasses.

The Gaze360 dataset [19] is a large-scale dataset for robust 3D gaze estimation in unconstrained environments. Most of the datasets developed until then were geared towards desktop or smartphone gaze tracking, all of them had a static setup with a fixed recording setup. For this dataset, the ground-truth was calculated by placing a panoramic camera at the center of the scene between the subjects and a large rigid target marked with an AprilTag [25]. The Gaze360 dataset focused on containing a wide range of indoor and outdoor environments. Since it was not constrained to a static setup to capture

data, the Gaze360 dataset has a variation of head pose much higher than any of the previous datasets.

The EVE dataset [28] is collected from 54 participants from 4 camera views. It contains over 12 million frames adding up to 105 hours of video data. A unique particularity of the EVE dataset is that along with the subject video data, the viewer's content on the screen was captured simultaneously. The screen content is denominated *visual stimuli*. The intent is to refine the initial estimate by using the *visual stimuli* presented to the subject at the estimate's time. Currently, the EVE dataset is the largest and most recent gaze estimation dataset.

iTracker Model

Along with the GazeCapture dataset, the work of [20] also proposes the iTracker model for robust gaze prediction. The iTracker model is an end-to-end CNN architecture for *PoR* estimation, specifically on mobile devices. The goal of the iTracker model was to design a method that performed robust gaze estimation from a single image.

The model's inputs were: the image of the face, its location in the image (face grid), and an image of each eye. The model learns the head pose relative to the camera from the face image and the face grid. The eye crops enable the model to learn the pose of the eyes relative to the head.

The model's output is given as the distance of the *PoR* relative to the camera in centimetres. This enables the model to be used with various mobile devices by leveraging the notion that the camera is in the same plane as the screen.

The architecture of the CNN is based on the AlexNet architecture [21], and it is represented in Figure 2.9. The *CONV* layers represent the convolutional layers, while the *FC* layers represent fully connected layers. The Convolutional layers that process the 2 different eye images have shared weights, while the face image is processed by another set of convolutional layers. While testing the model, the authors

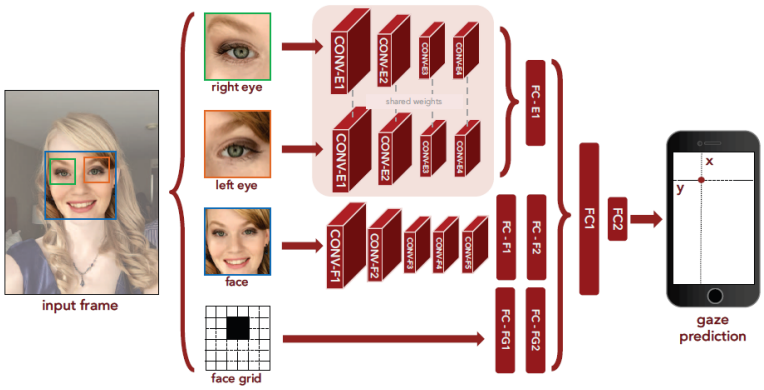


Figure 2.9: iTracker Architecture proposed by [20]

verified that fine tuning the network to each device and orientation helped the performance of the model, which was useful for dealing with unbalanced data on the GazeCapture dataset.

Full-Face Gaze Estimation with a Spatial Weights CNN Architecture - MPIIFaceGaze

The authors of the work in [36] wanted to evaluate how valuable the information contained in the facial appearance of the subject, apart from the eyes, would be for a gaze estimation model. Therefore they propose a spatial weights CNN architecture for full-face appearance-based 3D and 2D gaze estimation, denominated MPIIFaceGaze. The input of the CNN would be a full face image, and the output would be either: a gaze direction for 3D gaze estimation or a *PoR* for 2D gaze estimation. The representation of the architecture can be seen in Figure 2.10. The objective with the spatial weights mechanism is to

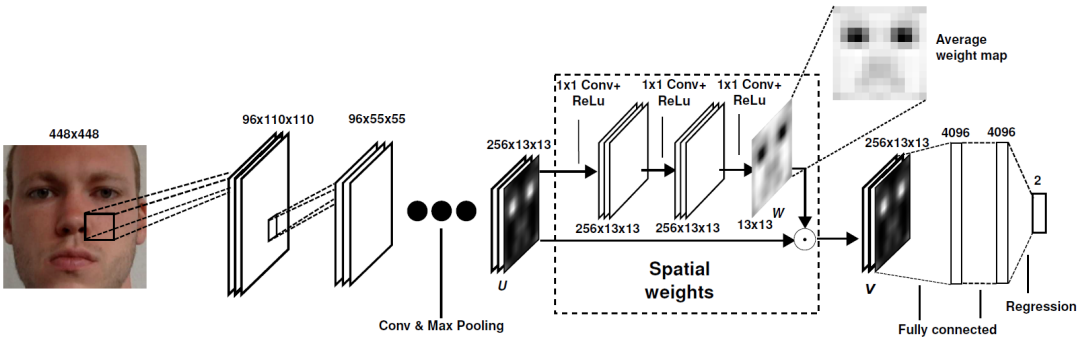


Figure 2.10: Spatial Weights CNN for full-face appearance-based estimation from [36].

force the network to learn and understand that different regions of the face have different importance for the estimation task in a given sample. The mechanism consists of an additional 3 Convolutional layers with 1x1 kernel size followed by ReLU activation layers. The input of the additional convolutional layers is the activation tensor $\boldsymbol{U}$ with size $N \times H \times W$ $(256 \times 13 \times 13)$. N is the number of feature channels, H and W are the weight and width of the output. The activation tensor is passed through the additional layers to yield the spatial weights matrix $\boldsymbol{W}$. After this, a element-wise multiplication described by (2.11) between $\boldsymbol{W}$ and $\boldsymbol{U}_c$, where $\boldsymbol{U}_c$ is the c-th feature channel of $\boldsymbol{U}$, is performed yielding $\boldsymbol{V}$.

$$\boldsymbol{V}_c = \boldsymbol{W} \odot \boldsymbol{U}_c \tag{2.11}$$

The weighted activation tensor $\boldsymbol{V}$ is then fed into the fully connected layers of the network leading to the final estimate of the gaze direction. The *PoR* estimate of the model is obtained by converting the gaze direction vector in the camera coordinate system to a 2D point in the screen coordinate system. This is done by intersecting the gaze vector with the screen plane in the camera coordinate system. The camera-screen relationship can be obtained through a mirror-based camera-screen method like in [32]. Using this relationship, we can find the target screen plane in the camera coordinate system and intersect the vector obtained from the model.

This method is implemented by the *OpenGaze* software toolkit proposed in [37]. *OpenGaze* provides a full gaze estimation pipeline for 3D gaze estimation. Additionally, when provided with the intrinsic and extrinsic camera parameters, *OpenGaze* can project the 3D gaze directions into 2D estimates on the target screen. *OpenGaze* is the first software toolkit developed for appearance-based gaze estimation and interaction.

Few-Shot Adaptive Gaze Estimation - FAZE Architecture

The work in [27] tackles the problem of person-specific gaze estimation. High accuracy gaze estimation is difficult to perform, often requiring personal calibration sequences to fine tune the network to each subject. Person-independent gaze estimation, is not currently suitable for applications which need high accuracy estimations. Therefore they propose the $FAZE$ framework for learning gaze estimation networks for new subjects using few calibration samples (Figure 2.11).

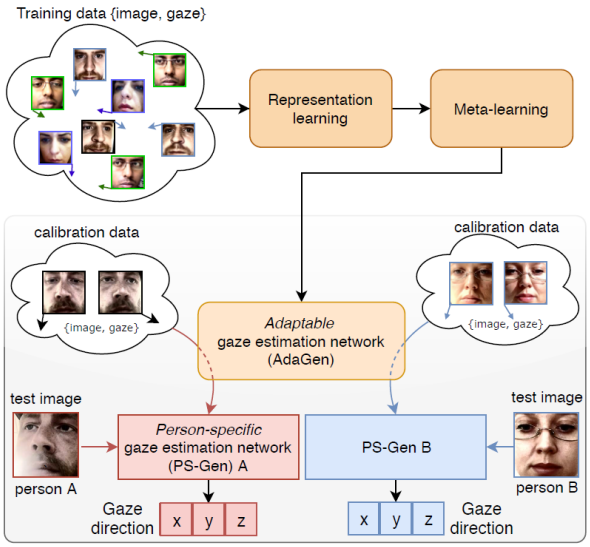


Figure 2.11: Overview of the $FAZE$ framework from [27]

The $FAZE$ framework needs to encode factors specific to the subject, but at the same time, it needs to leverage the observations it made of eye region appearance variation across a large variety of subjects with different head pose and gaze direction configurations. Consequently, the first step of the framework is to learn a generalizable latent embedding space, encoding meaningful information about gaze direction, including person specific traits.

To learn the latent embedding space $FAZE$ uses a transforming encoder-decoder architecture, denominated Disentangling Transforming Encoder-Decoder (DT-ED), which will consider 3 different factors: gaze direction, head orientation and appearance of the eye region in given images (Figure 2.12).

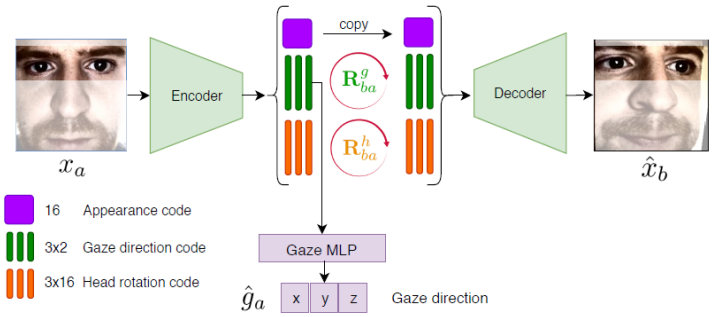


Figure 2.12: Disentangling Transforming Encoder-Decoder Architecture (DT-ED) from [27]

The encoder $\mathcal{E} : \boldsymbol{x} \rightarrow \boldsymbol{z}$ encodes image $\boldsymbol{x}$ into the latent embedding space $\boldsymbol{z}$, and the decoder

$\mathcal{D} : \boldsymbol{z} \rightarrow \hat{\boldsymbol{x}}$ decodes the sub-codes from the latent embedding space into an image $\hat{\boldsymbol{x}}$. The latent space embedding $\boldsymbol{z}$ is formed by 3 parts: the appearance code $\boldsymbol{z}^a$, the gaze direction code $\boldsymbol{z}^g$ and the head pose $\boldsymbol{z}^h$, hence $\boldsymbol{z} = \{\boldsymbol{z}^a\,;\,\boldsymbol{z}^g\,;\,\boldsymbol{z}^h\}$. The job of the encoder $\mathcal{E}$ can be explained as the conversion of the input image into 3 sub-codes that represent the appearance, gaze direction and head rotation of the subject in that image. The decoder $\mathcal{D}$ does the inverse operation of the encoder, it takes in the 3 sub-codes and it converts them into an image of the subject. The output image of the decoder is the same as the input image of the encoder, as long as the output sub-codes of the encoder are not altered. Both the $\mathcal{E}$ and $\mathcal{D}$ were implemented as CNNs, specifically with the DenseNet architecture [18]. The gaze estimation using this latent embedding space is done with a MLP denominated GazeMLP. GazeMLP takes as input the gaze direction sub-code $\boldsymbol{z}^g$ and outputs the final gaze direction estimate $\hat{\boldsymbol{g}}$.

The DT-ED provides a robust feature extractor capable of providing good and consistent features. In practical setups, only a few calibration samples are available. However, neural networks tend to overfit when fine-tuned with few calibration samples. Therefore, the $FAZE$ framework uses meta-learning algorithms, which can be leveraged to learn few-shot person-specific gaze estimators that generalize better to new people without overfitting.

The meta-learning algorithm chosen was the MAML (Model-Agnostic Meta-Learning) algorithm [10], which learns a set of initial network weights (Gaze MPL weights) optimal for fine tuning without the overfitting problem. This produces a highly Adaptable Gaze Estimation Network (AdaGen). The MAML algorithm has the objective of training a model on a variety of learning tasks, such that it can adapt to new learning tasks using few training/calibration samples. In the $FAZE$ case, the different learning tasks are the gaze estimation for different subjects. Using MAML, the model parameters (the GazeMLP weights) are set in such a way that a few number of gradient steps with training/calibration data from a different subject will lead to a good generalization performance.

After obtaining the AdaGen, the only step remaining is the adaptation of the weights learnt using MAML to produce a person-specific gaze estimator (PS-GEN). The process of obtaining the PS-GEN from the AdaGen consists in fine tuning the network parameters with person-specific calibration samples. In the end, we have a person-specific gaze estimator with just a few calibration samples capable of performing gaze estimation tasks at a state-of-the-art level.

2.3.4 Performance Comparison

The work from [37] studies the gap between the state-of-the-art of the 3 different types of approaches of gaze estimation methods. Feature-based approaches were represented by the commercial eye tracker Tobii EyeX, the model-based approaches were represented by the GazeML [26] method, and the MPIIFaceGaze from [36] represented the appearance-based methods. The method's accuracy was evaluated in function of the distance from the screen and number of calibration samples provided. The results are presented in Figure 2.13.

In terms of distance from the screen, the Tobii EyeX was the best performing method. However, it is only able to give estimates when the subject is located 50cm to 75cm from the screen, making it very

limited and unsuited for unconstrained settings. The appearance-based method performs well across all distances, always outperforming the model-based method. The model-based method could not perform well when the subject was placed further from the screen.

In terms of calibration samples, the Tobii EyeX was the best performing method again, outperforming the appearance-based method. However, we can see that appearance-based methods can achieve accuracy competitive with the Tobii EyeX with 4 calibration samples. The model-based approach lagged behind the others. In addition, we can see that the calibration-free accuracy is basically the same between the Tobii EyeX and the appearance-based approach.

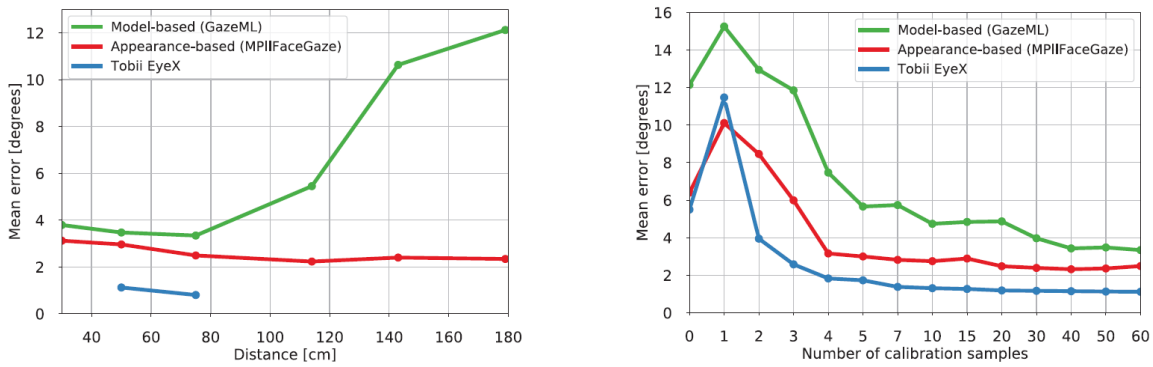


Figure 2.13: Evaluation of different approaches of gaze estimation from [37]

From this comparison, we can conclude that appearance-based approaches can be competitive with the commercial eye trackers in terms of accuracy and are robust across a range of distances, being much more suited to ”in the wild” gaze estimation.

The works from [36] and [27] both compared its models with the state-of-the-art model at the time of publishing. The work [36] compares its MPIIFaceGaze method with the iTracker model from [20]. The results showed that the MPIIFaceGaze method achieved a significant performance improvement over the iTracker model.

The work from [27] compares its $FAZE$ framework with the MPIIFaceGaze method from [36]. The results showed that the $FAZE$ framework needed much fewer calibration samples to converge to lower accuracy than the MPIIFaceGaze. However, the calibration-free accuracy of the MPIIFazeGaze method was slightly better than the $FAZE$ network.

2.4 Conclusion

In this chapter, we went over the three different types of approaches used for gaze estimation: feature-based, model-based and appearance-based. Overall, feature-based methods, which consist of the commercial eye trackers available to the public, have the best performance. However, this type of approach requires special hardware (normally expensive) and is not flexible, only being applicable to very constrained environments. In recent years, due to advancements in neural network technology, appearance-based approaches relying on neural networks, mainly CNNs, have been able to reach per-

formances competitive with commercial eye trackers (feature-based approaches). This type of approach can be applied to various environments, being much more flexible than commercial eye trackers. Additionally, it does not require special hardware making appearance-based approaches much more accessible than feature-based approaches. Model-based approaches, even though they don't require special hardware, have significantly worse performance than the other 2 types of approaches.

In addition, three different appearance-based methods have been analyzed: the iTracker model, MPIIFaceGaze and the $FAZE$ framework. The iTracker model, being the oldest of the three, is the worst performing one. Although it was specially designed to perform 2D gaze tracking on mobile devices, while the other two perform 3D gaze estimation and, optionally, convert it to 2D gaze predictions using the extrinsic camera parameters. The $FAZE$ framework was the best performing appearance-based method and only needed a few calibration samples to reach this performance. However, the MPIIFaceGaze method had slightly better calibration-free accuracy (no calibration samples given) than the $FAZE$ framework.

For this work, we are going to utilize the *OpenGaze* software toolkit since appearance-based approaches are much more accessible and flexible than model- or feature-based approaches, while being competitive in terms of performance. From the appearance-based approaches available, we chose the *OpenGaze* software because it provides a full gaze estimation pipeline and it does not require any calibration sequence along with being easy-to-use. Additionally, the *OpenGaze* software can convert 3D gaze estimations to 2D gaze estimations, which is essential for the classification of the doctor's gaze.

Chapter 3

Implementation

The goal of this project is to quantify the amount of time a doctor is looking at a patient, and not at the computer screen, during virtual consultations and presential appointments. Additionally, we want to compare both types of consultations to verify quantitatively the preliminary observations made by the Luz Saúde group.

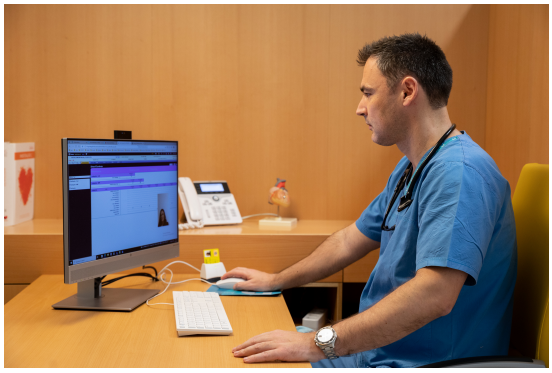
In this chapter, a description of the proposed approach that will be developed is presented based on the work outlined in Chapter 2. Section 3.1 describes the 2 different types of consultations that are going to be the target of the study and the possible environments for each of them. Section 3.2 explains the gaze estimation method which is going to be used and how it is implemented. Section 3.3 explains the methodology used for the study. And finally, section 3.4 breaks down the work necessary to implement this approach during the thesis and contains the timeline of the thesis. Section 3.5 will present some preliminary experiments made using the gaze estimation software.

3.1 Consultation Environments

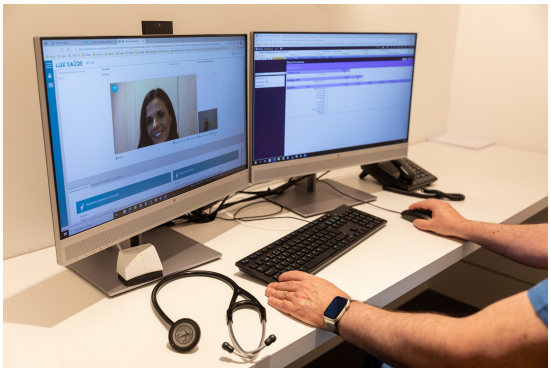
In this study, we are going to analyze the gaze of the doctor in both presential and virtual consultations. In presential consultation, the doctor and the patient are in the same room, the consultation room. During this type of consultation, the doctor's gaze will typically shift between the patient and the screen, which contains the patient's health records. In virtual consultations, the patient and the doctor communicate through a video call and are not in contact physically, the patient is seen as a video feed on the doctor's screen. During virtual consultations, the doctor's gaze will also shift between the patient's and its health records.

The consultations will occur in 2 different types of consultation rooms: the hybrid consultation room and the virtual consultation room. The hybrid consultation room allows the doctor to perform both virtual and presential consultations. During presential consultation, the patient is in the room with the doctor and the patient's health records are shown on the screen. During virtual consultation, the patient's image is in a window within the doctor's computer screen (*Picture in Picture*). The virtual consultation room is specially designed for virtual consultations. It has two different screens, one where the patient's video

feed is shown and another which shows the health records of the patient.



(a) Hybrid Consultation Room



(b) Virtual Consultation Rooms

Figure 3.1: Consultation Rooms Setups.

In each of the consultation rooms, a webcam is going to be set up in order to look directly at the doctor's face. The webcam will provide the video of the doctor which will be inputted into the gaze estimator to obtain the gaze direction of the doctor and *PoR*.

3.2 Gaze Estimation

For the gaze estimation task, the *OpenGaze* software toolkit proposed in [37] is going to be used. The choice comes from the fact that the toolkit does not require any calibration sequence and provides both 3D and 2D gaze estimation. Additionally, as seen in [37], the performance of the state-of-the-art appearance-based methods can be competitive with the commercial eye trackers.

The *OpenGaze* toolkit offers a full pipeline for gaze estimation, which takes a webcam feed, video or image, and outputs the 3D gaze direction of the subject in the picture. It can also project the 3D gaze directions into *PoR* estimations in the screen coordinate system when the intrinsic and extrinsic camera parameters are given. The full *OpenGaze* pipeline is illustrated in Figure 3.2.

Figure 3.2: *OpenGaze* pipeline from [37]

The *OpenGaze* pipeline integrates the OpenFace 2.0 [3] for facial landmark detection, which will be used to estimate the 3D head pose. The 3D head pose will be used for the data normalization task, which consists of warping the image such that the variations in eye appearance caused by different head poses are mostly cancelled. Using the normalized data, the MPIIFaceGaze model will be used to estimate the 3D gaze direction, which will be optionally converted into a projection on the screen plane in the world coordinate system.

To project the 3D gaze direction onto a screen, the camera intrinsic and extrinsic parameters need to be calculated. Therefore, intrinsic and extrinsic camera calibrations need to be performed. The intrinsic calibration consists of finding the camera's intrinsic parameters, this will be done with the methods offered by the Python library, *OpenCV* [6]. The extrinsic calibration consists of finding the relative rotation and translation between the camera coordinate system and the world coordinate system. This can be achieved through a mirror-based extrinsic calibration method, specifically, we are going to use the method from [32].

3.3 Methodology

As said in 3.1, both presential and virtual consultations are going to be analyzed, while presential consultations will only occur in the hybrid consultation room, virtual consultations can occur in either type of consultation room. In this section we discuss how we will acquire and process the data.

3.3.1 Data Acquisition

The protocol for acquiring was created with the intent of removing as much bias from the data as possible, such that the data analyzed would resemble a normal consultation as closely as possible.

One source of bias comes from the fact that the doctor's behaviour is conditioned by the knowledge that he is being recorded. By being aware of the study the doctor will alter his normal behaviour introducing unwanted bias into the data.

Another possible source of bias is when the doctor has his first interaction with the patient (first consultation). In a first consultation, the doctor alters his behaviour mainly due to not knowing the patient, the first consultation is normally longer and the doctor has a more inquisitive role so that he learns about the patient's prior health history. Additionally, it serves to form a relationship between the patient and doctor.

Therefore to remove any possible bias in the doctor's behaviour during the recorded consultations we are only going to analyze a subset of the available captured data. The hope is that the doctor's behaviour will not be conditioned, since he does not know which of the consultations are going to be the target of analysis. Additionally, we will only analyze follow up consultations to remove the influence of the first consultation procedures from the data.

3.3.2 Data Processing

The processing of the doctor's video from the consultation differs slightly for virtual and presential consultation. Additionally, in the case of a virtual consultation performed in the hybrid consultation room, we need to have some additional processing.

During presential consultations, the *OpenGaze* toolkit will be used to estimate the *PoR* of the doctor's gaze on the screen. In this case, to quantify the percentage of time the doctor looks at the patient, we

classify the frames in which the doctor is not looking at the screen (when *OpenGaze* cannot estimate *PoR* on the screen) as moments in which he is looking at the patient.

Virtual consultations have 2 distinct cases, one for each type of consultation room, demanding different approaches. For virtual consultations in the virtual consultation room, the process is largely the same as for presential consultations. However, in this case, the frames in which the doctor is looking at the screen are classified as the moments in which the doctor looked at the patient.

For virtual consultations in the hybrid consultation room, there is a need for additional input to the process. Since the image of the patient can move inside the doctor's screen (*Picture-in-Picture*), in addition to estimating the *PoR* on the screen, we need to know where the patient's image is located on the doctor's screen. For this we are going to use the OpenFace 2.0 [3] to detect the face on the doctor's screen. With the *PoR* and the location of the patient's face within the screen, the frames are classified according to whether they coincide or not. When they coincide, we consider the doctor is looking at the patient, otherwise, he is looking at the health records of the patient, if *OpenGaze* doesn't give a *PoR*, we consider the doctor is not looking at the health records nor the patient.

3.4 Timeline

The following chart (Figure 3.3) has a timeline that divides the proposed approach in the tasks that will be performed throughout the available time. In addition to the chart, an explanation of what each task entails will be given.

Task Name	Start	Finish
Master Thesis	**Fri 10-09-21**	**Mon 30-05-22**
Software Implementation and Testing	Fri 10-09-21	Fri 29-10-21
Pilot Testing	Fri 01-10-21	Fri 29-10-21
Gaze Data Collection	Mon 01-11-21	Mon 31-01-22
Gaze Data Analysis	Tue 01-02-22	Thu 03-03-22
Results Discussion	Fri 04-03-22	Thu 31-03-22
Write Thesis	Fri 04-03-22	Mon 30-05-22

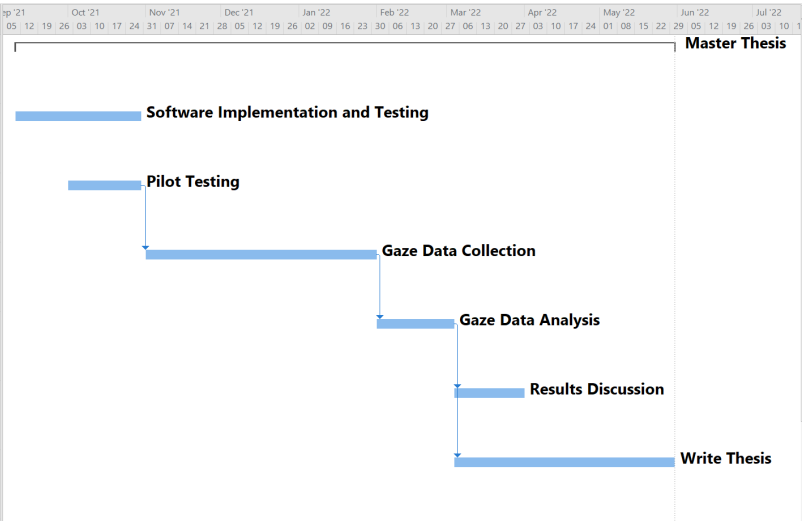


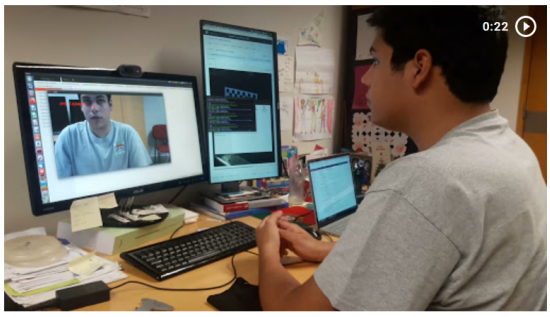
Figure 3.3: Masters Thesis planning.

- **Software Development and Testing**: Develop and test the software that will be used for gaze estimation using *OpenGaze* and the camera calibration methods mentioned.
- **Pilot Testing**: Test the Software on a pilot environment before moving on to real consultations.
- **Gaze Data Collection**: Use the Software to collect the gaze data of the physician during consultations.

- **Gaze Data Analysis**: Analyze the collected gaze data, classifying it in function of whether the physician is looking at the patient or not
- **Results Discussion**: Analyze the results obtained and perform more tests to deepen the insights taken from this work.
- **Write Thesis**: Writing of the thesis, starting when some chapters are concluded and gradually becoming more important.

3.5 Preliminary Experiments

As preliminary experiments, we used *OpenGaze* software to estimate our gaze. It was used both in a 2D gaze estimation task and in a 3D gaze estimation task. The videos of both experiments are available at `https://drive.google.com/drive/folders/1oMrk7EI-0LiNCp7KWTLXmHob3l9dDZAC?usp=sharing`.



(a) 3D Gaze Estimation

(b) 2D Gaze Estimation

Figure 3.4: Preliminary Experiments.

In the 3D gaze estimation task, the software performed well. However, if the subject is using a mask the software is not able to perform the face and facial landmark detection of the subject correctly, leading to inaccurate estimates or no estimates at all, if the face detection fails. In the 2D gaze estimation task, the software presented some deviation from the correct position. This deviation is mainly due to errors made during the extrinsic calibration procedure which leads to a poor projection of the 3D gaze direction onto the screen.

Bibliography

[1] On the importance of nonverbal communication in the physician–patient interaction. *Patient Education and Counseling*, 67(3):315–318, 2007. ISSN 0738-3991. doi: https://doi.org/10.1016/j.pec.2007.03.005. EACH Conference Basel 2006.

[2] S. Albawi, T. A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

[3] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 59–66, 2018. doi: 10.1109/FG.2018.00019.

[4] R. S. Beck, R. Daughtridge, and P. D. Sloane. Physician-patient communication in the primary care office: a systematic review. *The Journal of the American Board of Family Medicine*, 15(1):25–38, 2002. ISSN 1557-2625. URL `https://www.jabfm.org/content/15/1/25`.

[5] J. Bird and S. Cohen-Cole. The three-function model of the medical interview. an educational device. *Advances in psychosomatic medicine*, 20:65—88, 1990. ISSN 0065-3268. URL `http://europepmc.org/abstract/MED/2239506`.

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] D. Cazzato, A. Evangelista, M. Leo, P. Carcagnì, and C. Distante. A low-cost and calibration-free gaze estimator for soft biometrics: An explorative study. *Pattern Recognition Letters*, 82, 11 2015. doi: 10.1016/j.patrec.2015.10.015.

[8] X. Cha, X. Yang, Z. Feng, T. Xu, X. Fan, and J. Tian. Calibration-free gaze zone estimation using convolutional neural network. pages 481–484, 12 2018. doi: 10.1109/SPAC46244.2018.8965441.

[9] J. Chen and Q. Ji. 3d gaze estimation with a single camera without ir illumination. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, 2008. doi: 10.1109/ICPR.2008.4761343.

[10] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL `http://arxiv.org/abs/1703.03400`.

[11] T. Fischer, H. J. Chang, and Y. Demiris. RT-GENE: Real-Time Eye Gaze Estimation in Natural Environments. In *European Conference on Computer Vision*, pages 339–357, September 2018.

[12] K. Funes Mora, F. Monay, and J.-M. Odobez. Eyediap: a database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras. pages 255–258, 03 2014. doi: 10.1145/2578153.2578190.

[13] A. George and A. Routray. Real-time eye gaze direction classification using convolutional neural network, 2016.

[14] D. Gutstein, E. Montague, J. Furst, and D. Raicu. Optical flow, positioning, and eye coordination: Automating the annotation of physician-patient interactions. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 943–947. IEEE, 2019.

[15] J. A. Hall, J. A. Harrigan, and R. Rosenthal. Nonverbal behavior in clinician—patient interaction. *Applied and Preventive Psychology*, 4(1):21–37, 1995. ISSN 0962-1849. doi: https://doi.org/10.1016/S0962-1849(05)80049-6. URL https://www.sciencedirect.com/science/article/pii/S0962184905800496.

[16] Y. Hart, E. Czerniak, O. Karnieli-Miller, A. E. Mayo, A. Ziv, A. Biegon, A. Citron, and U. Alon. Automated video analysis of non-verbal communication in a medical setting. *Frontiers in Psychology*, 7:1130, 2016. ISSN 1664-1078. doi: 10.3389/fpsyg.2016.01130. URL https://www.frontiersin.org/article/10.3389/fpsyg.2016.01130.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[18] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[19] P. Kellnhofer, A. Recasens, S. Stent, W. Matusik, , and A. Torralba. Gaze360: Physically unconstrained gaze estimation in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[20] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba. Eye tracking for everyone. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2176–2184, 2016.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[23] G. Liu, Y. Yu, K. A. F. Mora, and J. Odobez. A differential approach for gaze estimation. *CoRR*, abs/1904.09459, 2019. URL http://arxiv.org/abs/1904.09459.

[24] M. Mast and G. Cousin. *The role of nonverbal communication in medical interactions: Empirical results, theoretical bases, and methodological issues*, pages 38–53. 01 2013.

[25] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

[26] S. Park, X. Zhang, A. Bulling, and O. Hilliges. Learning to find eye region landmarks for remote gaze estimation in unconstrained settings. In *ACM Symposium on Eye Tracking Research and Applications (ETRA)*, ETRA '18, New York, NY, USA, 2018. ACM.

[27] S. Park, S. D. Mello, P. Molchanov, U. Iqbal, O. Hilliges, and J. Kautz. Few-shot adaptive gaze estimation. In *International Conference on Computer Vision (ICCV)*, 2019.

[28] S. Park, E. Aksan, X. Zhang, and O. Hilliges. Towards end-to-end video-based eye-tracking. In *European Conference on Computer Vision (ECCV)*, 2020.

[29] W. Rawat and Z. Wang. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29(9):2352–2449, 09 2017. ISSN 0899-7667. doi: 10.1162/neco_a_00990. URL https://doi.org/10.1162/neco_a_00990.

[30] B. Smith, Q. Yin, S. Feiner, and S. Nayar. Gaze locking: Passive eye contact detection for human-object interaction. pages 271–280, 10 2013. doi: 10.1145/2501988.2501994.

[31] Y. Sugano, Y. Matsushita, and Y. Sato. Learning-by-synthesis for appearance-based 3d gaze estimation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1821–1828, 2014. doi: 10.1109/CVPR.2014.235.

[32] K. Takahashi, S. Nobuhara, and T. Matsuyama. A new mirror-based extrinsic camera calibration using an orthogonality constraint. In *Proc. of CVPR*, 2012.

[33] T. Tan, E. Montague, J. Furst, and D. Raicu. Robust physician gaze prediction using a deep learning approach. In *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 993–998, 2020. doi: 10.1109/BIBE50027.2020.00168.

[34] C. Zhang, R. Yao, and J. Cai. Efficient eye typing with 9-direction gaze estimation. *CoRR*, abs/1707.00548, 2017. URL http://arxiv.org/abs/1707.00548.

[35] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Appearance-based gaze estimation in the wild. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4511–4520, June 2015.

[36] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. It's written all over your face: Full-face appearance-based gaze estimation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 2299–2308. IEEE, 2017.

[37] X. Zhang, Y. Sugano, and A. Bulling. Evaluation of appearance-based methods and implications for gaze-based applications. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2019. doi: 10.1145/3290605.3300646.